

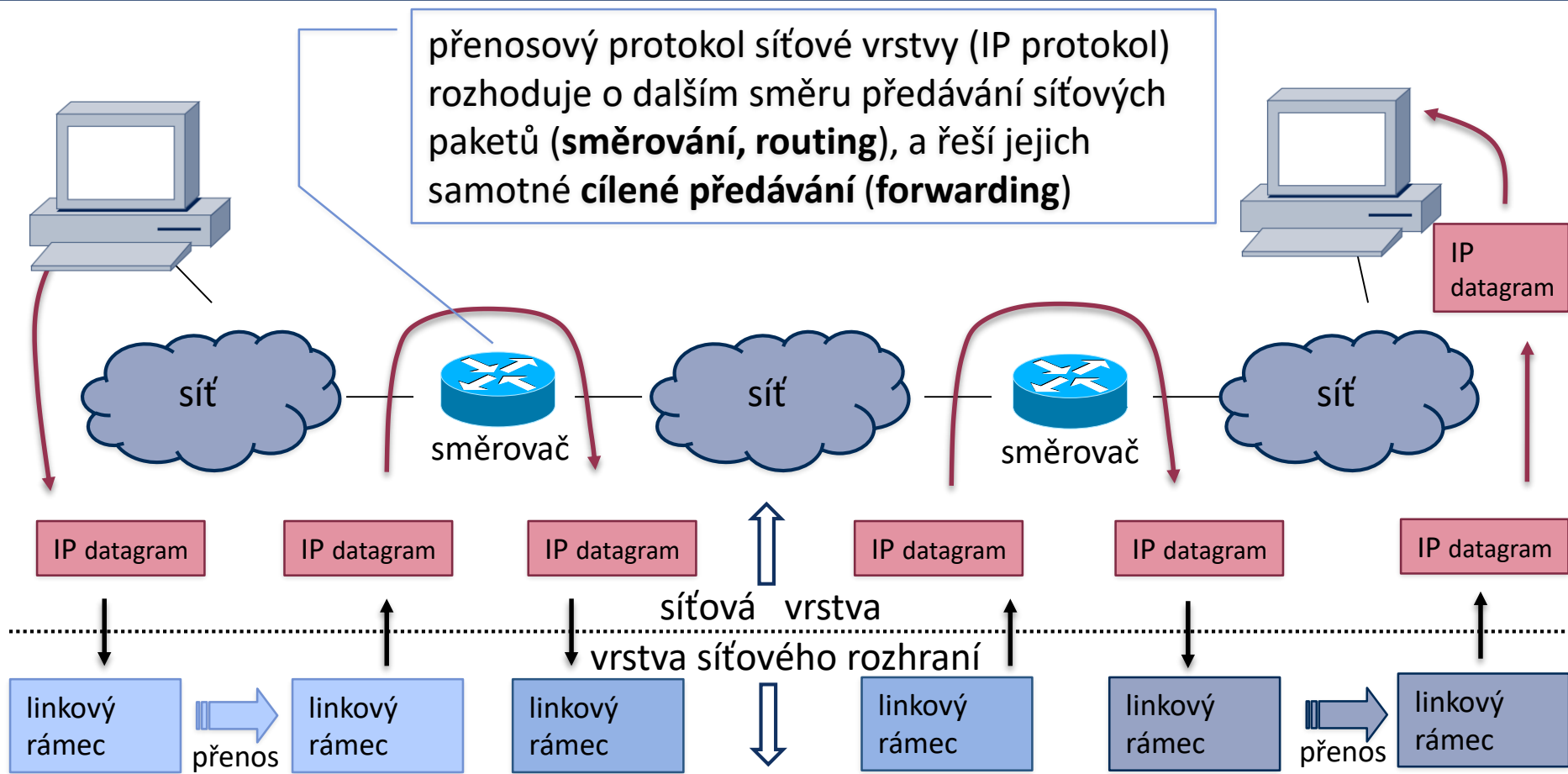
Rodina protokolů TCP/IP verze 3

Téma 5: Protokol IPv4

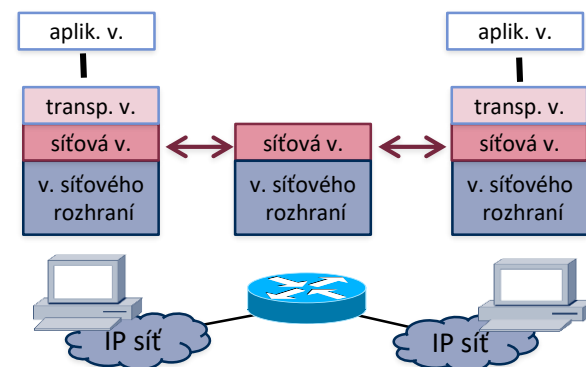
Jiří Peterka

- 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 rychlost
 - 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

