

Rodina protokolů TCP/IP

verze 3

Téma 5: Protokol IPv4

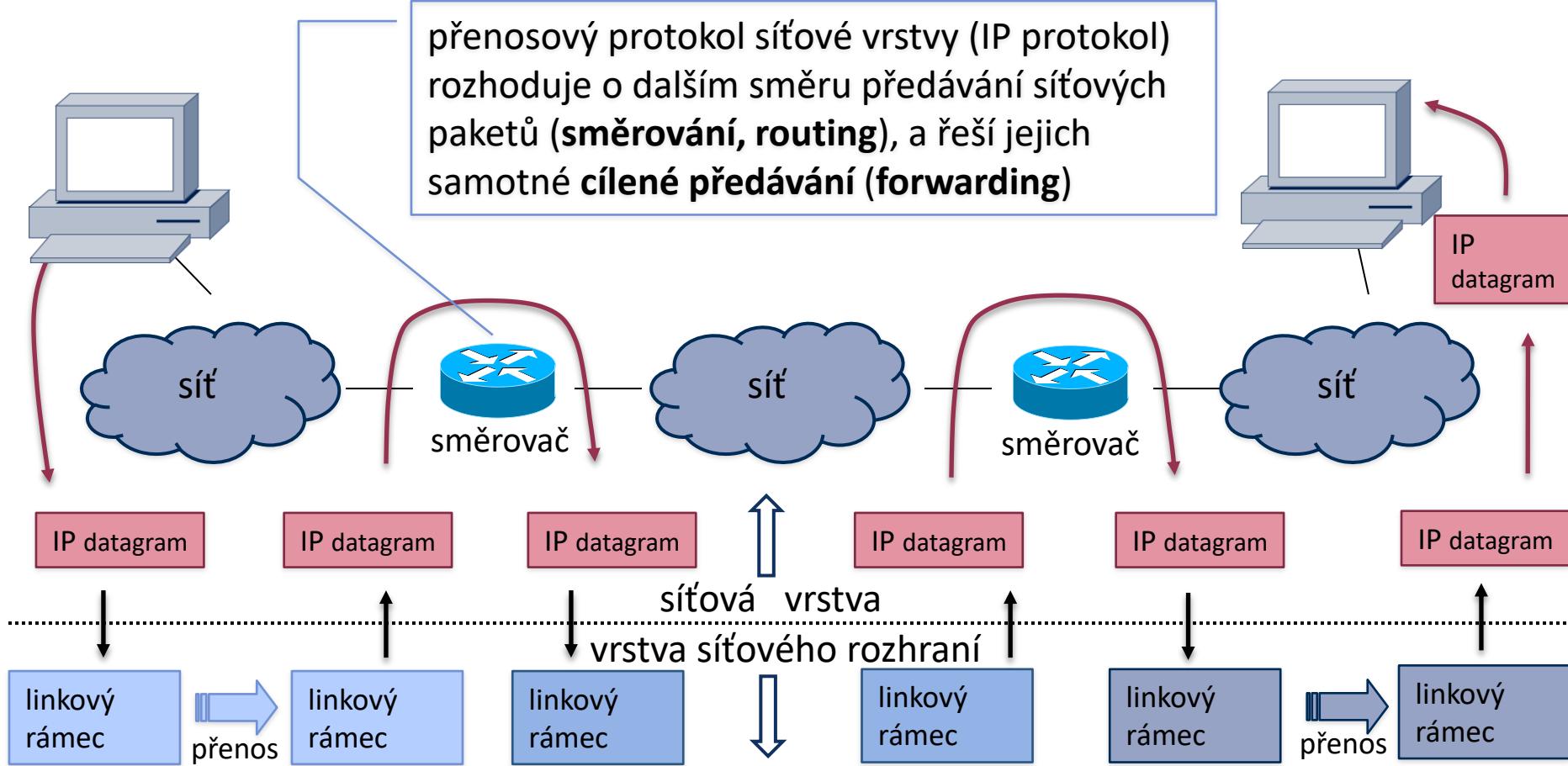
Jiří Peterka

charakteristika protokolu IPv4

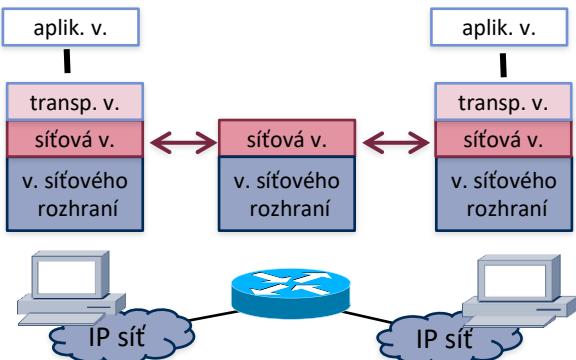
- je univerzální
 - snaží se fungovat „nad vším“
 - viz: IP over Everything
 - nevyužívá specifika přenosových technologií vrstvy síťového rozhraní
 - chce jen "společné minimum"
 - snaží se zakrýt odlišnosti
 - vytváří jednotné prostředí pro všechny aplikace („jednotnou pokličku“)
- pracuje s virtuálními pakety
 - nemají ekvivalent v HW, musí se zpracovávat v SW
 - říká se jim **IP datagramy**
 - protože se přenáší nespojovaně
 - zatímco u IPv6 se hovoří o paketech
 - mají proměnnou velikost
 - problém: může docházet ke fragmentaci
- je zaměřen na jednoduchost, efektivnost a rychlosť
 - je nespojovaný
 - nečísluje přenášené datagramy
 - negarantuje pořadí doručení
 - negarantuje dobu doručení
 - funguje jako nespolehlivý
 - negarantuje doručení
 - negarantuje nepoškozenost dat
 - nepoužívá potvrzení
 - nepodporuje řízení toku
 - je „síťově neutrální“
 - pracuje stylem best effort
 - ke všem datům se chová stejně
 - nepodporuje QoS

- protokol IP je jediným přenosovým protokolem síťové vrstvy
 - experimentální protokol ST se neujal a nepoužívá
- ale kromě něj patří do síťové vrstvy i další protokoly a mechanismy:
 - ICMP (Internet Control Message Protocol)
 - pro přenos zpráv týkajících se fungování protokolu IP („posel špatných zpráv“)
 - IGMP (Internet Group Management Protocol)
 - pro správu multicastových skupin
 - protokoly pro směrování
 - RIP (Routing Information Protocol)
 - OSPF (Open Shortest Path First)
 -
 - mechanismy překladu adres
 - NAT (Network Address Translation)
 - překlad z IP na IP
 - ARP
 - překlad z IP na HW adresu
- s fungováním síťové vrstvy úzce souvisí také:
 - vše kolem IP adres
 - a jejich využití, přidělování, distribuce
 - protokoly RARP, BOOTP, DHCP,
 - vše kolem směrování
 - systém DNS
 - překlad mezi doménovými jmény a IP
 - protokoly SLIP a PPP
 - byť patří do vrstvy síťového rozhraní

fungování IP jako síťového protokolu



- protokol IP je posledním protokolem (počítáno „odspodu“), který musí být implementován i ve vnitřních uzlech sítě
 - i ve směrovačích



IPv4 datagram a jeho formát

- datagram má dvě hlavní části:

- **hlavičku (header)**

- proměnné velikosti: minimum je 20 B, ale může být větší
 - je nutný explicitní údaj o délce hlavičky:
 - IHL (Internet Header Length), 4 bity, v násobcích 4 bytů

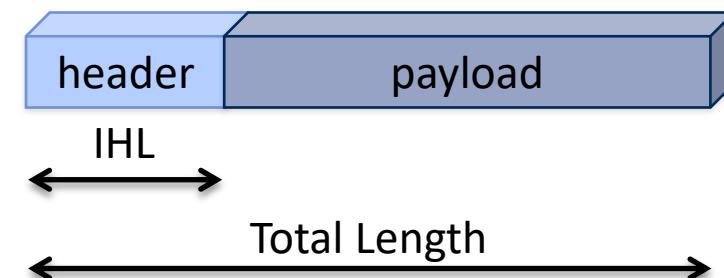
- **tělo, resp. datovou část (body, payload)**

- také proměnné velikosti
 - je nutný (další) údaj o velikosti
 - Total Length, 16 bitů, max. velikost 64 kB
 - zahrnuje jak tělo, tak i hlavičku

- datagram naopak nemá:

- patičku
 - s kontrolním součtem celého datagramu
 - kontrolní součet má pouze hlavička !!!
 - data v těle (nákladové části) nejsou kryta kontrolním součtem
 - jejich integritu si musí zajistit „ten, komu data patří“ (protokol vyšší vrstvy)

možnost rozšíření se využívá jen zřídka,
proto hlavička má obvykle jen 20 bytů

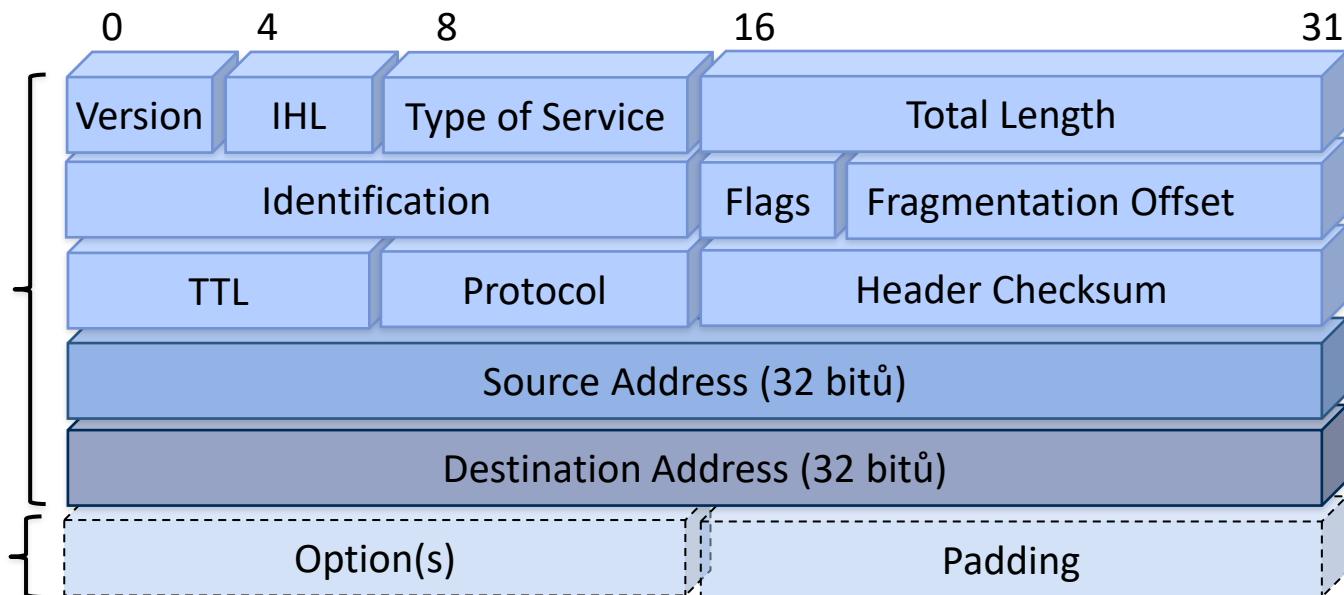


v praxi bývá podstatně menší, kvůli velikosti linkového rámce

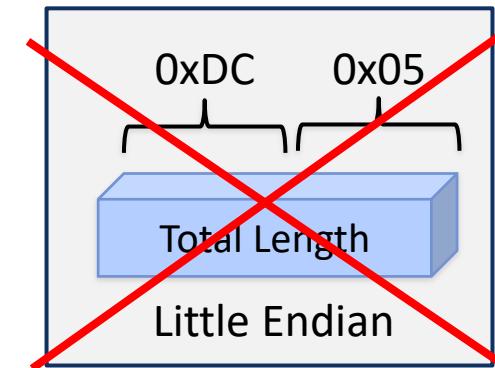
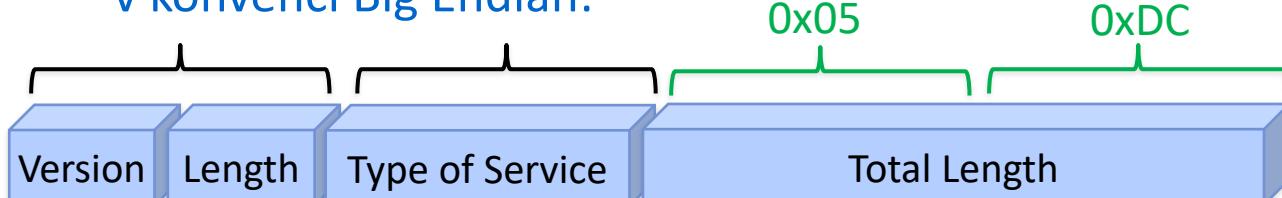
hlavička IPv4 datagramu

- má celkem 14 položek, z nichž:

- 13 je povinných
 - dohromady mají velikost 20 bytů
 - minimální, současně obvyklá velikost hlavičky
- 1 je volitelná
 - doplňky (Options)

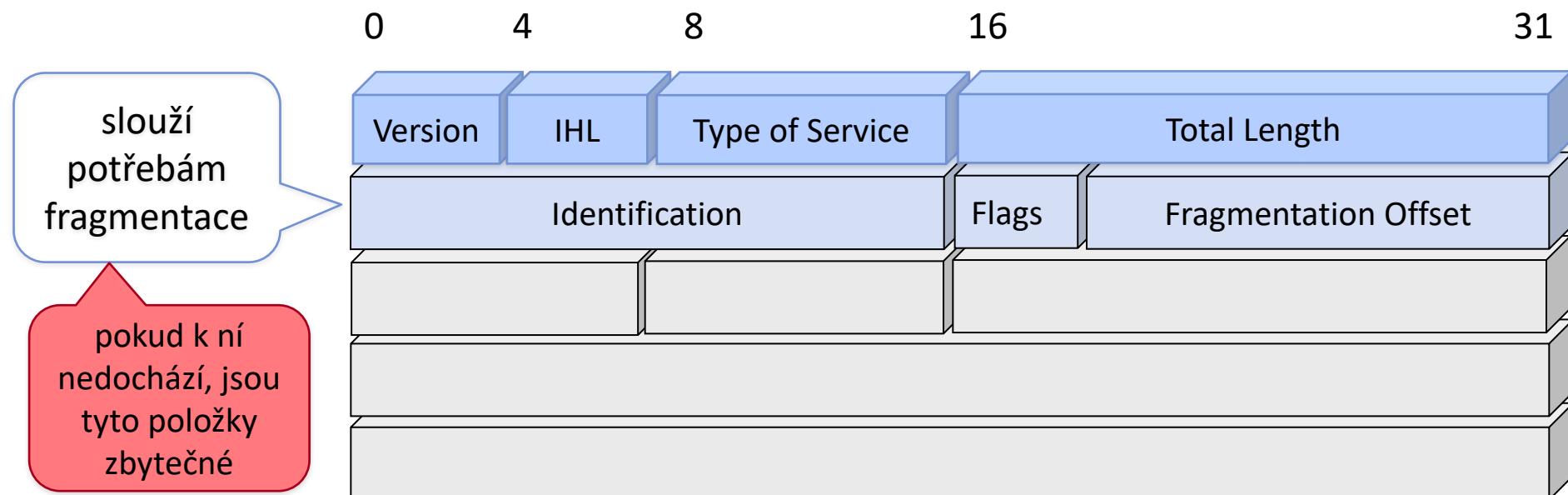


- údaje jsou do všech položek zapisovány podle konvence Big Endian
 - tj. „nejvýznamnější byte nejdříve“
- příklad: IP datagram velikosti 1500 bytů (0x5DC)
 - v konvenci Big Endian:



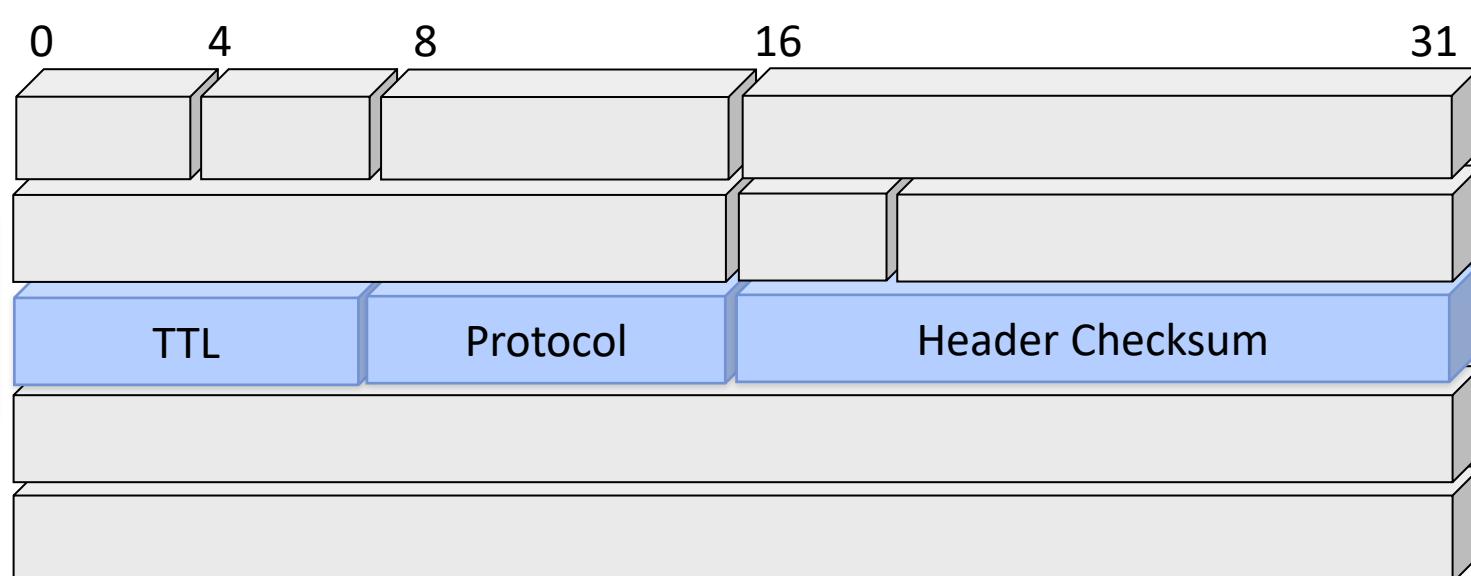
formát hlavičky IPv4 datagramu

- **Version**, 4 bity
 - dnes = 4 (IPv4)
- **IHL** (Internet Header Length), 4 bity
 - **velikost hlavičky v jednotkách 32-bitů**
 - při minimální/typické velikosti hlavičky (bez rozšíření) je IHL = 5
- **Total Length**, 16 bitů
 - celková délka datagramu v bytech, včetně hlavičky!
- **Type of Service (TOS)**, 8 bitů
 - „*zapomenutý byte*“
 - jeho původní význam dnes již není znám
 - bylo definováno několik nových významů,
 - hlavně pro potřebu podpory QoS



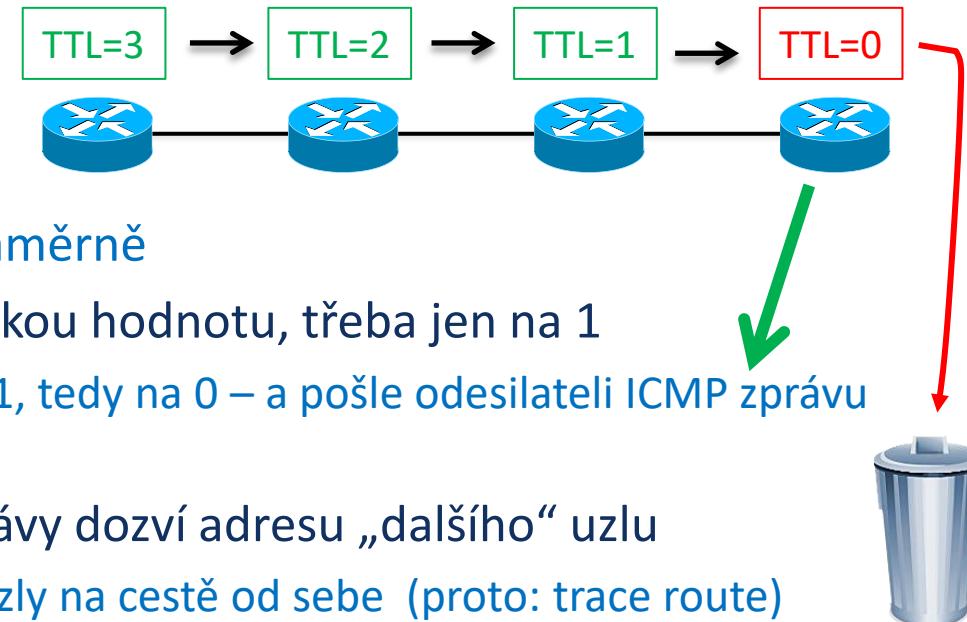
formát hlavičky IPv4 datagramu

- **TTL** (Time To Live), 8 bitů
 - původně se mělo jednat o časový údaj, dnes je využíváno jako počet přeskoků
- **Protocol**, 8 bitů
 - udává typ dat v těle (nákladové části) datagramu
 - např.: 1=ICMP, 6=TCP, 17=UDP
 - konvence o hodnotách je společná s IPv6, spravuje ji organizace IANA, zveřejňuje na <http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml>
- **Header Checksum**, 16 bitů
 - kontrolní součet hlavičky (nikoli CRC)



položka TTL

- položka **TTL** chrání před zacyklením, funguje jako (klesající) čítač
 - tzv. hop count (v IPv6 se jmenuje: **Hop Limit**)
 - odesilatel nastaví tuto položku na určitou počáteční hodnotu
 - při každém průchodu směrovačem se hodnota této položky sníží o 1
 - pozor: kvůli tomu je nutné přepočítávat kontrolní součet hlavičky !!!!
 - pokud hodnota TTL klesne na 0, má směrovač právo datagram zahodit
 - má právo myslet si, že došlo k zacyklení
 - ale: má povinnost poslat o tom zprávu odesilateli datagramu
 - pomocí protokolu ICMP
 - ICMP zprávu Time Exceeded
- další využití (traceroute):
 - vynulování položky TTL lze navodit záměrně
 - počátečním nastavením TTL na nízkou hodnotu, třeba jen na 1
 - nejbližší další směrovač sníží TTL o 1, tedy na 0 – a pošle odesilateli ICMP zprávu Time Exceeded
 - odesilatel datagramu se z této zprávy dozví adresu „dalšího“ uzlu
 - takto může „vystopovat“ všechny uzly na cestě od sebe (proto: trace route)



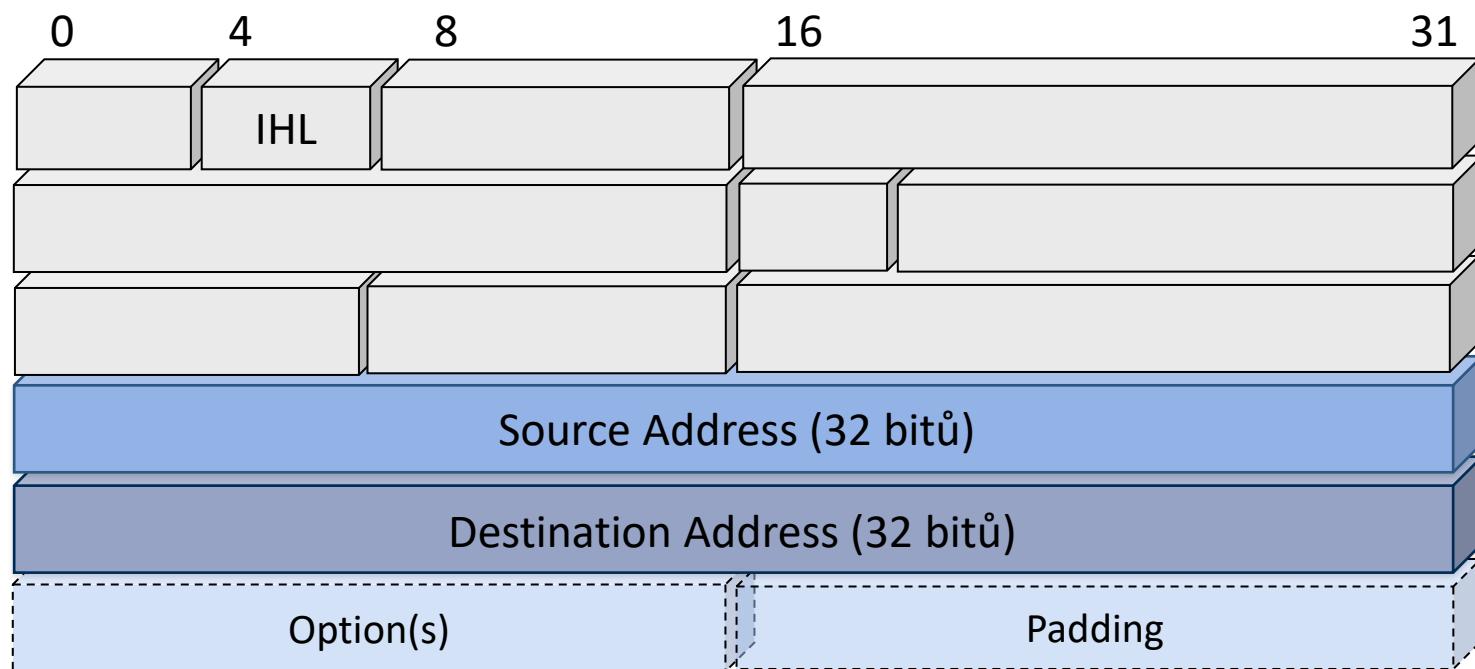
položka Header Checksum

- položka Header Checksum zajišťuje integritu hlavičky
 - umožňuje detekovat případnou změnu obsahu hlavičky
- výpočet kontrolního součtu:
 - hlavička se interpretuje jako posloupnost 16-bitových slov
 - samotná položka Header Checksum se do výpočtu nezahrnuje
 - k součtu se připočítají přetoky a udělá se z něj jedničkový doplněk
 - tj. invertují se jednotlivé bity součtu
 - výsledek (16 bitů) se zapíše do položky Header Checksum
- problém (zatěžující implementaci IPv4):
 - obsah položky (hodnota kontrolního součtu) se musí přepočítávat
 - při každé změně položky TTL (tj. při každém průchodu směrovačem)
 - při každém překladu adres (NAT)
- ověření kontrolního součtu
 - počítá se včetně hodnoty položky Header Checksum
 - jinak je postup stejný
 - pokud vyjde 0, nedošlo ke změně
 - jiná hodnota = došlo ke změně
 - datagram může/musí být zahozen
 - ale: **neodesílá se žádná ICMP zpráva**
 - protože není záruka toho, že by došla správnému odesilateli
 - kvůli možnému poškození adresy odesilatele

formát hlavičky IPv4 datagramu

- **Source Address, Destination Address**, á 32 bitů
 - IPv4 adresy odesilatele a (koncového) příjemce
- **Options**, různá velikost (od 1 bytu výše)
 - volitelné doplňky
 - mohou mít různou velikost, nemusí být zarovnány na celé násobky 32 bitů
 - jak vyžaduje konstrukce položky IHL (Internet Header Length)
 - která počítá s délkou, která je násobkem 32 bitů (4 bytů)
- **Padding**
 - případné dorovnání hlavičky do celistvého násobku 32 bitů

dnes se
v praxi moc
nepoužívají

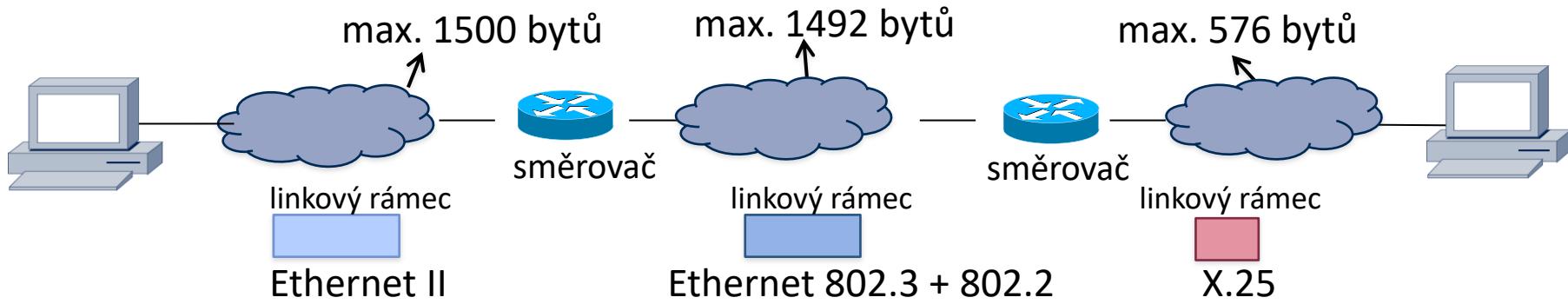
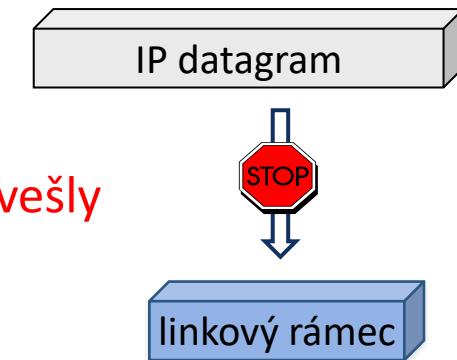


- mají vlastní (strukturovaný) formát
 - zahrnuje:
 - typ doplňku (Option Type), 1 byte
 - délku doplňku (Option Length), žádný nebo 1 byte
 - data doplňku (Option Data), žádný nebo více bytů
 - příklady doplňků
 - **Record Route**
 - zaznamenává, kudy datagram prochází
 - každý směrovač, přes který datagram prochází, vloží do jeho hlavičky svou IP adresu
 - **Timestamp**
 - zaznamenává čas průchodu jednotlivými směrovači
 - **Source Routing**
 - v hlavičce IP datagramu je vložena cesta (posloupnost směrovačů)
 - *Strict Source Route*: posloupnost směrovačů musí být přesně dodržena
 - *Loose Source Route*: na cestě mezi předepsanými směrovači může datagram přecházet i přes jiné směrovače
 - ale musí projít všemi směrovači na „source route“

(pravděpodobný) smysl doplňků:
aby bylo možné měnit způsob, jakým protokol IP standardně nakládá s datagramy

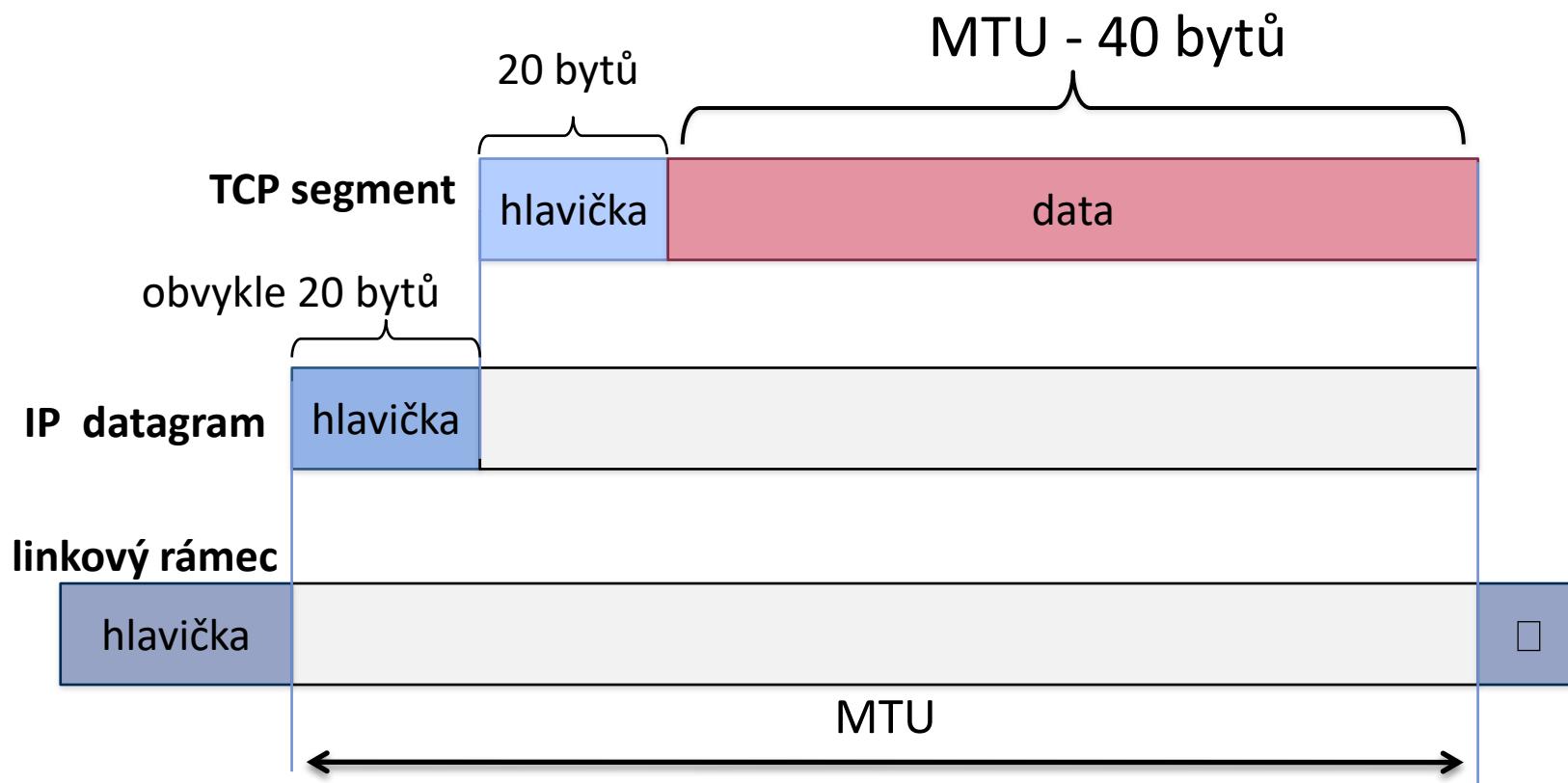
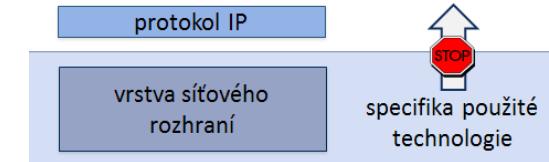
problém fragmentace

- prvotní příčina:
 - technologie vrstvy síťového rozhraní pracují s linkovými rámcemi omezené velikosti
 - do kterých se IP datagram nemusí vejít!!
- řešení:
 - dělat IP datagramy jen tak velké, aby se do linkových rámců vešly
- problém:
 - ne vždy je to možné (volit IP datagram dostatečně malý)
 - o velikosti IP datagramu rozhoduje:
 - protokol TCP, pokud jde o data přenášená tímto protokolem
 - aplikace, pokud jde o data přenášená protokolem UDP
 - „po cestě“ mohou být IP datagramy vkládány do linkových rámců různé velikosti
 - různé linkové technologie pracují s různě velkými rámcemi !!!