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**

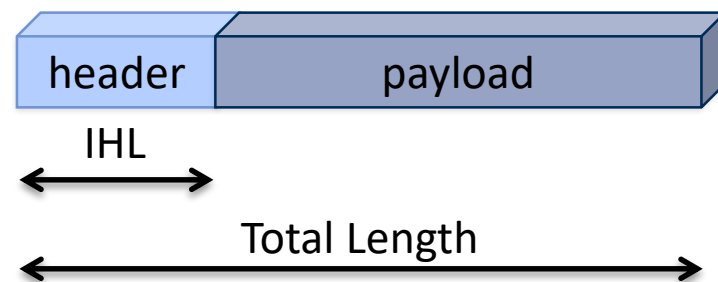
- s kontrolním součtem celého datagramu

- **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

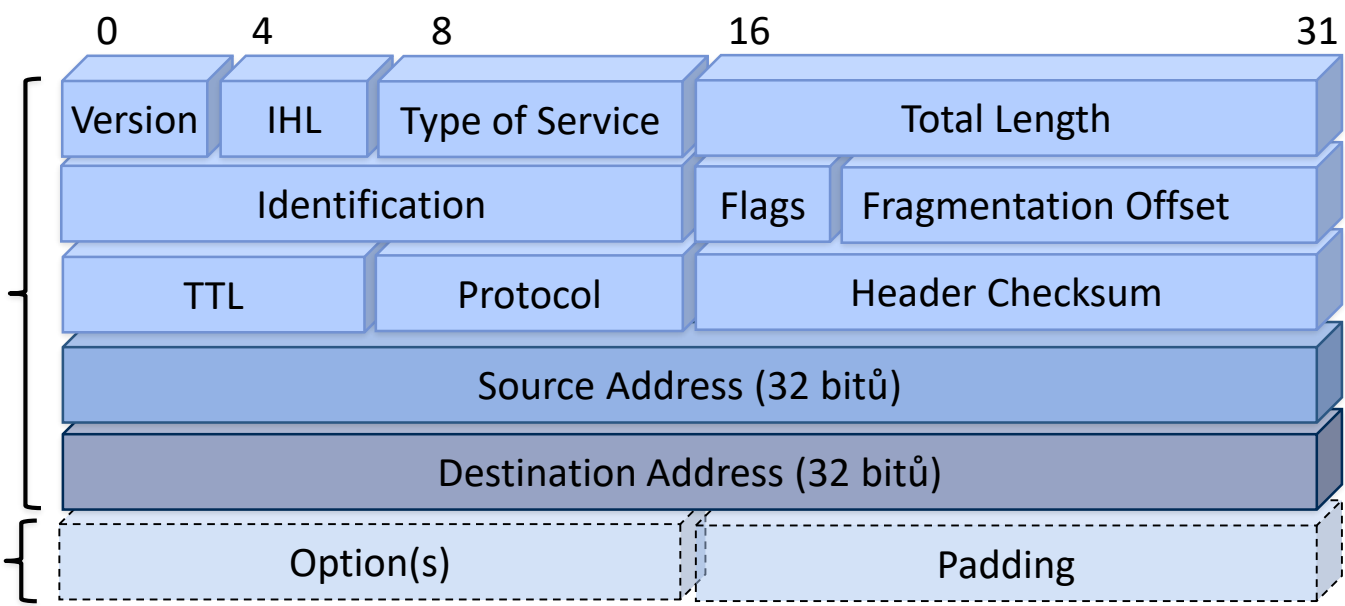


v praxi bývá podstatně
menší, kvůli velikosti
linkového rámce

hlavička IPv4 datagramu

- má celkem 14 položek, z nichž:

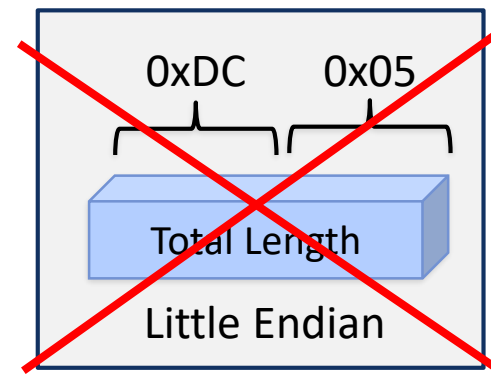
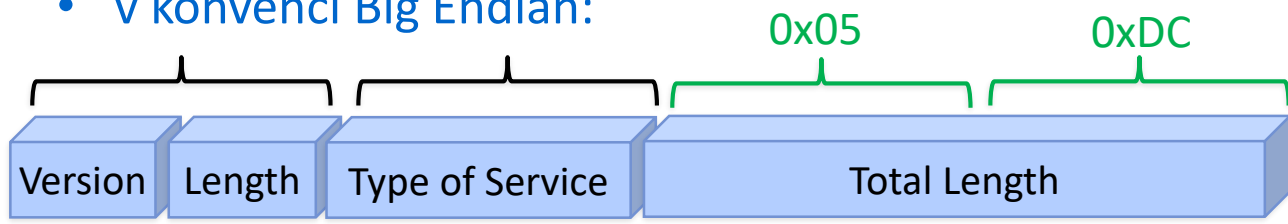
- 13 je povinných
 - dohromady mají velikost 20 bytů
 - minimální, současně obvyklá velikost hlavičky
- 1 je volitelná
 - doplňky (Options)