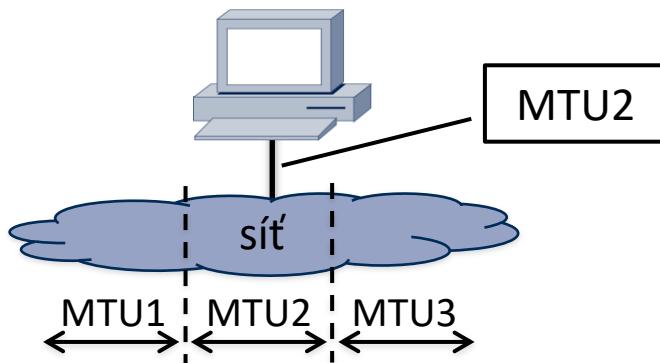
MTU: Maximum Transmission Unit

- protokol IP zakrývá všechna specifika linkových technologií
 - výjimkou je informace o velikosti (nákladové části) jejich rámce
 - skrze parametr **MTU** (Maximum Transmission Unit)
 - tuto informaci se dozvídá jak protokol TCP, tak i aplikace
 - podle ní mohou „porcovat“ svá data
 - ovšem ani to nemusí stačit – viz různá MTU „po cestě“



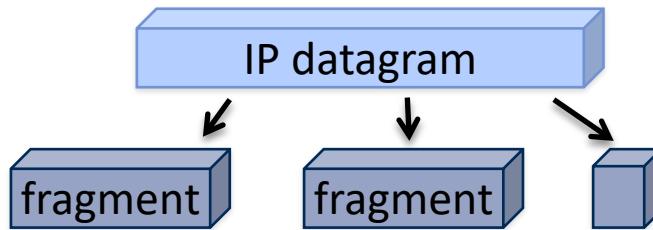
dosah MTU

- hodnota parametru MTU se vztahuje pouze:
 - k danému rozhraní
 - důležité pro směrovače, každé jejich rozhraní může mít jiné MTU !!
 - k místnímu (linkovému) segmentu
 - různé linkové technologie (s různou velikostí linkového rámce) mohou být nasazeny i v jedné a téže síti
 - například:
 - Ethernet II: max. 1500 bytů
 - Ethernet 802.2+802.3: 1492 bytů
 - 802.11 (Wi-Fi): 7981 bytů



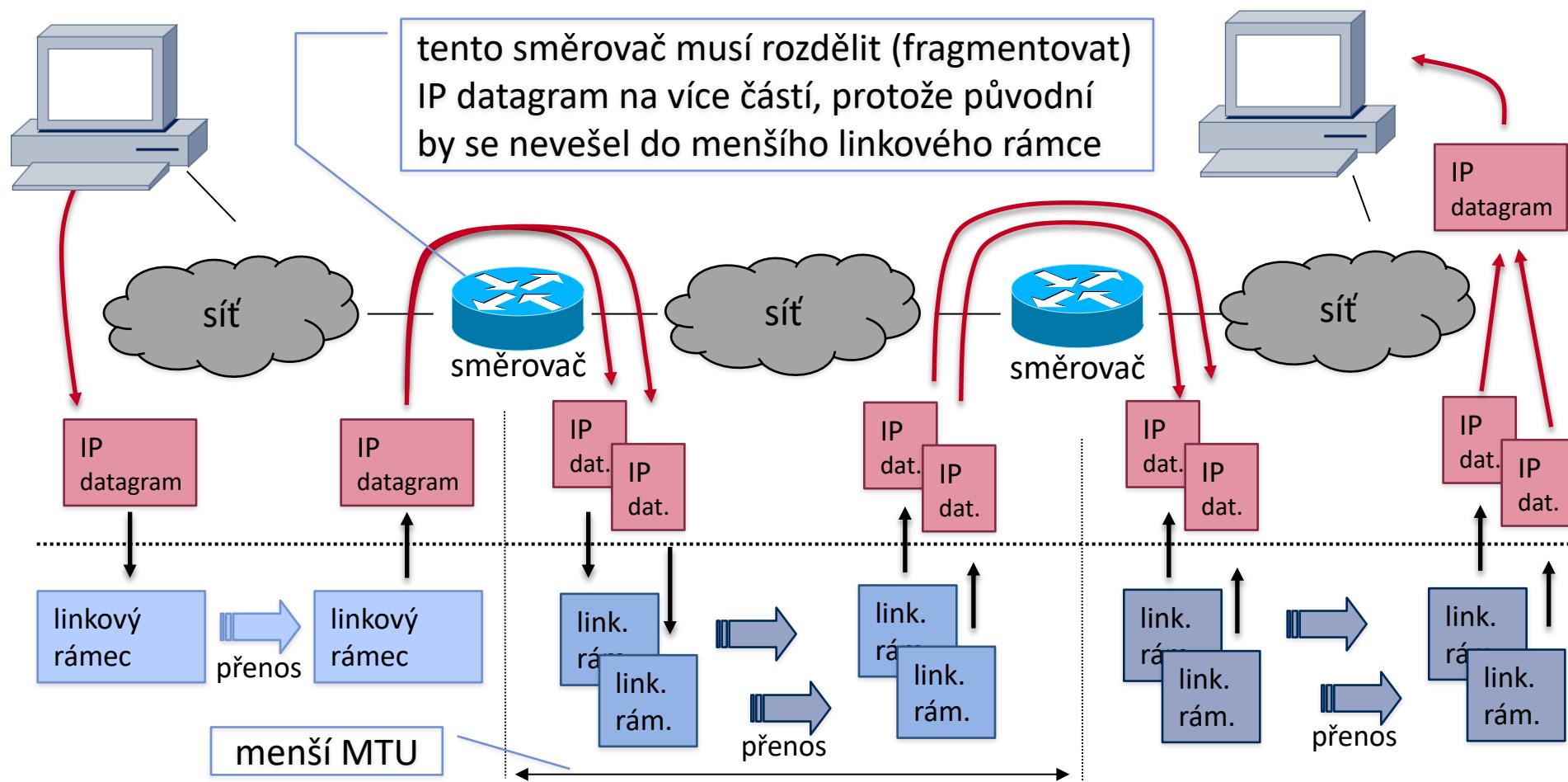
- Path MTU („MTU po celé cestě“)
 - fakticky: minimum přes všechna MTU od zdrojového uzlu až po cílový
 - dá se zjistit pomocí postupu zvaného Path MTU Discovery
 - ale: nemusí vždy fungovat správně
 - kvůli nespojovanému způsobu fungování protokolu IP
 - skutečná data mohou být přenášena jinou cestou
 - garantované minimum Path MTU:
 - IPv4: 68 bytů (RFC 791)
 - v praxi: 576 bytů
 - minimum, které musí zpracovat každý uzel (bez fragmentace)
 - IPv6: 1280 bytů

- možné strategie (odesilatelů):
 - generovat jen tak velké IP datagramy, které se vždy „vejdou“ do linkových rámců
 - IPv4: do 576 byte, IPv6: do 1280 byte
 - řídit se Path MTU
 - „nákladné“
 - kvůli nutnosti zjišťování
 - nemusí vždy stačit
 - kvůli nespojovanému způsobu fungování protokolu IP
 - řídit se jen „místním“ MTU
 - nemusí vždy stačit
 - kvůli menšímu MTU „po cestě“
 - neomezovat velikost IP datagramů
 - zvláště pokud není v silách

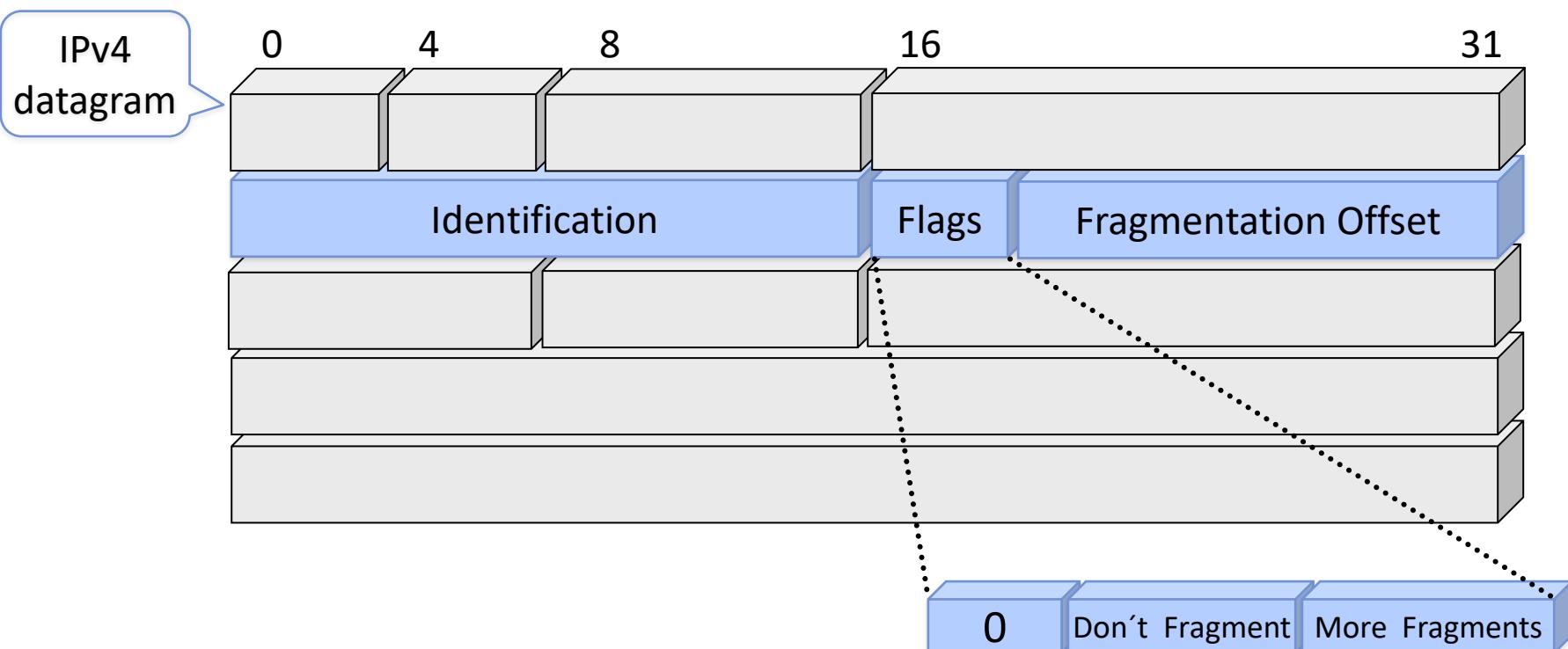
- podmínka:
 - je k dispozici možnost **fragmentace**
 - rozdelení „příliš velkého“ IP datagramu na menší datagramy
 - označované jako **fragmenty**
 - které se „již vejdou“ do linkových rámců
 - a mohou se přenášet samostatně
- A diagram illustrating the fragmentation process. At the top, a large blue rectangular box is labeled "IP datagram". Three arrows point downwards from the right side of this box to three smaller blue rectangular boxes below it, each labeled "fragment".
- fungování:**
 - IPv4:
 - fragmentovat může odesírající uzel i kterýkoli směrovač „po cestě“
 - IPv6:
 - fragmentuje jen odesírající uzel

de-fragmentace

- jednotlivé fragmenty skládá zpět (do původního datagramu) vždy až jejich koncový příjemce !!!
 - žádný jiný uzel to dělat nemůže
 - nemusí mít k dispozici všechny fragmenty (mohou být přenášeny mimo něj)

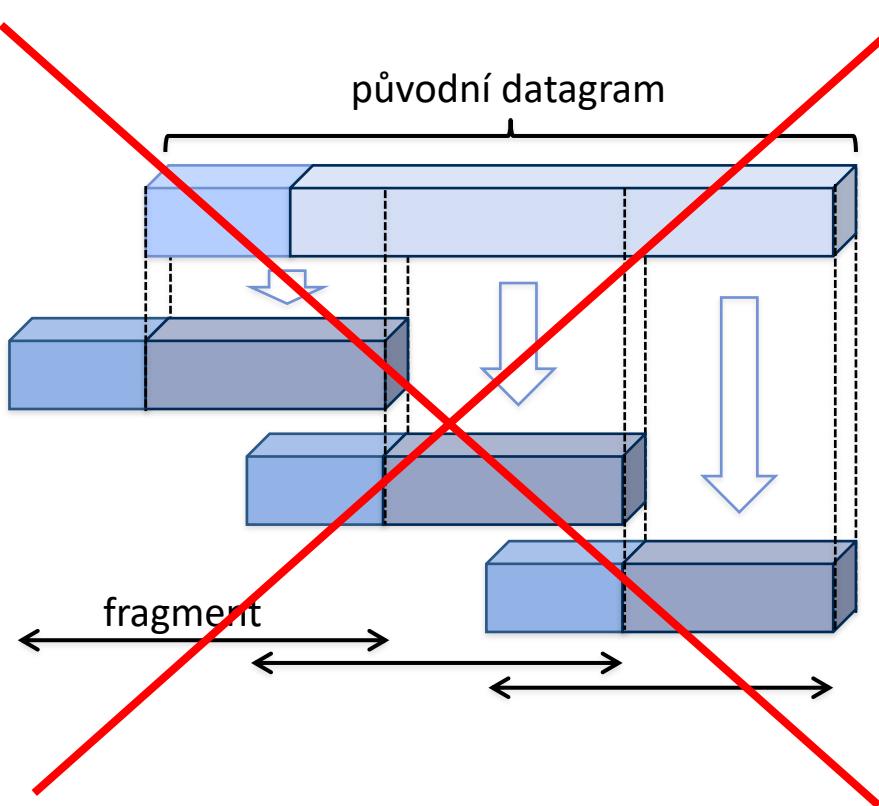


- podmínkou pro fragmentaci je podpora v protokolu IP
 - IPv6 ji řeší pomocí rozšiřujících hlaviček
 - které připojuje k základní hlavičce až v případě potřeby
 - až když se skutečně fragmentuje
 - IPv4 ji řeší položkami, které jsou přítomné v každé hlavičce
 - a tudíž jsou zbytečné (a zabírají místo) tam, kde k fragmentaci nedochází

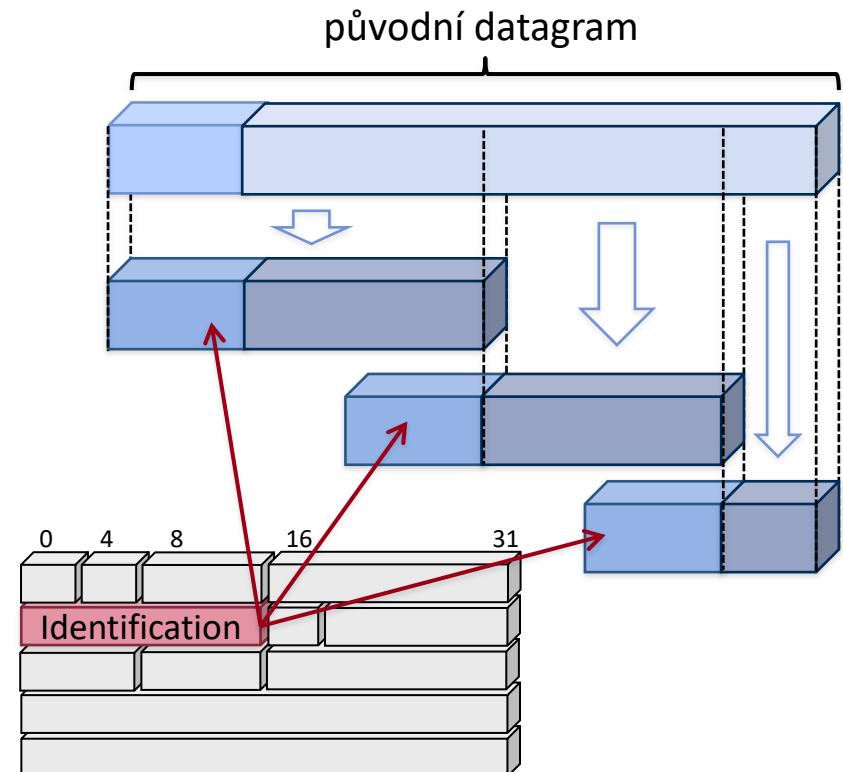


způsob fragmentace v IPv4

- fragmentace nepředstavuje zapouzdření původního IP datagramu



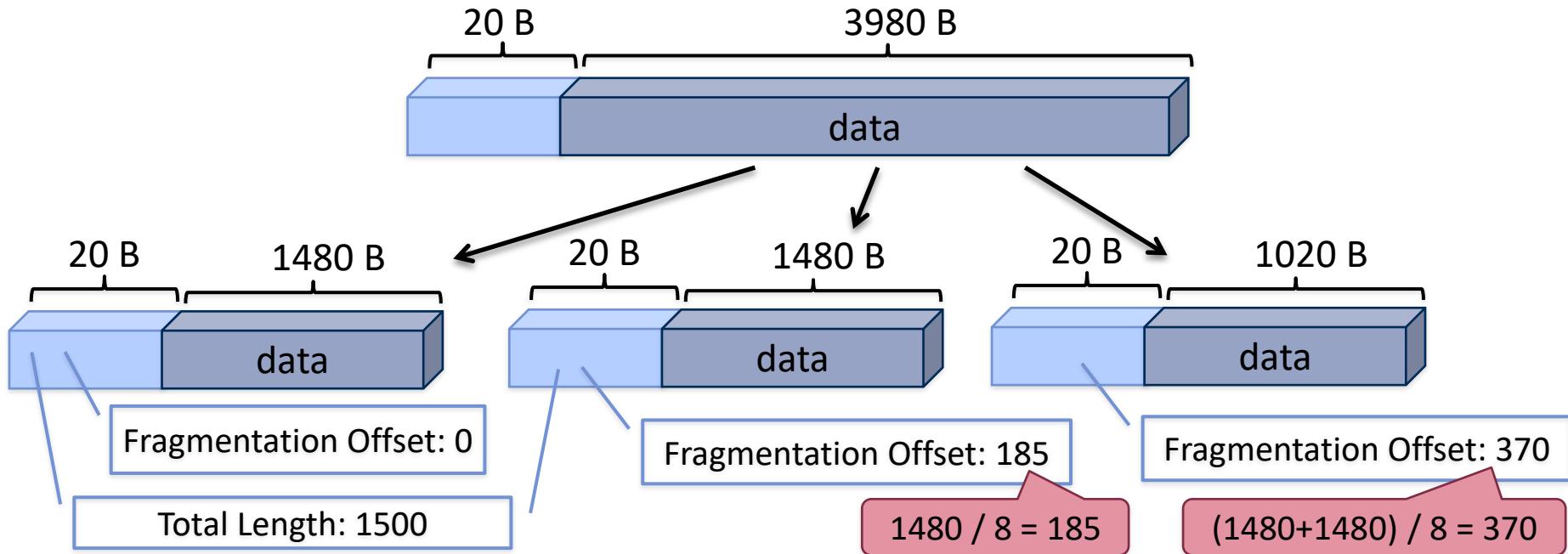
- ale překlad (transformaci) původního datagramu



- všechny fragmenty mají v položce **Identification** (16 bitů) stejnou hodnotu jako původní datagram
- tím se pozná, že „patří k sobě“

způsob fragmentace v IPv4

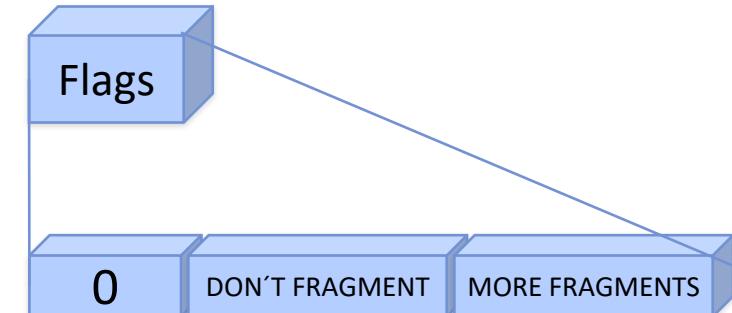
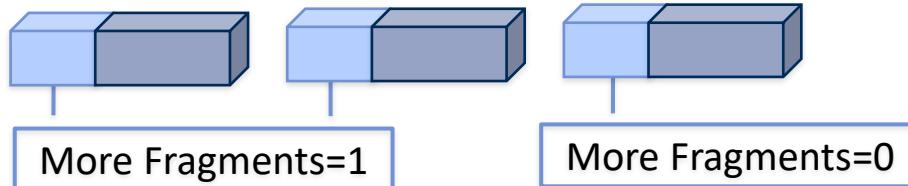
- položka:
 - **Fragmentation Offset, 13 bitů (nikoli 16 !!!)**
 - udává offset (posun) začátku datové části fragmentu oproti datové části původního datagramu
 - v násobcích 8 byte (64 bitů)
 - proto musí být velikost fragmentů zaokrouhlena na celistvé násobky 8 byte
 - příklad: IP datagram o velikosti 4000 B, hlavička bez doplňků (tj. 20 B)
 - je třeba jej vložit do linkového rámce s MTU=1500 byte



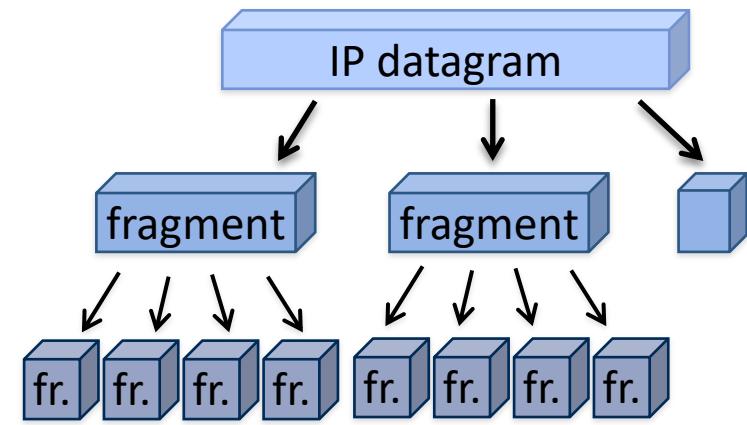
způsob fragmentace v IPv4

- příznaky fragmentace (Flags):

- Don't Fragment, 1 bit**
 - požadavek na to, aby datagram nebyl fragmentován, i když by to bylo zapotřebí
 - v jeho přenosu pak ale nelze pokračovat
 - musí být zahozen
 - odesilateli datagramu je zaslána ICMP zpráva Destination Unreachable
 - někdy je tento stav vyvoláván záměrně, pro potřeby hledání Path MTU (nejmenšího MTU „po cestě“)
- More Fragments, 1 bit**
 - příznak, udávající zda jde o poslední fragment
 - 1 = je to poslední
 - 0 = je to poslední



- fragmentaci lze opakovat
 - fragmenty lze dále fragmentovat
 - pokud se opět nevejdou do linkového rámce



- u doplňků (options) není jasné, jak s nimi naložit
 - každý doplněk má příznak, který specifikuje zda daný doplněk má být zkopirován i do jednotlivých fragmentů, nebo nikoli
- připomenutí:
 - (u IPv4): fragmentovat může jak odesírající uzel, tak kterýkoli směrovač „po cestě“
 - ale zpětné sestavení původního datagramu z fragmentů („de-fragmentaci“) provádí **až koncový příjemce !!!!**
- problém:
 - zpětné sestavování (de-fragmentace) je složité a časově náročné
 - koncový příjemce musí čekat určitou dobu, zda dostane všechny fragmenty
 - mohou mu přicházet v různém pořadí, s různým zpožděním
 - musí si je ukládat do vhodného bufferu a volit vhodnou dobu čekání
 - je to ve sporu s celkovým stylem fungování protokolu IP
 - ten funguje bezestavově, nemá (jinak) žádné časové limity, čekání atd.
 - pokud příjemci nepřijdou (do zvoleného časového limitu) všechny fragmenty, musí všechny dosud přijaté fragmenty zahodit
 - a odesilateli pošle ICMP zprávu **Time Exceeded**

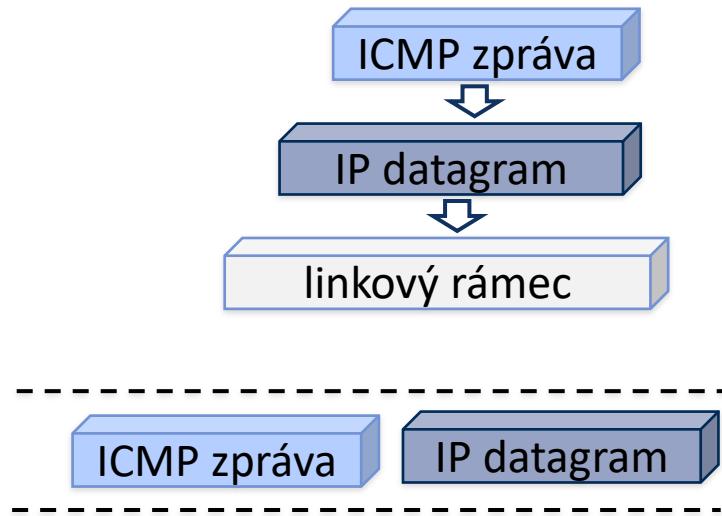
- protokol IP je velmi jednoduchý a přímočarý
 - postrádá:
 - mechanismy pro signalizaci (hlášení) chyb a nestandardních situací
 - například zahodení datagramu, nesprávné směrování, přetížení,
 - testování a další „speciální úkoly“
 - proto:
 - k protokolu IP byl vyvinut „doplňkový“ protokol
 - ICMP: Internet Control Message Protocol
 - který přenáší zprávy
 - tzv. ICMP zprávy
 - je povinnou součástí (implementace) protokolu IP
 - je součástí síťové vrstvy
 - tudíž musí být implementován i ve směrovačích
 - které také generují ICMP zprávy nejčastěji
 - protokol IPv4 má vlastní protokol ICMP (ICMPv4)
 - protokol IPv6 také: ICMPv6
- příklady ICMP zpráv:
 - Time Exceeded
 - vypršený čas
 - Destination Unreachable
 - nedosažitelný cíl
 - Source Quench
 - hrozí zahlcení
 - Redirect
 - přesměrování
 - Echo Request/Reply
 - testování dostupnosti
 -

jak se přenáší ICMP zprávy?

- původně:
 - ICMP zprávy měly generovat jen směrovače
 - dnes:
 - ICMP právy generují i hostitelské počítače
 - dosah:
 - ICMP zprávy nejsou omezeny na danou síť
 - (nejčastěji) jsou určeny pro příjemce v jiných sítích
 - proto: ICMP zprávy je nutné směrovat
 - přenášet přes směrovače vhodnou cestou až k jejich cíli
 - otázka:
 - jak to udělat?
- možnost:
 - vkládat ICMP zprávy do linkových rámců
 - odpovídá zařazení ICMP do síťové vrstvy
 - ale **vyžadovalo by to podporu směrování ICMP zpráv ve směrovačích !!!**
 - ty by musely být multiprotokolové
 - kromě IP směrovat i ICMP
 - alternativa:
 - vkládat ICMP zprávy do IP datagramů
 - stejně jako např. TCP či UDP datagramy
 - ale: **je to spor s tím, že ICMP patří do síťové vrstvy !!**
-
- ```
graph LR; ICMP[ICMP zpráva] --> LinkFrame1[linkový rámec]; ICMP --> IPDatagram[IP datagram]; IPDatagram --> LinkFrame2[linkový rámec];
```

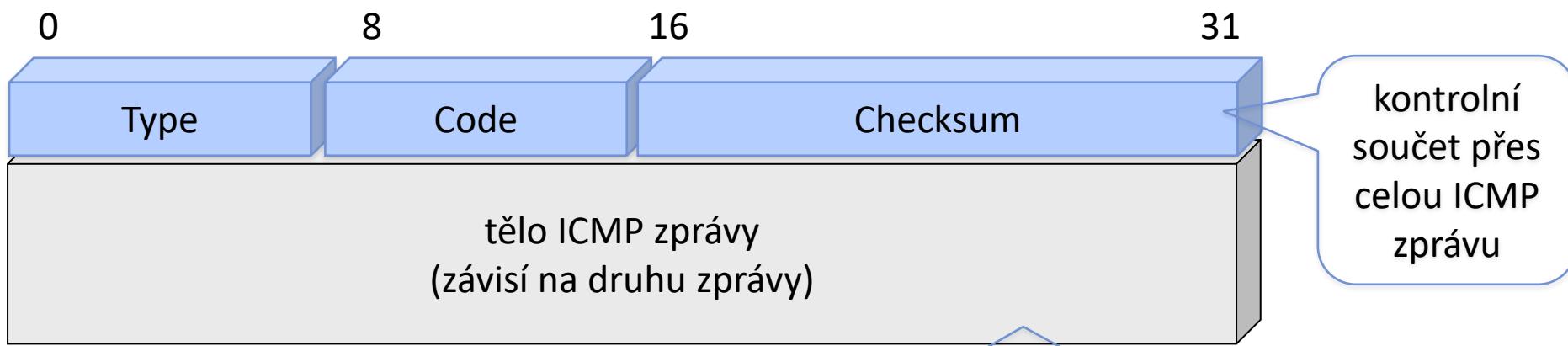
# jak se přenáší ICMP zprávy?

- zvolené řešení:
  - pro potřeby přenosu:
    - ICMP zprávy se vkládají do IP datagramů
      - a díky tomu je lze „dopravovat“ kamkoli
      - do libovolné sítě
    - pro potřeby zpracování (a implementace)
      - ICMP se považuje za součást síťové vrstvy
  - ale:
    - je to porušení konceptu vrstvových modelů
      - ICMP protokol by tak měl (správně) patřit na transportní vrstvu
        - a pak by nebyl implementován ve směrovačích
    - důvody jsou hlavně praktické
      - aby ICMP mohl fungovat a směrovače nemusely být multiprotokolové
  - výjimka: ICMP zprávy nejsou generovány
    - když je nesprávný kontrolní součet hlavičky IP datagramu, který „něco způsobil“
      - protože chyba může být právě v adrese odesilatele IP datagramu
    - když musí být zahozen IP datagram obsahující ICMP zprávu



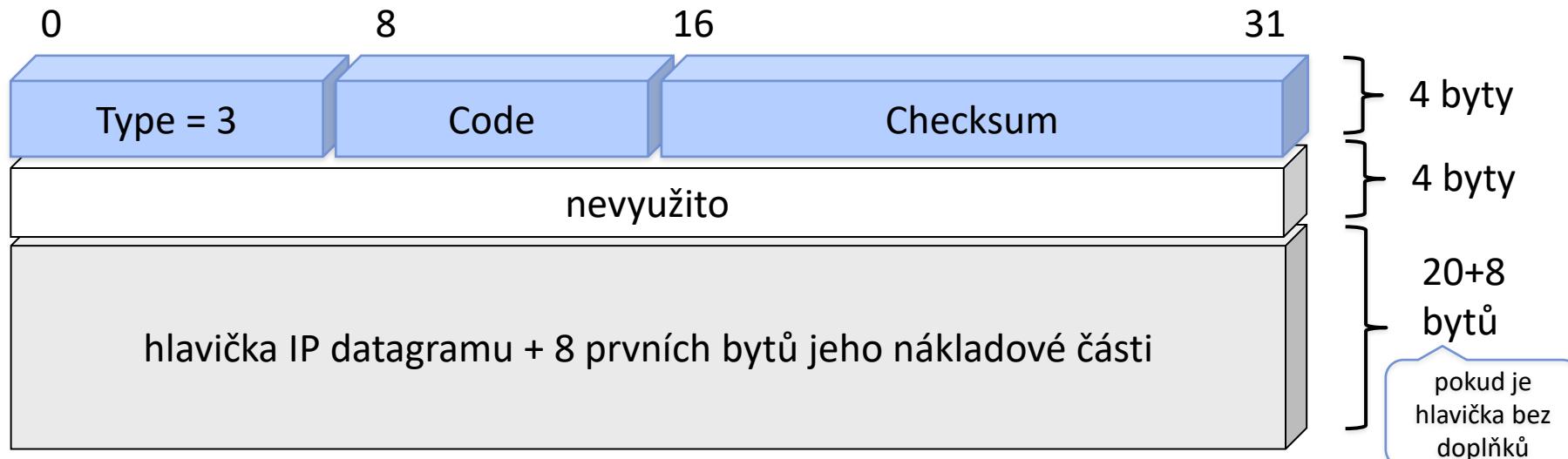
# ICMP zprávy

- Ize je dělit na:
  - chybové zprávy
    - informují o chybách, nejčastěji při zpracování IP datagramů
  - dotazy, výzvy, odpovědi a informační zprávy
    - informují o význačných skutečnostech, vznáší dotazy/podněty a reagují na ně
- rozlišují se podle:
  - ICMP Message Type, 8 bitů
    - (hlavní) typ ICMP zprávy
  - ICMP Message Code, 8 bitů
    - podtyp, upřesňuje druh zprávy



u chybových zpráv obsahuje hlavičku datagramu, kterého se zpráva týká, a prvních 8 byteů jeho těla (datové části)