- údaje jsou do všech položek zapisovány podle konvence Big Endian
 - tj. „nejvýznamnější byte nejdříve“

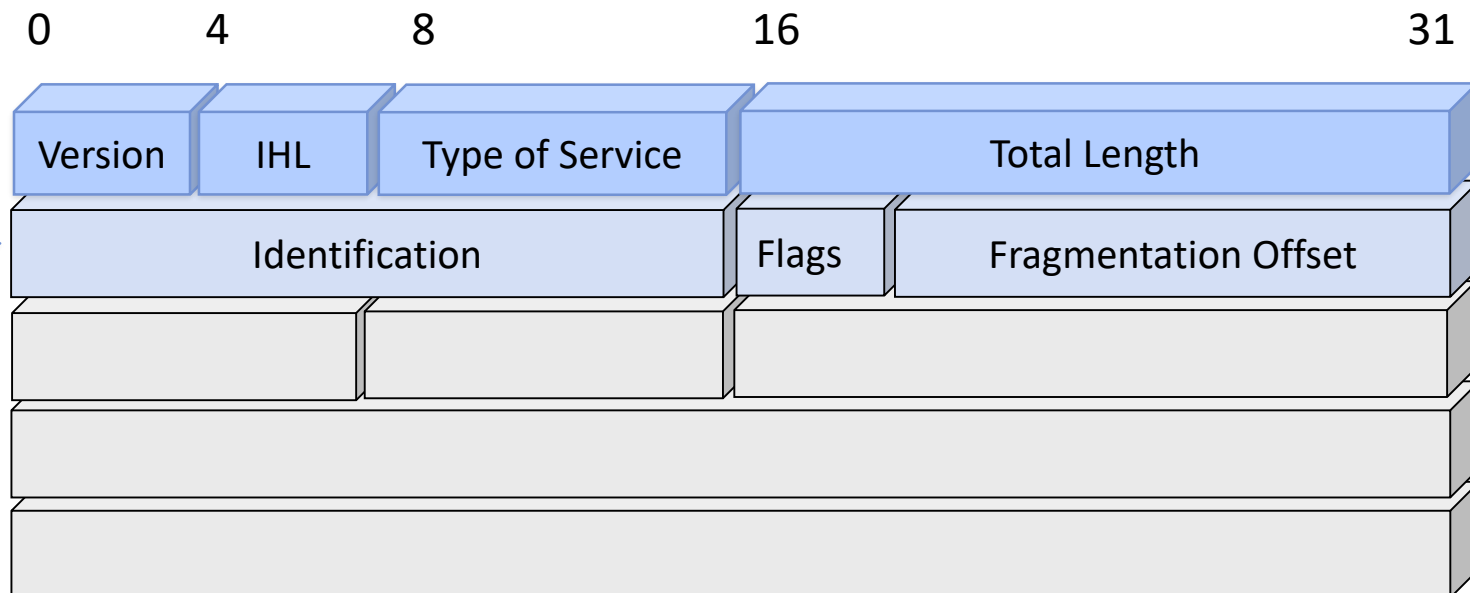
- příklad: IP datagram velikosti 1500 bytů (0x5DC)

- 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
 - dnes ignorováno
 - nebo se využívá pro DiffServ