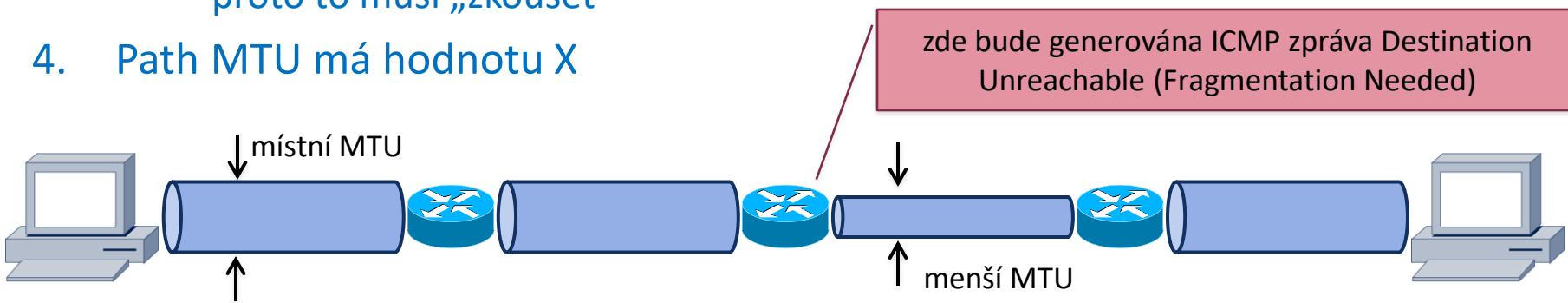
- je příkladem chybové ICMP zprávy
  - informuje o tom, že uzel (protokol IP) nemohl pokračovat v požadovaném zpracování IP datagramu a musel ho zahodit
    - typ ICMP zprávy je „Destination Unreachable“ (Type = 3)
  - důvodů, proč musel být IP datagram zahozen, může být celá řada
    - a jsou podrobněji rozvedeny v podtypu ICMP zprávy (položce Code), například:
      - **Code=0: Network Unreachable**
        - nelze pokračovat v přenosu do sítě určené v síťové části cílové adresy
      - **Code=1: Host Unreachable**
        - datagram byl doručen do cílové sítě, ale nelze ho předat cílovému uzlu v této síti



- další možné důvody (pro generování zprávy Destination Unreachable)
  - **Code=2: Protocol Unreachable**
    - cílový uzel neakceptuje protokol, určený v hlavičce IP datagramu
  - **Code=3: Port Unreachable**
    - nedostupný port, specifikovaný v hlavičce UDP datagramu nebo TCP segmentu
  - **Code=4: Fragmentation Needed and DF Set**
    - ICMP zpráva, generovaná v situaci, kdy je třeba provést fragmentaci, ale tato je zakázána nastavením bitu „Don't Fragment“ v hlavičce IP datagramu
    - .....
- obecné vlastnosti ICMP zpráv Destination Unreachable
  - protokol IP funguje stylem „best effort“, a stejně fungují i zprávy ICMP Destination Unreachable
  - tj. nejsou garantované !!!!
    - nelze se spoléhat, že budou doručeny původnímu odesilateli vždy, když dojde k zahodení nějakého IP datagramu
    - protože ICMP zprávy jsou přenášeny v IP datagramech
      - jejich doručení není garantováno
      - při zahodení IP datagramu s ICMP zprávou se již další ICMP zpráva negeneruje

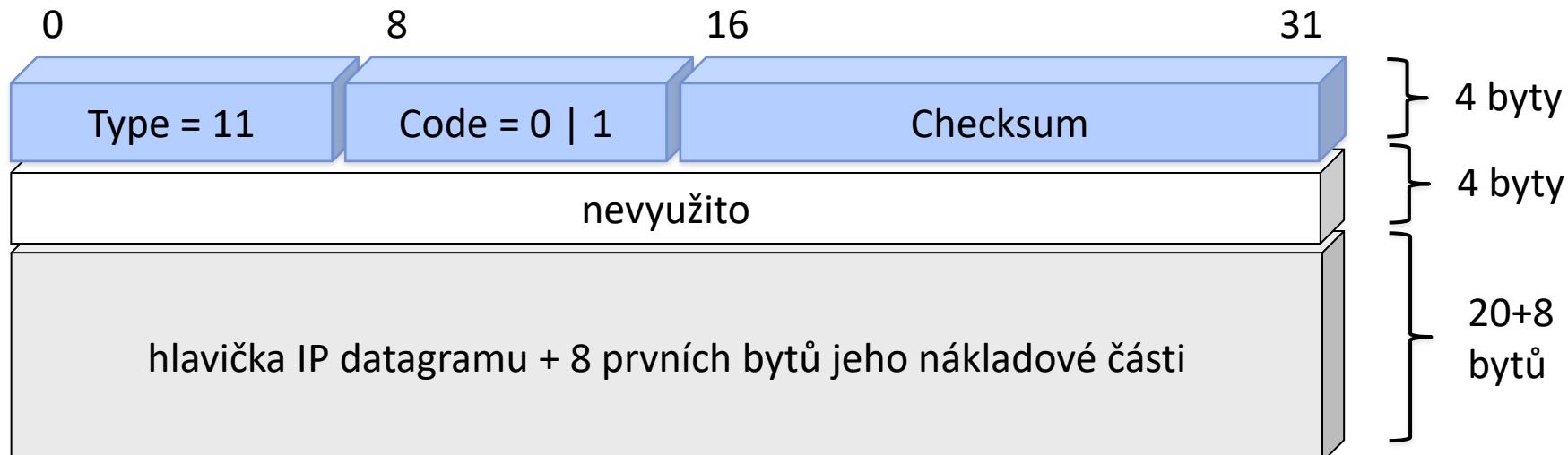
# MTU Path Discovery

- jde o postup, prostřednictvím kterého lze nalézt nejmenší MTU
  - na cestě (Path) mezi dvěma uzly (A a B)
  - připomenutí:
    - „výchozí“ uzel (A) zná pouze své „místní“ MTU
      - toto MTU je současně maximem MTU po celé cestě k B (Path MTU)
- postup uzlu A:
  1. X nastaví na velikost „místního“ MTU
  2. uzel A připraví IP datagram velikosti X, nastaví mu příznak Don't Fragment a odešle jej uzlu B
  3. pokud A dostane zpět ICMP zprávu Destination Unreachable (Type 3), s podtypem (Code=4, tj. Fragmentation Needed), sníží X a jde zpět na bod 2
    - uzel A se nedozví, kvůli jaké hodně MTU mělo dojít k fragmentaci
      - proto to musí „zkoušet“
  4. Path MTU má hodnotu X



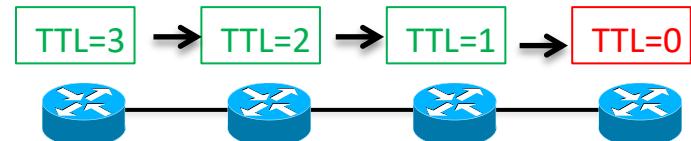
- je generována ve dvou situacích:

- když dojde k vynulování položky TTL v hlavičce IP datagramu
  - je to interpretováno jako zacyklení při směrování
  - do ICMP zprávy je vložen začátek „zacykleného rámce“
    - jeho hlavička + prvních 8 byteů
  - ICMP Message Code je nastaven na 0
- když při sestavování fragmentů vyprší časový limit (a fragmenty nejsou všechny)
  - význam: nelze sestavit (celý) původní datagram
  - ICMP Message Code je nastaven na 1



# traceroute

- vyvolání ICMP zprávy Time Exceeded může být i záměrné
  - využívá se toho (například) pro zjišťování cesty v síti (od uzlu A k uzlu B)
    - utilitou traceroute (ve Windows jen tracert)
- postup:
  - pošle se „blok dat“ na adresu uzlu B, TTL se nejprve nastaví na 1
    - nejbližší směrovač sníží TTL na 0, IP datagram (s UDP datagramem) zahodí a vrátí zpět zprávu ICMP Time Exceeded
      - díky čemuž se uzel A dozví adresu prvního směrovače „po cestě“
  - posílají se další bloky dat, TTL se postupně zvyšuje o 1, vrací se Time Exceeded ...
  - tím se postupně „odhalí“ celá cesta z A do B
    - standardně se dělají 3 pokusy, při kterých se také měří čas odezvy



uzel negeneruje  
ICMP zprávy Time  
Exceeded

Výpis trasy k f.root-servers.net [192.5.5.241] s nejvýše 30 směrováními:

|   |        |        |        |                                           |
|---|--------|--------|--------|-------------------------------------------|
| 1 | < 1 ms | < 1 ms | < 1 ms | 192.168.1.1                               |
| 2 | *      | *      | *      | Vypršel časový limit žádosti.             |
| 3 | 10 ms  | 8 ms   | 18 ms  | ip-86-49-52-65.net.upcbroadband.cz        |
| 4 | 18 ms  | 16 ms  | 19 ms  | 84.116.222.213                            |
| 5 | 20 ms  | 20 ms  | 31 ms  | 84-116-130-229.aorta.net [84.116.130.229] |
| 6 | 35 ms  | 18 ms  | 16 ms  | 84.116.135.1                              |
| 7 | 19 ms  | 18 ms  | 18 ms  | 84.116.132.146                            |
| 8 | 21 ms  | 34 ms  | 18 ms  | de-cix.r1.fra1.isc.org [80.81.194.57]     |
| 9 | 19 ms  | 21 ms  | 17 ms  | f.root-servers.net [192.5.5.241]          |

Trasování bylo dokončeno.

# varianty traceroute

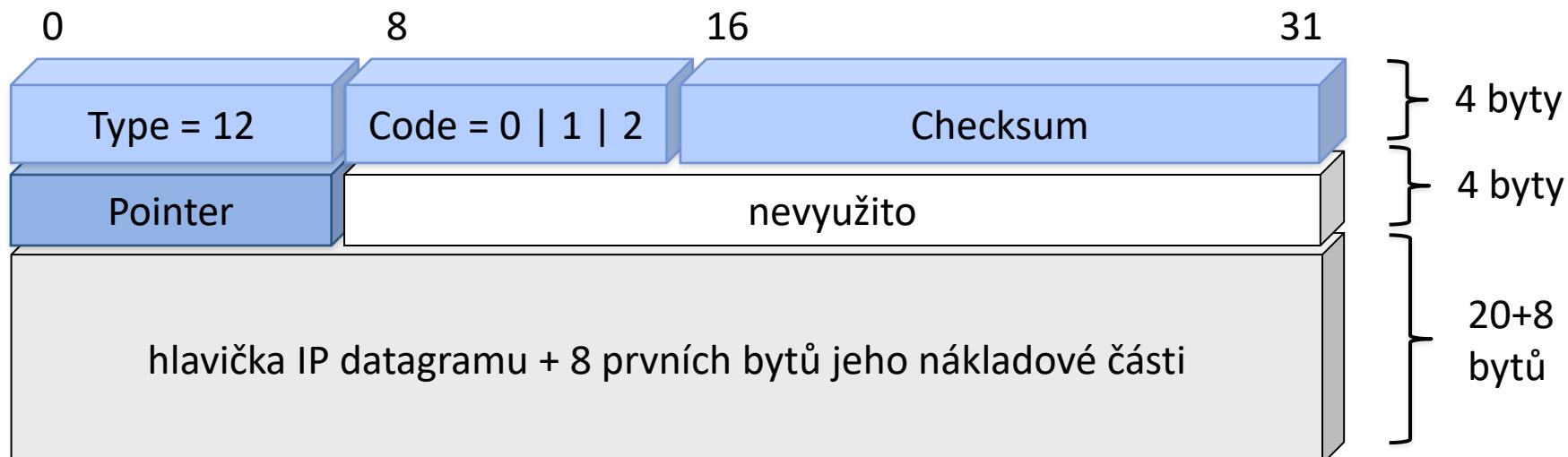
- původní varianta traceroute:
  - „blokem dat“ je ICMP zpráva Echo (Request)
  - uzel A ji vkládal do IP datagramů s TTL nastaveným (nejprve) na 0 .....
  - ale někde to nefungovalo (zprávy Time Exceeded se negenerovaly)
  - protože standardy říkají, že **při zahazování IP datagramu s ICMP zprávou se další ICMP zprávy negenerují**
    - později to bylo změněno na: „*negenerují se při zahazování datagramu s chybovou ICMP zprávou*
- dodnes takto funguje traceroute v MS Windows
  - skrze utilitu tracert
- modifikace: Van Jacobsonův traceroute
  - „blokem dat“ je UDP datagram
  - uzel A je posílá na neexistující čísla portů
    - na dostatečně vysoké číslo portu (od 33434 výše), které obvykle není používáno
  - přitom také postupně zvyšuje TTL
  - uzly „po cestě“ generují ICMP zprávu Time Exceeded
  - koncový uzel (uzel B) generuje ICMP zprávu Port Unreachable
  - nikoli zprávu Time Exceeded
- takto funguje traceroute na unixových platformách

důsledek: firewally mohou reagovat různě

- důvodem pro zahovení IP datagramu může být i zahlcení (congestion)
- řešení pomocí ICMP:
  - příjemce, který „nestihá“, může generovat ICMP zprávu **Source Quench**
    - Type = 4, Code = 1, jinak standardní formát ICMP zprávy
    - Quench znamená „hašení“
  - a poslat ji tomu, o kom si myslí, že způsobuje jeho zahlcení
- problém:
  - je to „jednostranný výkřik“ ve smyslu: zpomal
    - příjemci to neříká, jak moc má zpomalit
    - reakce na zprávu Source Quench není definovaná
      - záleží na příjemci této zprávy, jak se zachová
  - neexistuje opačná zpráva
    - zpráva, která by informovala o konci zahlcení
- dnes:
  - používání ICMP zprávy Source Quench se nedoporučuje
    - řízení toku i předcházení zahlcení se řeší jinými mechanismy



- ICMP zpráva **Redirect** (Type=5)
  - signalizuje nesprávné (neoptimální) směrování
    - podrobněji viz problematika směrování – příjemce by se měl „poučit“
- ICMP zpráva **Parameter Problem** (Type=12)
  - obecná zpráva pro jakýkoli jiný problém
  - neříká o jaký problém jde, ale pouze kde je (v hlavičce datagramu)
    - Code=0: v položce Pointer je offset toho bytu hlavičky, který problém způsobuje
    - Code=1: v hlavičce chybí požadovaný doplněk (Option)
    - Code=2: špatná délka datagramu



# informační a další ICMP zprávy

- vedle chybových ICMP zpráv
  - které informují o chybách a problémech se zpracováním IP datagramů
- existují také informační ICMP zprávy (a dotazy, výzvy a odpovědi)
  - které zprostředkovávají řádné fungování protokolu IP
- například:
  - ICMP Echo (Request) a Echo Reply
    - testování dostupnosti: výzva a odpověď na ni
  - ICMP Router Advertisement a Router Solicitation
    - „inzerát“ o existenci směrovače, výzva: „je zde nějaký směrovač“
      - podrobněji viz téma 7
  - ICMP Timestamp (Request) a Timestamp Reply Messages
    - jeden uzel může požádat jiný uzel o sdělení jeho systémového času

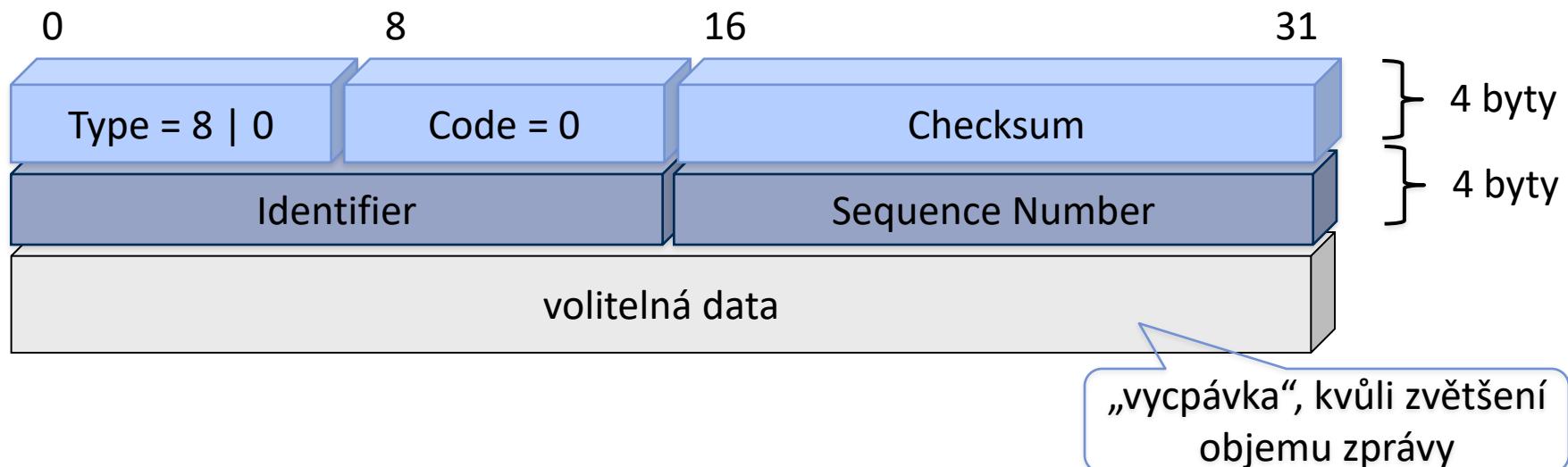
stále se používají

- 
- ICMP Address Mask Request a Address Mask Reply
    - možnost vyžádat si síťovou masku od směrovače
  - ICMP Traceroute Message
    - efektivnější než pomocí TTL a Time Exceeded – ale

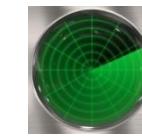
už se nemají používat  
(deprecated)

# ICMP zprávy Echo a Echo Reply

- slouží k testování dostupnosti síťových uzlů
  - ICMP zpráva **Echo** (někdy označovaná jako **Echo Request**) je výzvou protistraně
    - Type=8
  - ICMP zpráva **Echo Reply** je reakcí (odpovědí) protistrany na výzvu ICMP Echo
    - Type=0
- další položky ICMP zpráv Echo a Echo Reply
  - **Identifier**: podle něj se párují zprávy Echo (Request) a Echo Reply
    - aby se vědělo, k jaké výzvě se odpověď vztahuje
  - **Sequence Number**: pořadové číslo výzev (Echo Request) a odpovědí (Echo Reply)



- slouží k testování dostupnosti a reakce síťových uzlů
  - jméno odvozeno od fungování sonaru ("ping")
    - a má i „backronym“: Packet InterNet Groper
- způsob fungování:
  - uzel, kde je utilita ping spuštěna, posílá série zpráv Echo cílovému uzlu
  - cílový uzel odpovídá zprávami Echo Reply
    - tyto odpovědi generuje již TCP/IP stack
- vyhodnocuje se:
  - jak ztrátovost odpovědí, tak i jejich zpoždění (RTT, Round Trip Time)
    - a udává se jako minimální, střední a maximální hodnota
- podpora Echo a Echo Reply:
  - dle RFC 1122 povinná
  - dnes:
    - je to považováno za možné bezpečnostní riziko



```
Příkaz PING na ksi.ms.mff.cuni.cz [195.113.20.128] - 32 bajtů dat:
Odpověď od 195.113.20.128: bajty=32 čas=25ms TTL=54
Odpověď od 195.113.20.128: bajty=32 čas=9ms TTL=54
Odpověď od 195.113.20.128: bajty=32 čas=12ms TTL=54
Odpověď od 195.113.20.128: bajty=32 čas=12ms TTL=54
```

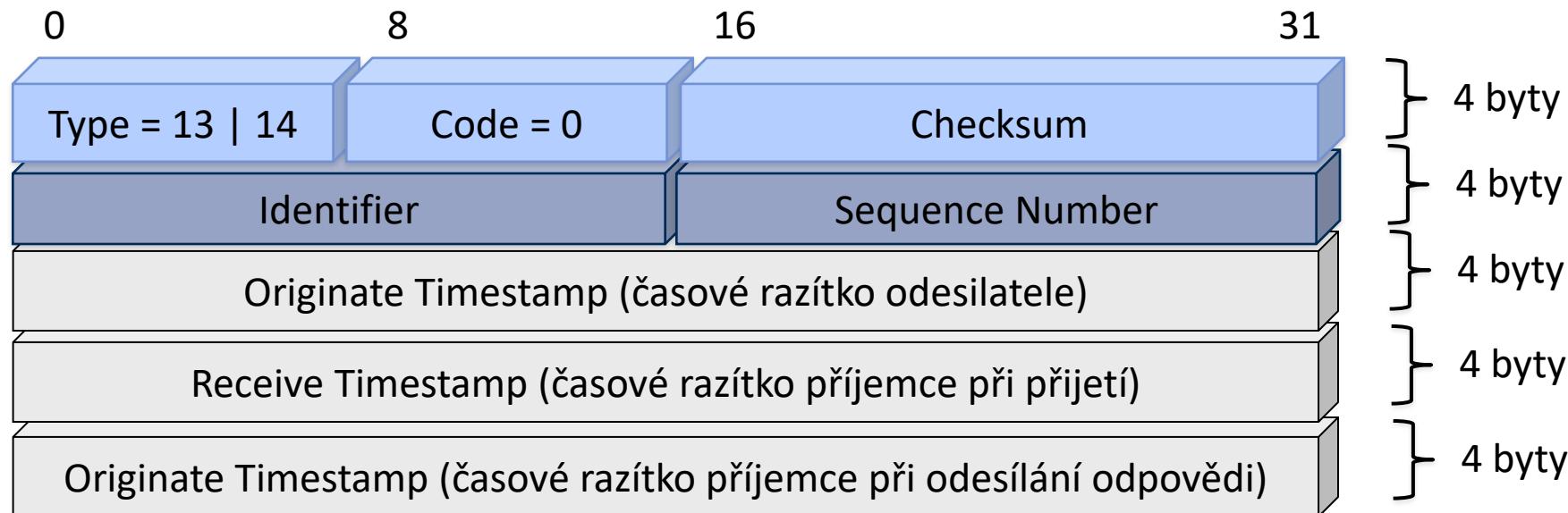
Statistika ping pro 195.113.20.128:

Pakety: Odeslané = 4, Přijaté = 4, Ztracené = 0 (ztráta 0%),  
Přibližná doba do přijetí odezvy v milisekundách:

Minimum = 9ms, Maximum = 25ms, Průměr = 14ms

# ICMP Timestamp Request a Reply

- umožňuje vzájemnou (časovou) synchronizaci uzlů
  - jeden uzel (A) si může vyžádat údaj o systémovém čase jiného uzlu (B)
    - pomocí ICMP zprávy **Timestamp** (někdy: **Timestamp Request**), Type=13
      - do odesílané žádosti přidá svůj údaj o čase (Originate Timestamp)
  - oslovený uzel (B) :
    - přijme žádost a vloží do ní svůj údaj o čase přijetí žádosti (Receive Timestamp)
    - odpoví pomocí ICMP zprávy **Timestamp Reply**, Type = 14
      - přitom do ní vloží svůj údaj o okamžiku odesílání odpovědi
  - uzel A: ze tří časových údajů dokáže zjistit, jak dlouho trval přenos



- ARP: Address Resolution Protocol
  - slouží potřebám převodu IP adres na HW (linkové) adresy
- princip fungování
  - uzel A zná IP adresu uzlu B, a potřebuje znát jeho HW adresu
  - sestaví ARP zprávu, ve které uvede IP adresu uzlu B
    - a také svou IP a HW adresu
  - tuto ARP zprávu vloží do linkového rámce a pomocí (linkového) broadcastu rozešle jako dotaz po celé síti, ve které se nachází
  - uzel B rozpozná svou IP adresu a odpoví
    - sestaví ARP zprávu s odpovědí, ve které uvede svou HW adresu
    - tuto zprávu pošle již přímo (unicast-em) uzlu A

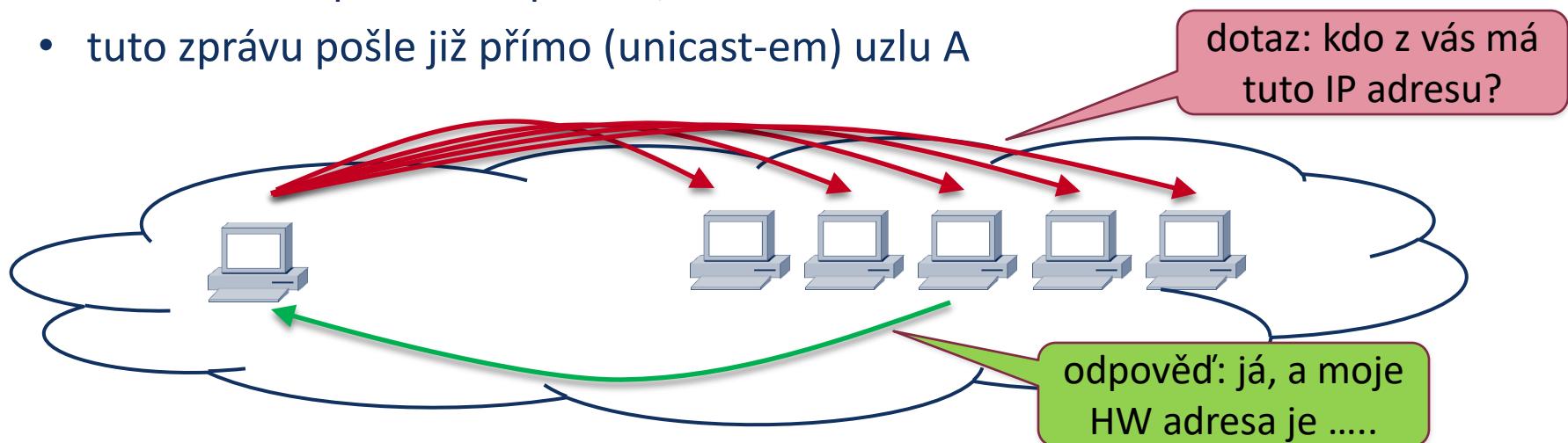
IP adresa

↓ ARP

HW adresa

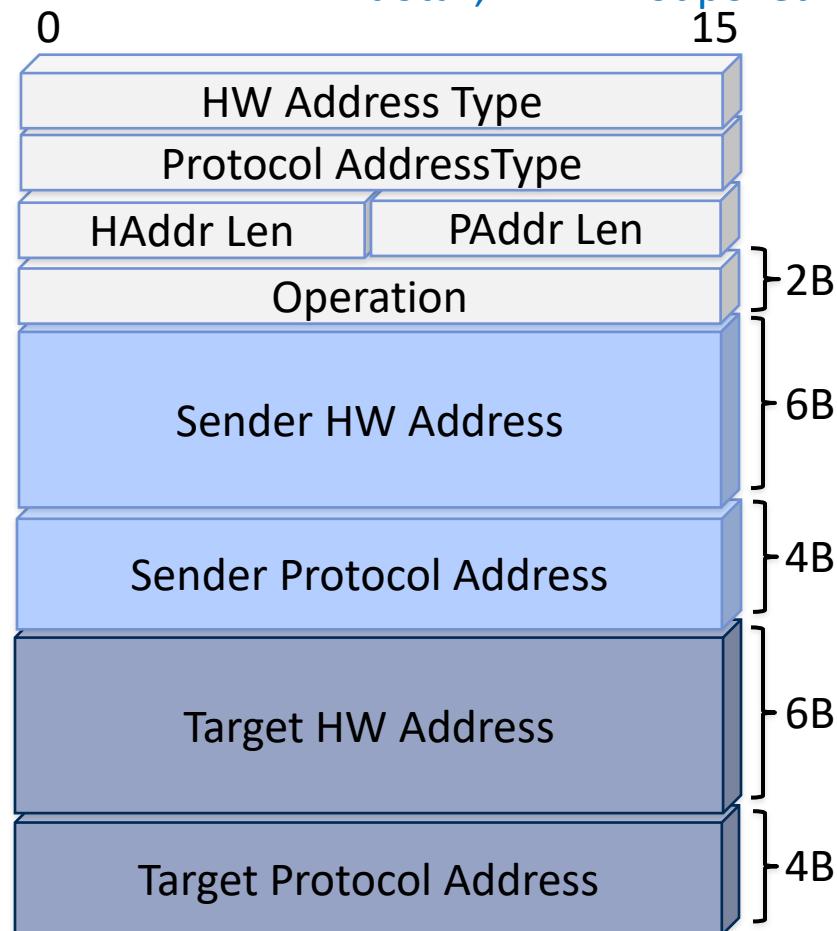
(např. Ethernetová  
adresa)

broadcast musí být k dispozici,  
jinak ARP nelze použít



# protokol ARP

- do které vrstvy patří protokol ARP?
  - funguje jen v dané síti, nepřekračuje její hranice
  - kvůli tomu by měl patřit do vrstvy linkové, resp. vrstvy síťového rozhraní
  - ARP zprávy se vkládají do linkových rámců a přenáší v těchto rámcích
    - stejně jako IP datagramy
      - které mají Ethertyp 0x0800
      - ARP má Ethertyp 0x0806
    - to řadí protokol ARP na síťovou vrstvu
- formát ARP zprávy
  - je stejný pro dotaz i odpověď
  - rozlišuje se jen položkou **Operation**
    - 1= ARP dotaz, 2= ARP odpověď



vyplní tazatel při dotazu:

- Sender HW Address = jeho HW adresa
- Sender Protocol Address = jeho IP adresa
- **Target Protocol Address = IP adresa, na kterou se ptá**

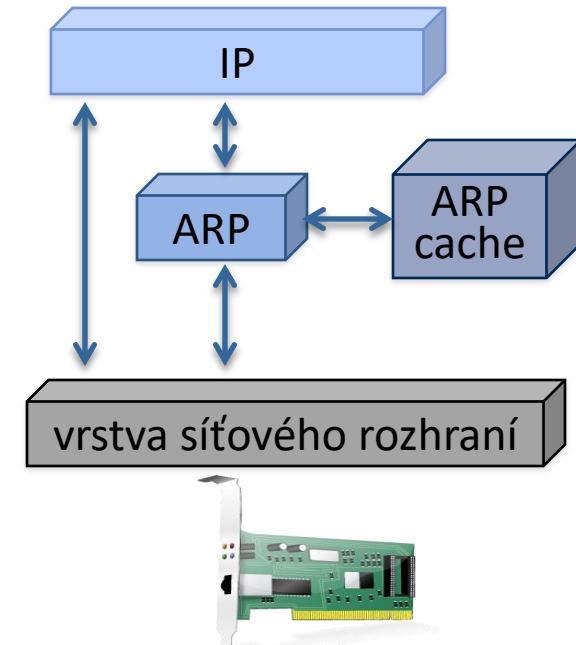
vyplní uzel, který odpovídá na dotaz:

- **Target HW Address = jeho HW adresa**

# ARP cache

- protokol ARP má nezanedbatelnou režii
  - hlavně kvůli broadcastu
- je snaha optimalizovat jeho fungování
- a co nejvíce používat **ARP cache**
  - vyrovnávací paměť, ve které si ARP pamatuje výsledky převodů IP->HW
    - tzv. resolutions
    - představa: jde o tabulku, kde jsou informace o vazbách mezi IP a HW adresami (tzv. bindings)
- fungování ARP cache:
  - položky v ARP cache mohou být **statické**
    - pokud je to žádoucí, např. kvůli bezpečnosti
  - jinak jsou **dynamické**
    - musí být pravidelně zapomínány
      - aby mohly reflektovat změny v síti
    - a také osvěžovány
      - aby se omezovaly nové dotazy

| IP adresa     | HW adresa         | vazba     |
|---------------|-------------------|-----------|
| 192.168.1.1   | 00-14-6c-23-7f-32 | dynamická |
| 192.168.1.136 | a4-67-06-55-ac-02 | dynamická |
| 192.168.1.255 | ff-ff-ff-ff-ff-ff | statická  |



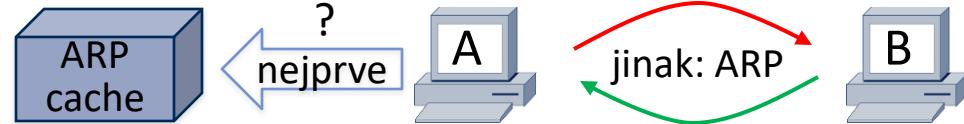
# zpracování ARP dotazu

- výchozí situace:

- uzel A zná IP adresu uzlu B, a potřebuje znát jeho HW adresu

- postup:

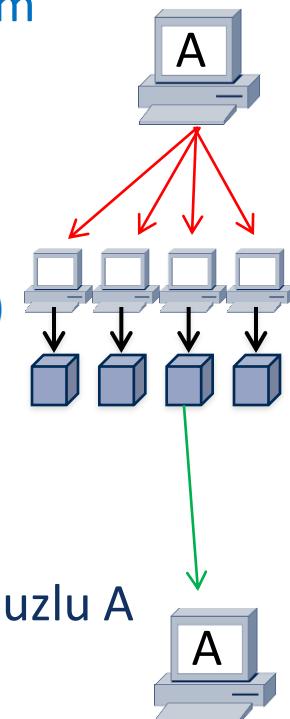
- uzel A se podívá do své ARP cache
  - pokud zde najde HW adresu k IP adrese uzlu B, končí
- uzel A sestaví a rozešle (linkovým broadcastem) ARP zprávu s dotazem
  - vyplní v ní svou IP a HW adresu, a IP adresu uzlu B