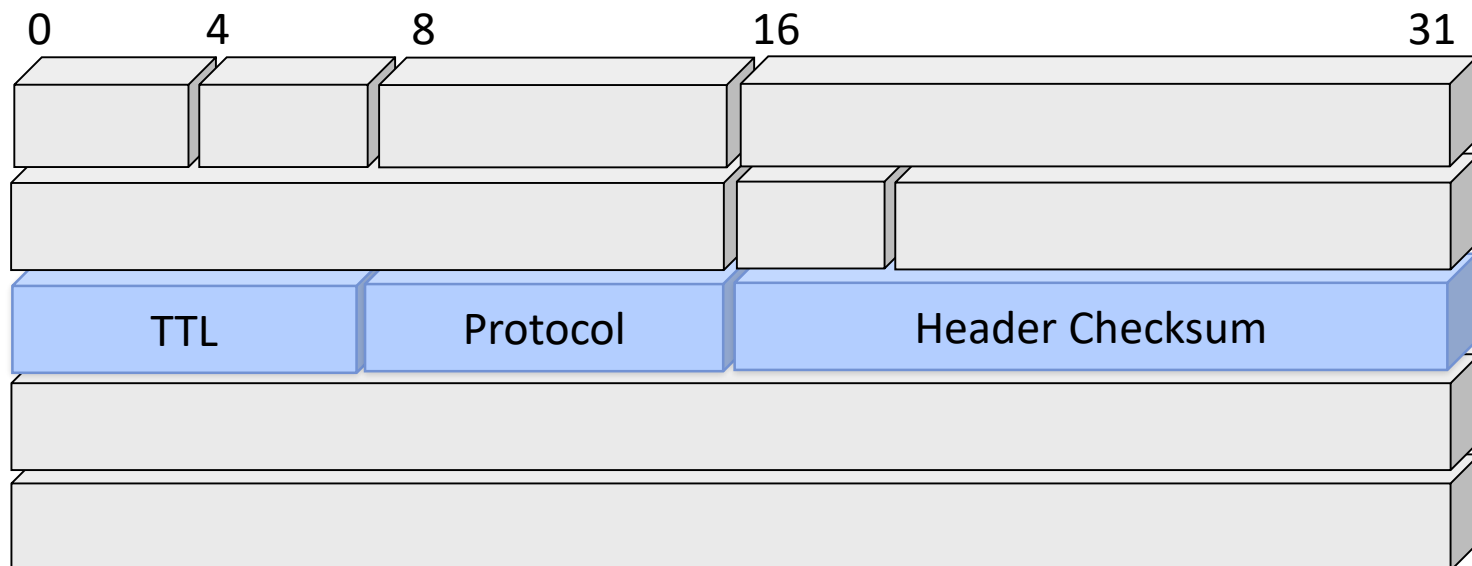


slouží
potřebám
fragmentace

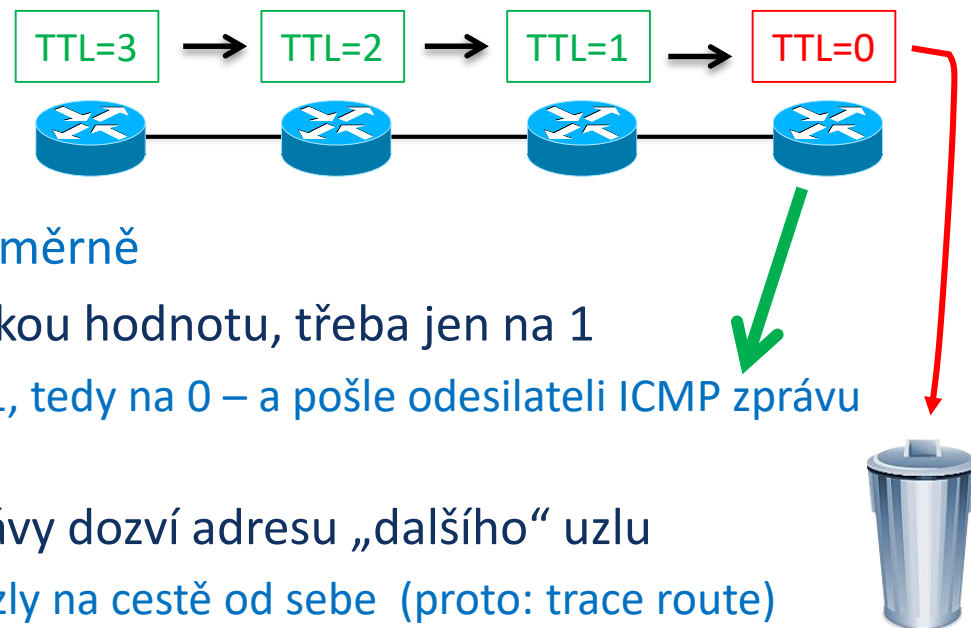
pokud k ní
nedochází, jsou
tyto položky
zbytečné

- **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)

pokud kontrolní součet nesedí, datagram může být zahozen



- položka **TTL** chrání před zacyklením, funguje jako (klesající) čítač
 - tzv. hop count (v IPv6 se jmenuje: **Hop Limit**)
 - odesílatel 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 odesílateli 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 odesílateli ICMP zprávu Time Exceeded
 - odesílatel datagramu se z této zprávy dozví adresu „dalšího“ uzlu
 - takto může „vystopovat“ všechny uzly na cestě od sebe (proto: trace route)