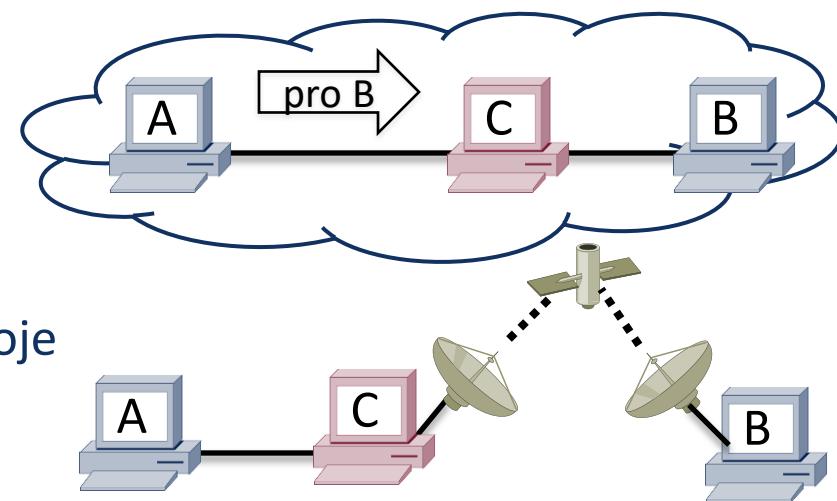
- každý uzel v síti zachytí ARP zprávu (vysílanou broadcastem), a:
  - vyjme ze zprávy vazbu (binding) mezi IP a HW adresu uzlu A
    - a pokud ji už má ve své ARP cache, tak ji osvěží (prodlouží její platnost)
  - zjistí, zda je užlem B (zda IP adresa uzlu B je jeho IP adresou)
    - pokud ne, ARP zprávu zahodí a končí

- uzel B:

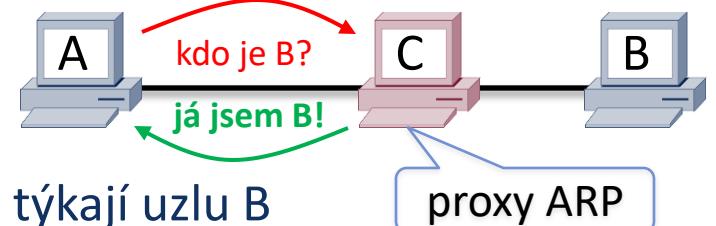
- si do své ARP cache zanese vazbu (binding) mezi IP a HW adresou uzlu A
  - nebo ji osvěží, pokud ji ve své ARP cache již měl
- sestaví ARP zprávu s odpovědí a odešle ji cíleně (unicast-em) uzlu A
  - v dotazu přehodí Sender/Target, příznak Dotaz/Odpověď a vyplní svou HW adresu



- existují situace, kdy:
  - uzly A a B se nachází ve stejné síti, A chce komunikovat s B přímo, ale
  - uzel B je „schován“ za jiným uzlem (uzlem C) a není pro A přímo dosažitelný
- například:
  - je-li uzel B za „mobilním agentem“
    - používá se při realizaci mobility uzlů v IP
  - je-li uzel B na spoji s extrémní latencí
    - například na druhé straně satelitního spoje
  - je-li uzel B za směrovačem
    - který spojuje dvě části téže sítě
      - resp. spojuje dva segmenty, ale chceme aby se jednalo o jednu síť

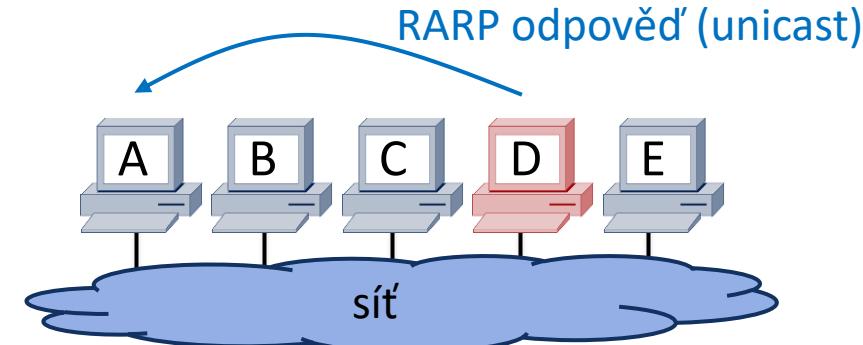
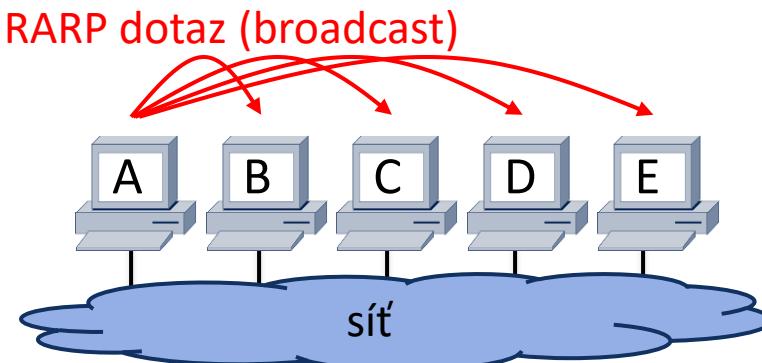


- fungování proxy ARP (ARP proxying):
  - uzel C se chová, jako kdyby byl uzlem B
    - ve smyslu: odpovídá na ARP dotazy, které se týkají uzlu B
    - generuje ARP odpovědi, v nich místo HW adresy uzlu B uvádí svou HW adresu
  - následně také „dostává“ veškerý provoz, určený uzlu B
    - a měl by ho uzlu B přeposílat (a od něj přebírat provoz v opačném směru)



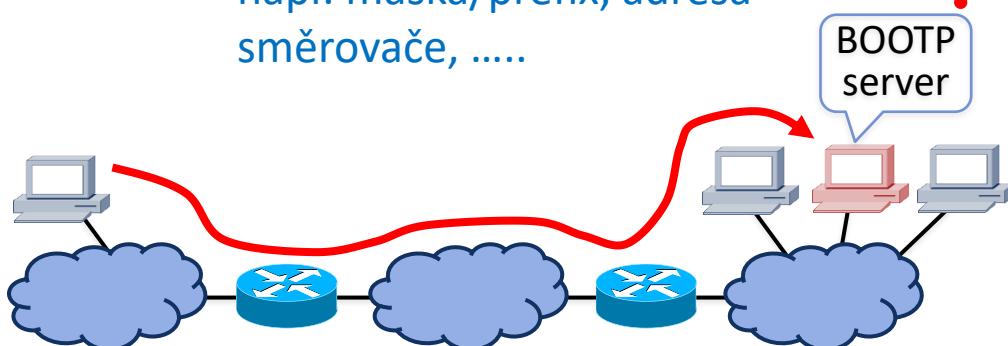
## reverzní ARP (RARP)

- fungování protokolu ARP lze obrátit (proto: Reverse ARP)
  - a dosáhnout tak převodu mezi HW a IP adresou
    - uzel zná svou HW adresu, a chce znát svou IP adresu
      - fakticky jde o (nejjednodušší variantu) přidělování IP adres jednotlivým uzelům
- postup:
  - uzel A, který nezná svou IP adresu, sestaví dotaz ve formě RARP zprávy
    - má stejný formát jako zpráva ARP, jen je „vyplněna opačně“ (obsahuje HW adr.)
      - a položka Operation má jiné hodnoty: 3 = RARP dotaz, 4 = RARP odpověď
  - uzel A vloží RARP zprávu do linkového rámce a ten rozešle pomocí broadcastu
    - příjemcem jsou všechny uzly v dané síti (dotaz se nedostane mimo danou síť)
  - ten uzel (D), který funguje jako RARP server, na dotaz odpoví (již pomocí unicastu)
    - pokud je takových uzelů více, může odpovědět kterýkoli z nich



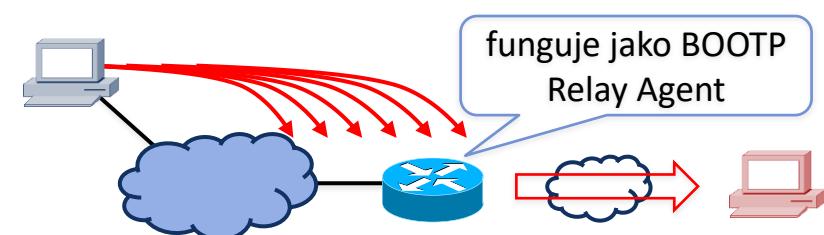
# protokol RARP vs. protokol BOOTP

- nevýhody protokolu RARP:
  - je starý a velmi jednoduchý
  - funguje na síťové vrstvě
  - dotazy se šíří pomocí linkového broadcastu
    - nefunguje v sítích bez broadcastu
  - RARP server musí být v každé síti
    - dotazy „neprojdou“ do jiných sítí
  - obsah RARP serveru musí být nastaven manuálně
  - **přiděluje se pouze IP adresa**
    - a nikoli již další potřebné parametry
      - např. maska/prefix, adresa směrovače, .....
- **BOOTP (Bootstrap Protocol)**
  - novější (RFC 951, 1985)
  - funguje na aplikační vrstvě
  - využívá transportní protokol UDP
  - dotazy se šíří pomocí IP broadcastu
  - BOOTP server se může nacházet v jiné síti
    - resp. jeden BOOTP server může „obsluhovat“ více sítí
  - vznik protokolu motivován potřebami bezdiskových stanic
    - které potřebují získat tzv. boot image
    - dokáže přidělovat IP adresy i další údaje
    - ale nikoli samotný boot image
      - na něj poskytne jen odkaz, stanice si jej pak musí stáhnout pomocí protokolu TFTP (Trivial FTP)



# fungování protokolu BOOTP

- pokud jsou klient a server ve stejné síti
  - klient odešle svůj dotaz na „broadcastovou“ IP adresu (255.255.255.255)
  - na dobře známý port 67 (na kterém server přijímá BOOTP dotazy)
  - server odpovídá (možnosti):
    - pomocí unicasta na IP adresu klienta
      - klient již mohl znát svou IP adresu, uvedl ji v dotazu a pomocí BOOTP se ptal na „další věci“ – například na svůj boot image
    - pomocí unicasta na HW adresu klienta
    - pomocí IP broadcastu
      - na dobře známý port 68 (určený pro příjem BOOTP odpovědí)
- pokud jsou klient a server v různých sítích
  - klient pošle svůj dotaz IP broadcastem
  - který se šíří jen v jeho síti
    - nikoli do dalších sítí
  - ve stejné síti, kde je klient, se musí nacházet **BOOTP Relay Agent**
    - obvykle zajišťuje směrovače
    - který „zachytí“ broadcast a předá dotaz BOOTP serveru do té sítě, ve které se nachází
    - pokud nezná umístění serveru, předá dotaz do další sítě také broadcastem



# protokol DHCP

- protokol BOOTP přiřazuje IP adresy trvale (staticky)
  - jakmile uzel dostane přidělenou IP adresu, může ji používat libovolně dlouho
- dnešní sítě potřebují přidělovat IP adresy dočasně (**dynamicky**)
  - na omezenou dobu – s možností následně přidělit stejnou IP adresu jinému uzlu
    - kvůli tomu, že koncové uzly se často „stěhují“ z jedné sítě do jiné sítě
- protokol **DHCP (Dynamic Host Configuration Protocol)**
  - je „evolucí“ protokolu BOOTP, navazuje na něj a využívá jeho principy fungování
- DHCP může přidělovat IP adresy:
  - „ručně“ (**manual allocation**)
    - správce sítě předepíše, jakou IP adresu má dostat konkrétní uzel
    - a BOOTP mu ji vlastně jen předá
  - trvale (**automatic allocation**)
    - IP adresu určí DHCP server sám, a přidělí ji trvale
      - na neomezenou dobu
    - používá se jen výjimečně
      - lepší už je „ručně“
  - dočasně (**dynamic allocation**)
    - IP adresu určí DHCP server sám, ale přidělí ji pouze na omezenou dobu

nejčastěji používaná varianta

# DHCP lease („pronájem“)

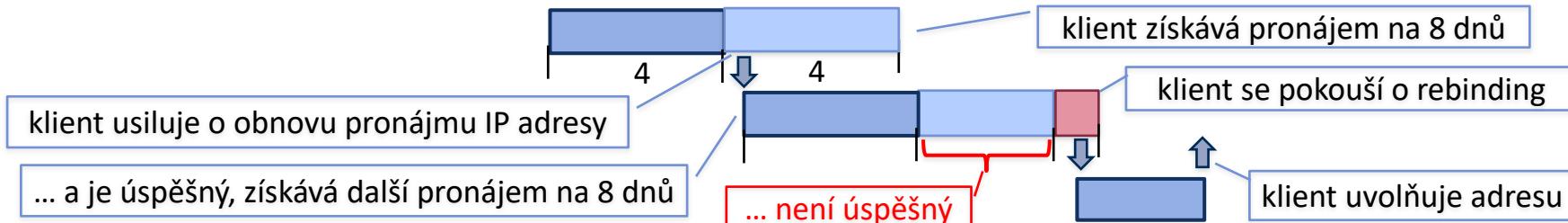
- původně:
  - uzel má svou IP adresu přidělenou (ve smyslu: je jejím držitelem, vlastníkem)
  - nemusí se starat o její obnovu/prodloužení či vrácení
  - nevýhodou je malá pružnost při hospodaření s IP adresami
- při dočasném přidělení (dynamic allocation):
  - uzel má svou IP adresu pouze dočasně pronajatou (leased)
  - a musí se aktivně starat (alespoň) o její obnovu
  - výhodou je větší pružnost při práci s IP adresami

- otázka:
  - na jak dlouho (dočasně) přidělovat IP adresy?
  - kratší doba = větší pružnost
    - ale menší „stabilita“ a větší režie
      - častější úkony spojené s přidělováním a obnovou
  - delší doba = menší pružnost
    - ale větší „stabilita“ a menší režie
- doba „pronájmu“ (**DHCP lease**)
  - může být například:
    - hodina
    - den
    - 3 dny (používá Microsoft)
    - týden
    - měsíc
    - rok (skoro už „trvalé“)

„pronájem“

## chování DHCP klienta

- DHCP klient prochází různými stavy a provádí různé úkony:
  - **alokace**: klient ještě nemá IP adresu a žádá DHCP server o „pronájem“ IP adresy
    - žádá o DHCP lease
  - **realokace**: klient již má „pronajatou“ svou IP adresu, ale snaží si tento pronájem potvrdit
    - například poté, co prošel rebootem či byl na nějakou dobu vypnut
      - ale ještě trvá doba předchozího „pronájmu“
  - **obnova**: před koncem „pronájmu“ se klient snaží o jeho prodloužení
    - kontaktuje DHCP server se žádostí o prodloužení „pronájmu“
      - začíná to zkoušet od poloviny stávajícího pronájmu (timer  $T_1 = 50\% \text{ lease time}$ )
  - **rebinding**: snaží se získat pronájem stejné IP adresy od jiného DHCP serveru
    - pokud se nepodaří obnova (např. když je původní DHCP server nedostupný), snaží se uzel o získání stejné IP adresy od jiného serveru (timer  $T_2 = 87,5\% \text{ l.t.}$ )
  - **uvolnění**: klient „vrací“ pronajatou IP adresu ještě před koncem jejího pronájmu



- DHCP server může mít k dispozici jeden nebo více **adresních rozsahů**
  - tzv. **pools (scopes, ranges)**, ze kterých přiděluje (pronajímá) IP adresy
    - část z nich může být vyhrazena/rezervována, pro „ruční“ (manual) přidělení
      - a dynamicky se přidělují pouze adresy ze zbytku rozsahu
- DHCP server může poskytovat svým klientům i další údaje, například:
  - masku sítě
  - místní časové pásmo
  - seznam směrovačů
    - které „vedou ven“ ze sítě klienta
      - poskytovány jsou jejich 32-bitové IP adresy,
      - pořadí udává preferenci použití
  - seznam DNS serverů
    - pořadí vyjadřuje preferenci / prioritu
    - DNS jméno uzlu (klienta) a doménu
    - jméno a velikost souboru s boot image
    - server přesného času (time server)
    - .....

The screenshot shows the 'Network DHCP Server Settings' configuration page for a subnet (Subnet 192.168.0.0 / 255.255.255.0). The interface includes sections for:

- Network Interface**: Shows the selected interface as 'Management LAN Interface'.
- DHCP Server**: Includes a checkbox for 'Enable DHCP Server'.
- Gateway**: A field for specifying the default gateway.
- Use interface address as gateway**: A checkbox for using the interface's own address as the gateway.
- Primary DNS**: A field for specifying the primary DNS server.
- Secondary DNS**: A field for specifying the secondary DNS server.
- Use this interface address as the DNS server**: A checkbox for using the interface's address as a DNS relay.
- Domain Name**: A field for specifying the domain name.
- Default Lease**: A field for specifying the default lease time.
- Maximum Lease**: A field for specifying the maximum lease time.