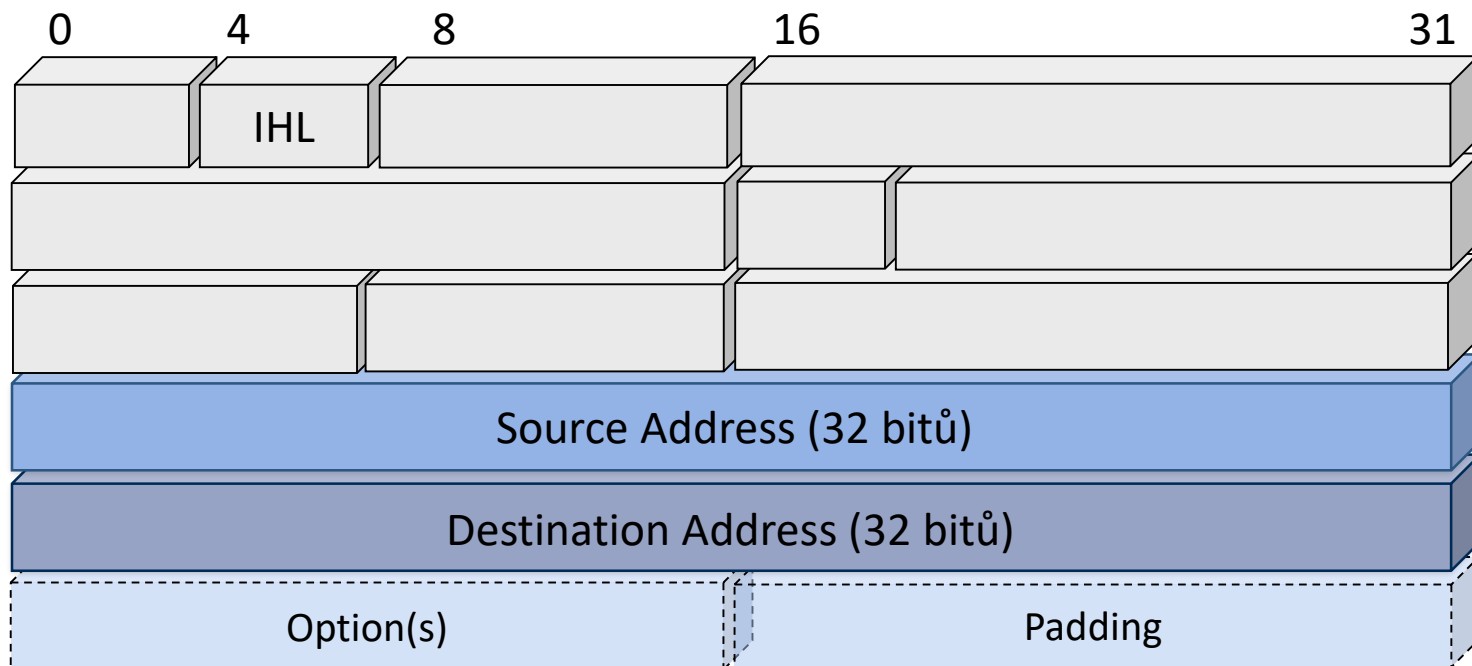
položka Header Checksum

- 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
- 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
- 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)

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



volitelné doplňky (options)

- 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“

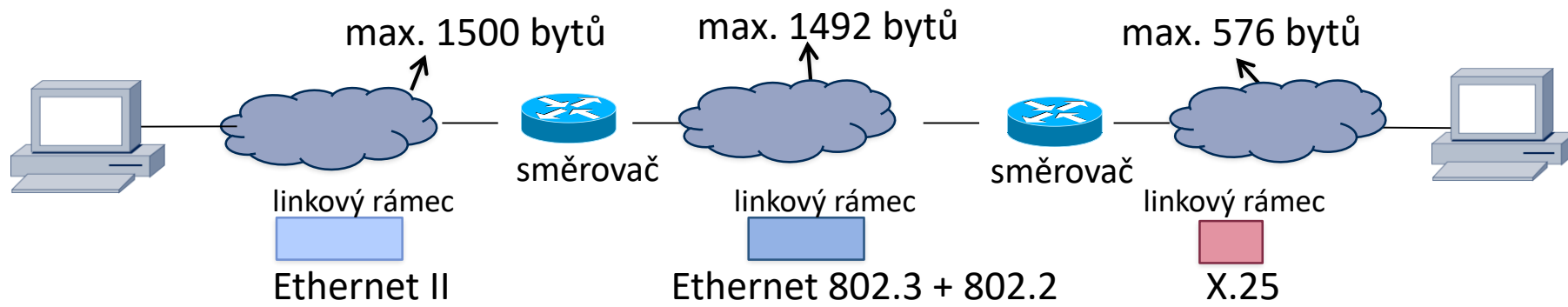
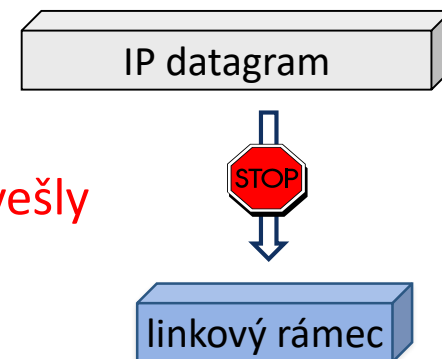
(pravděpodobný) smysl doplňků:

aby bylo možné měnit způsob, jakým protokol IP standardně nakládá s datagramy

problém fragmentace

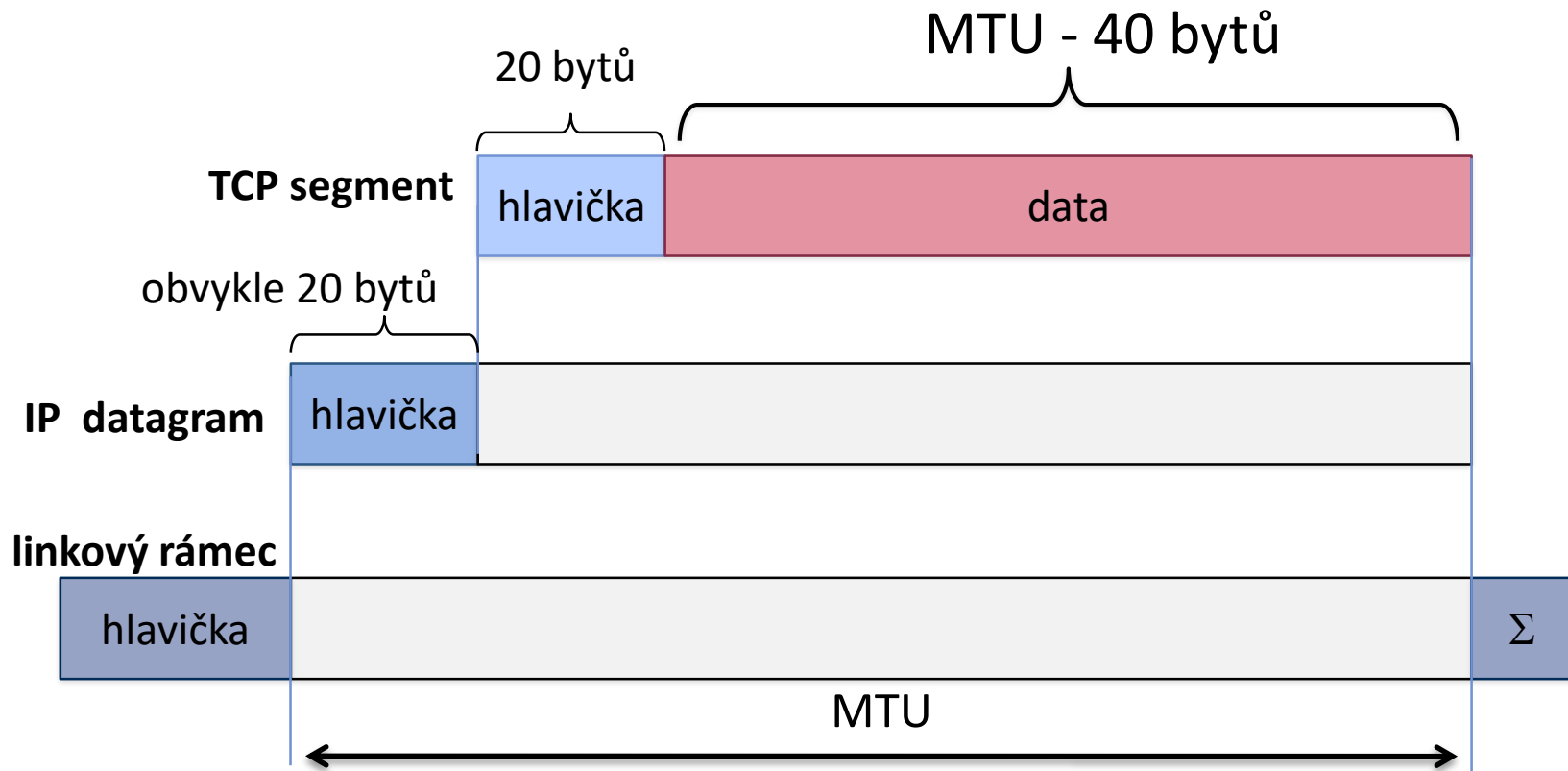
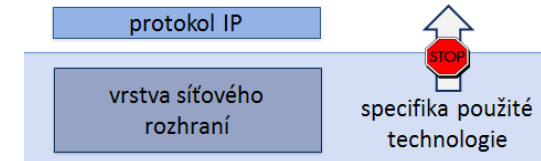
- prvotní příčina:
 - technologie vrstvy síťového rozhraní pracují s linkovými rámci omezené velikosti
 - do kterých se IP datagram nemusí vejít!!
- řešení:
 - dělat IP datagramy jen tak velké, aby se do linkových rámců vešly
- problém:
 - ne vždy je to možné (volit IP datagram dostatečně malý)
 - o 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 rámci !!!