Below these settings is a section for **Dynamic Address Allocation Pools**, which currently lists 'No address pools currently allocated.' and includes an 'Add' button. The final section is **Reserved Addresses**, which also lists 'No addresses currently reserved.' and includes an 'Add' button.

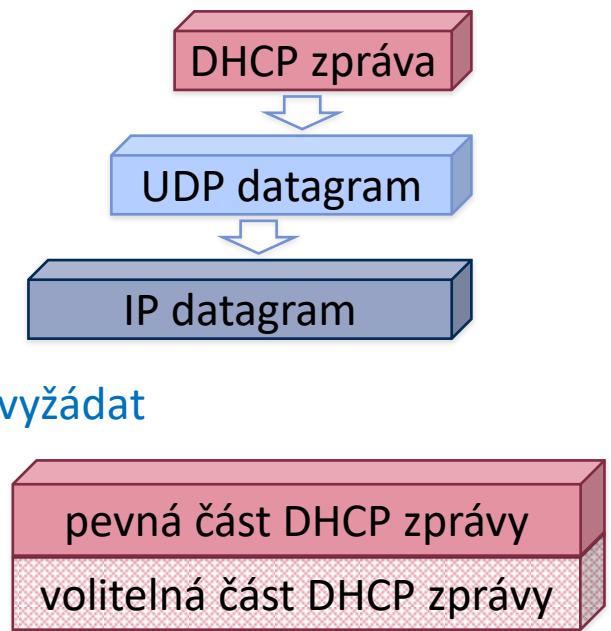
# zprávy protokolu DHCP

- formát zpráv DHCP je rozšířením BOOTP

- je stejný pro dotaz i odpověď
- používá i stejný způsob šíření jako BOOTP
  - dotaz pomocí broadcastu,
  - odpověď pomocí unicstu (ARP)
    - odpověď pomocí broadcastu si uzel musí explicitně vyžádat
  - vkládá se do UDP datagramů
    - využívá porty 67 a 68 (jako BOOTP)

- DHCP zpráva má:

- pevnou (a povinnou část), obsahuje mj.
  - rozlišení dotazu/odpovědi
  - ID transakce (pro spárování dotazu a odpovědi)
  - HW adresu klienta (jde-li o dotaz)
  - IP adresu, „pronajímanou“ klientovi (u odpovědi)
- volitelnou rozšiřující část (**DHCP Options**), obsahuje:
  - všechny ostatní údaje, poskytované DHCP serverem
    - masku, časové pásmo, adresy směrovačů, adresy DNS serverů, .....



} kvůli zpětné kompatibilitě s BOOTP (odpovědi DHCP serverů mohou přijímat i BOOTP klienti)

# příklad využití DHCP

- předpokládejme domácí síť, která:
  - využívá domácí bránu: plní roli směrovače i DHCP serveru
  - IP adresu pro své „vnější“ rozhraní dostává od ISP jako DHCP klient
  - používá rozsah IP adres **192.168.1/24** (tj. adresy 192.168.1.1 až 192.168.1.254)
    - tento rozsah je rozdělen na několik úseků, pro různé způsoby využití:
      - adresy **192.168.1.1 až 192.168.1.31** jsou vyhrazeny pro uzly, které nepotřebují DHCP
        - které mají svou IP adresu nastavenou pevně (staticky)
          - **192.168.1.1** je adresa rozhraní „místního směrovače“
      - adresy **192.168.1.32 až 192.168.1.254** jsou k dispozici pro DHCP (tzv. DHCP pool)
        - z toho některé (192.168.1.32 až 192.168.1.34) jsou rezervovány pro konkrétní uzly
          - DHCP server je **vždy přiděluje stejným uzlům**
        - ostatní jsou pomocí DHCP **přidělovány dynamicky**
          - ostatním uzlům, včetně „návštěvnických“, dle potřeby

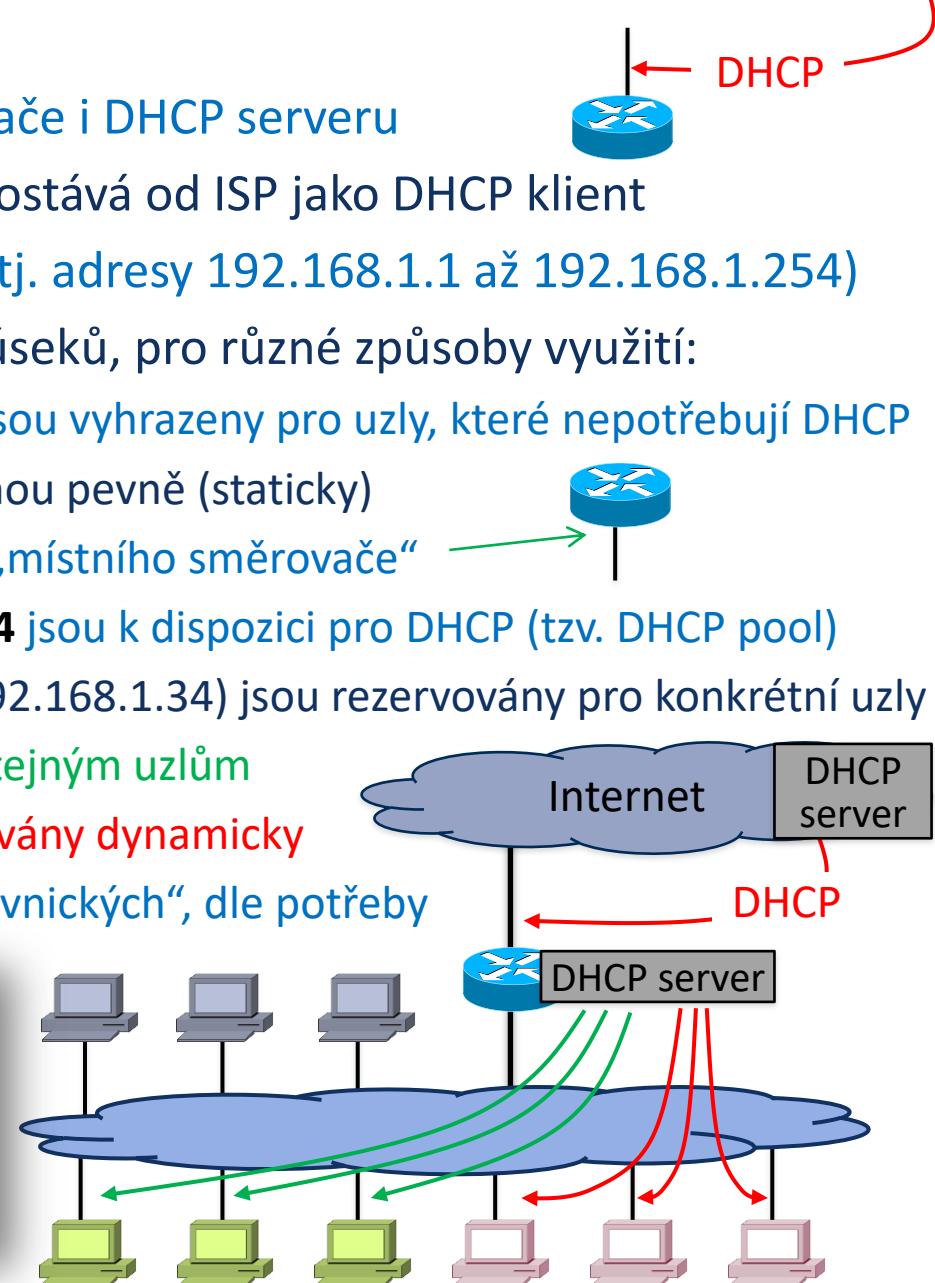
Use Router as DHCP Server

Starting IP Address: 192.168.1.1 · 192.168.1.31

Ending IP Address: 192.168.1.1 · 192.168.1.254

Address Reservation

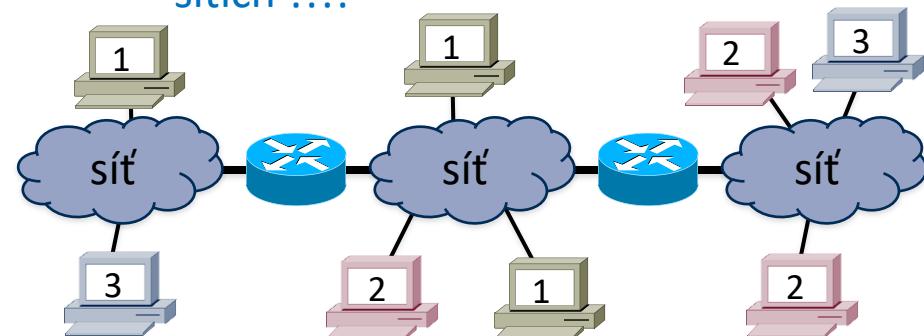
| # | IP Address   | Device Name      | MAC Address       |
|---|--------------|------------------|-------------------|
| 1 | 192.168.1.32 | HP LaserJet      | 00:14:38:DB:5F:EE |
| 2 | 192.168.1.33 | ZYXEL            | 00:23:F8:89:4A:FF |
| 3 | 192.168.1.34 | Grandstream VOIP | 00:0B:82:01:EB:44 |
| 4 |              |                  |                   |



# IPv4 multicast

- připomenutí:
  - multicast(ing) je rozesílání od jednoho zdroje k více příjemcům
- vyžaduje:
  - adresaci („skupinové“ adresy)
    - v IPv4 jde o adresy třídy D
      - adresují celé skupiny uzlů
  - správu multicastových skupin
    - vytváření a rušení skupin
    - přidávání a odebírání uzlů atd.,
    - pro IPv4 řeší protokol **IGMPv4**
  - přenos datagramů
    - jejich směrování a doručování všem členům multicastových skupin
      - mají na starosti směrovače
        - **je to pro ně komplikované !!**
    - odesilatelem může být i uzel, který není členem multicastové skupiny

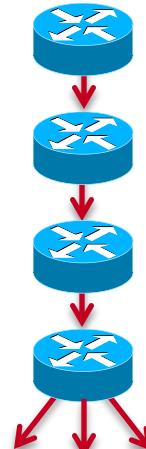
- otázka vrstvy:
  - multicast lze realizovat na linkové vrstvě
    - relativně jednoduché doručování
      - všechny uzly jsou ve stejné síti
  - multicast lze realizovat na síťové vrstvě
    - doručování mnohem složitější
      - protože uzly ve stejné multicastové skupině se mohou nacházet v různých sítích !!!!



IPv4 multicast funguje na síťové vrstvě

- jeho implementace **není povinná**
  - ale jen volitelná
  - v praxi: minimální

- doručování datagramů
  - vyžaduje speciální techniky směrování
- směrovače
  - musí mít informace o složení multicastových skupin
  - které se mohou průběžně měnit
  - musí mít informace o umístění uzlů z jednotlivých skupin
  - které se také může měnit
  - musí si budovat „distribuční stromy“
  - podle kterých distribuuje datagramy uzlům v jednotlivých skupinách
  - duplikuje datagramy
    - když je doručuje více uzlům
      - musí dávat pozor, aby tak nečinil zbytečně
      - aby duplikoval co nejméně
      - co nejpozději ....
- chování hostitelských počítačů
  - každý musí vědět, do kterých multicastových skupin je zařazen
  - tato informace musí korespondovat s informacemi, které mají k dispozici směrovače
- protokol IGMP
  - Internet Group Management Protocol
  - je protokolem pro komunikaci mezi směrovači a hostitelskými počítači
  - pro potřeby správy multicastových skupin
  - sdílení informací o skupinách a členech
  - zprávy protokolu IGMP se vkládají do IP datagramů
  - obdobně, jako zprávy ICMP



- připomenutí:
  - protokol IP „neobsazuje“ vrstvu síťového rozhraní (linkovou a fyzickou vrstvu)
    - předpokládá, že zde bude použita jiná, již existující technologie
      - například: Ethernet
- problém:
  - když jde o jednoduchý dvoubodový spoj, je i nasazení Ethernetu zbytečný luxus
    - (polo)duplexní dvoubodový spoj nevyžaduje adresování, řízení přístupu, ....
      - kolize vznikat nemohou (není potřeba přístupová metoda)
      - není třeba adresovat (příjemce je „ten, kdo je na druhé straně“)
  - jediné, co je třeba zajistit, je samotný framing
    - neboli: členění dat do rámců a jejich přenos
- řešení v rámci TCP/IP
  - „udělat to ještě jednodušeji a levněji než Ethernet“, pomocí vlastních protokolů
    - SLIP: Serial Line IP
    - PPP: Point-to-Point Protocol
  - jsou to linkové protokoly
    - přenáší linkové rámce, do kterých se vkládají IP datagramy

Point-to-Point



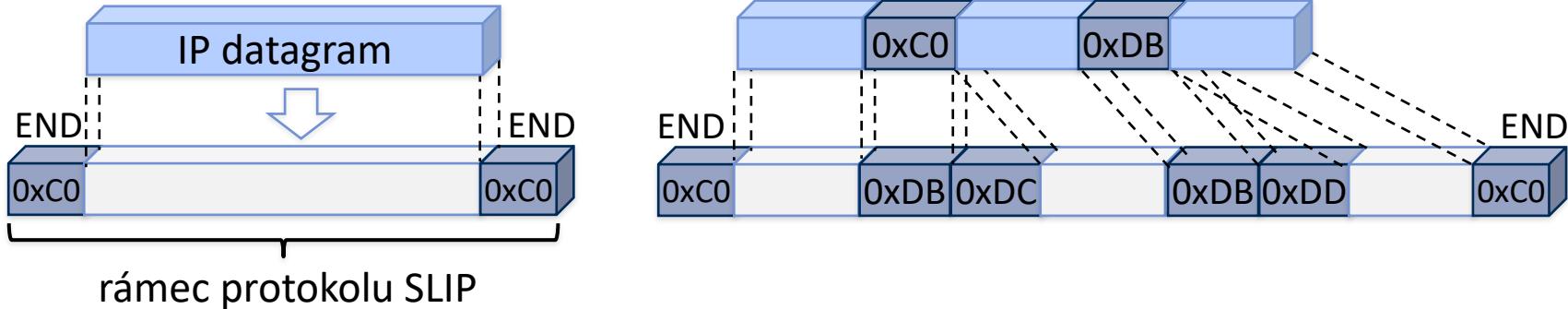
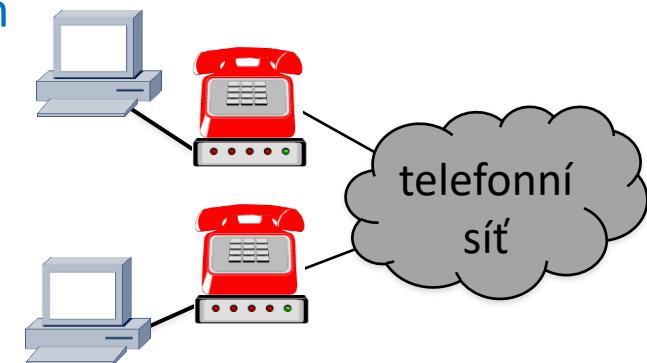
byť jde o výjimku z pravidla  
(že TCP/IP nepokrývá vrstvu  
síťového rozhraní)

IP datagram

rámeček SLIP nebo PPP

# SLIP: Serial Line IP

- velmi jednoduchý (znakově orientovaný) linkový protokol
  - inspirovaný potřebami přenosu po sériových linkách
    - například po telefonních linkách a modemech
  - předpokládá, že data jsou přenášena asynchronně
    - jako proud 8-bitových znaků
- protokol SLIP:
  - rozděluje proud 8-bitových znaků na jednotlivé rámce
  - začátek a konec rámce vymezuje znakem END (0xC0,  $11000000_B$ )
  - musí zajišťovat i transparenci dat:
    - pokud se v těle rámce vyskytne END, je nahrazen dvojicí ESC (0xDB) a 0xDC
    - pokud se v těle rámce vyskytne ESC, je nahrazen dvojicí ESC (0xDB) a 0xDD



# PPP: Point-to-Point Protocol

- do rámců protokolu SLIP lze vkládat pouze IP datagramy
  - není zde možnost vyjádřit, jakého typu je obsah rámce (proto: jen IP datagram)
- protokol PPP (Point-to-Point Protocol) je podstatně „bohatší“
  - je stále znakově orientovaným linkovým protokolem
    - tj. přenášená data chápe jako znaky, začátek a konec rámce označuje pomocí řídících znaků, transparenci dat zajišťuje pomocí vkládání znaků
    - podporuje řízení linkového spoje (používá protokol **LCP**, Line Control Protocol)
      - dokáže navazovat spojení na linkové vrstvě, ukončovat spojení, dohodnout s protistranou parametry
    - podporuje autentizaci (ověření identity)
      - podporuje více variant autentizace, např.:
        - **PAP** (Password Authentication Protocol)
        - **CHAP** (Challenge-Handshake Authentication Protocol)
      - podporuje vkládání síťových paketů různých druhů
        - dokáže rozlišit jejich druh (nemusí jít jen o IP datagramy)
        - vychází vstříč potřebám různých síťových technologií (např. jejich adresování)
        - pomocí „řídících protokolů“ (např. **IPCP**: Internet Protocol Control Protocol)