max. 64 kB



MTU: Maximum Transmission Unit

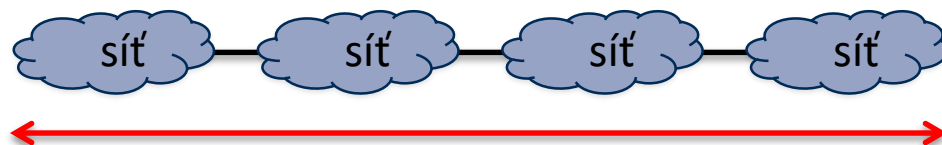
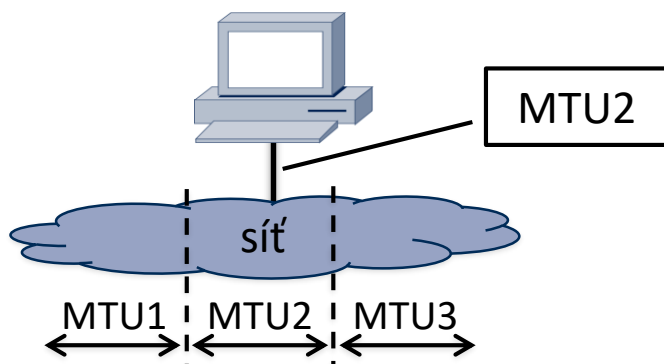
- protokol IP zakrývá všechna specifika linkových technologií
 - výjimkou je informace o velikosti (nákladové části) jejich rámce
 - skrze parametr **MTU** (Maximum Transmission Unit)
 - tuto informaci se dozvídá jak protokol TCP, tak i aplikace
 - podle ní mohou „porcovat“ svá data
 - ovšem ani to nemusí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



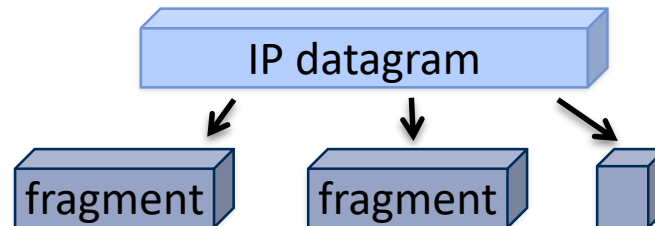
- **Path MTU** („MTU po celé cestě“)
 - fakticky: minimum přes všechna MTU od zdrojového uzlu až po cílový
 - dá se zjistit pomocí postupu zvaného **Path MTU Discovery**
 - ale: nemusí vždy fungovat správně
 - kvůli nespojovanému způsobu fungování protokolu IP
 - skutečná data mohou být přenášena jinou 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 (odesílatelů):

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

- podmínka:

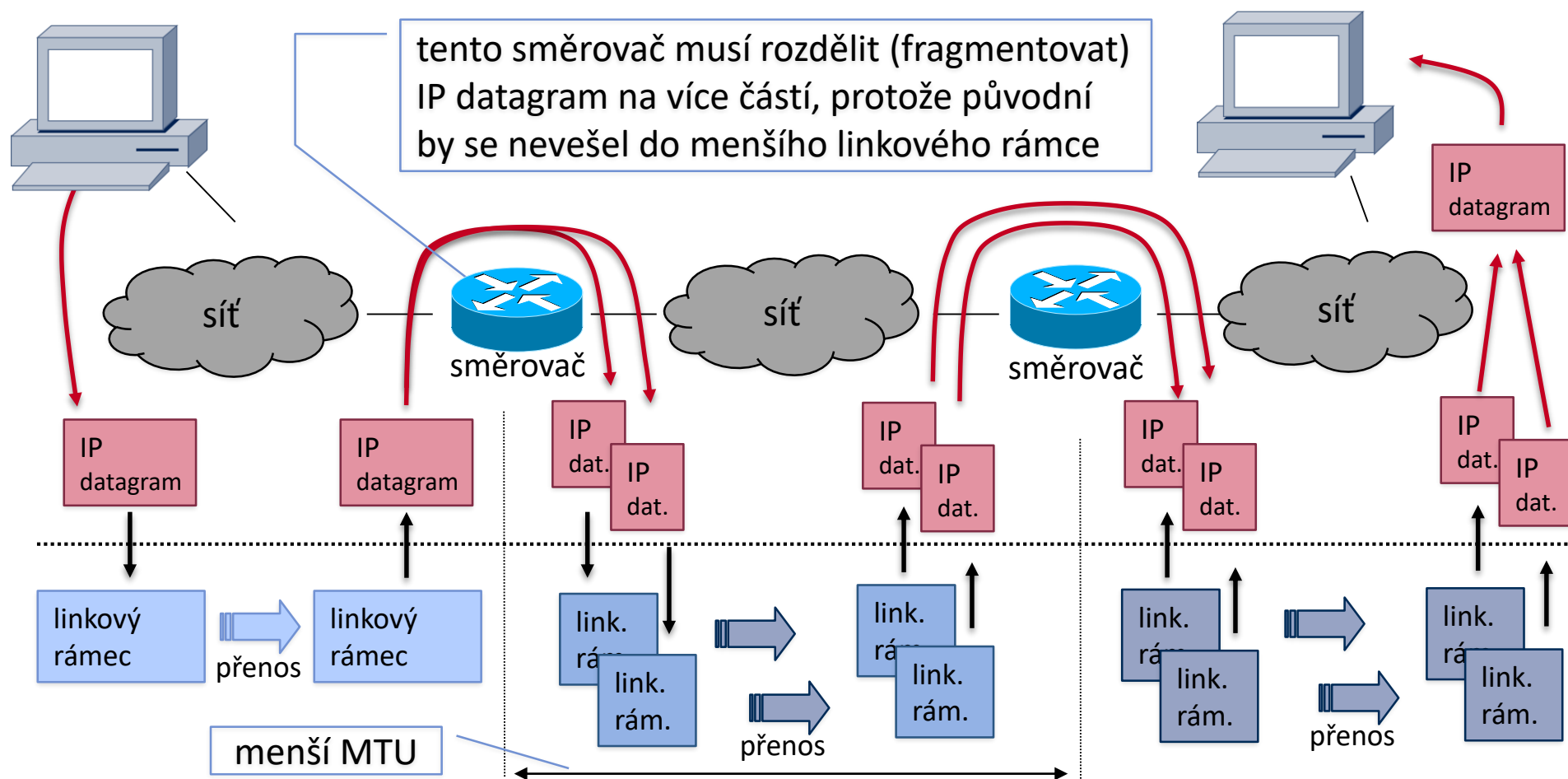
- je k dispozici možnost **fragmentace**
 - rozdělení „příliš velkého“ IP datagramu na menší datagramy
 - označované jako **fragmenty**
 - které se „již vejdou“ do linkových rámců
 - a mohou se přenášet samostatně



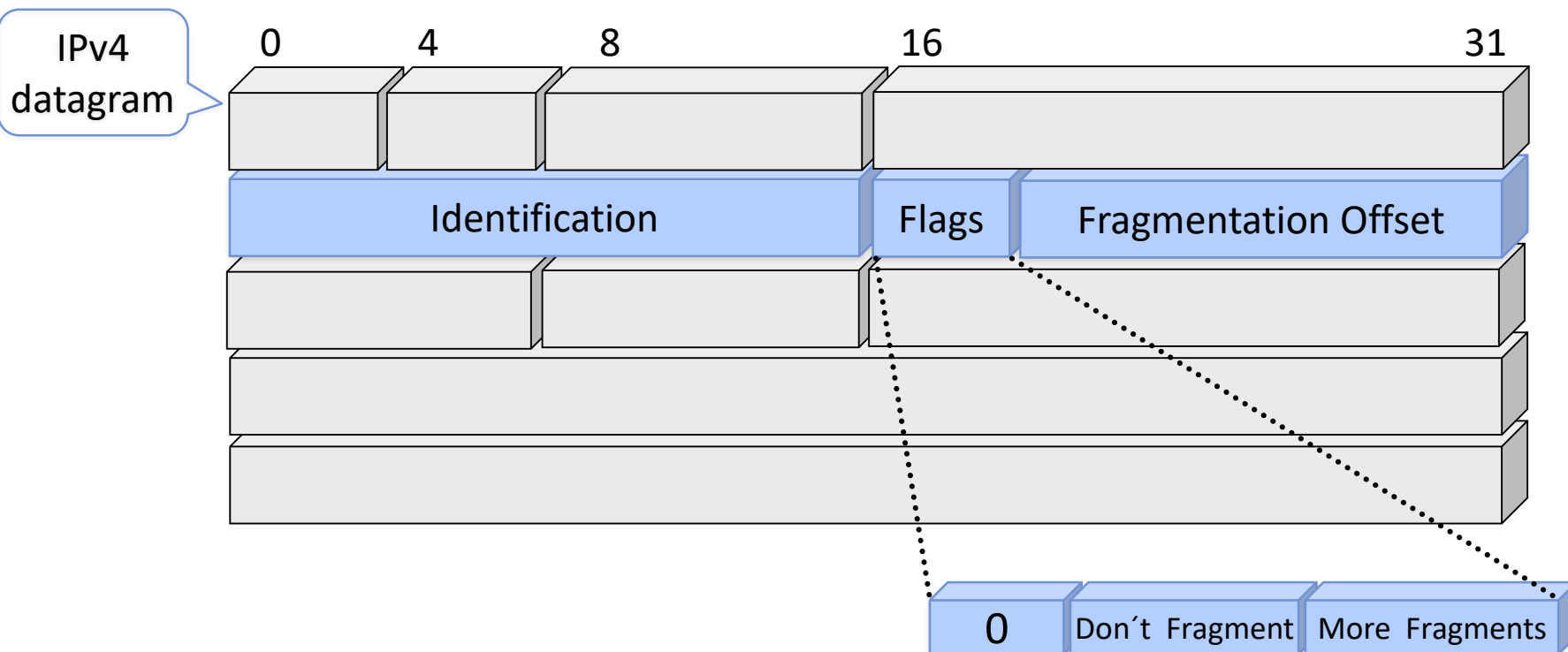
- fungování:

- IPv4:
 - fragmentovat může odesílající uzel i kterýkoli směrovač „po cestě“**
- IPv6:
 - fragmentuje jen odesílající uzel**

- 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)

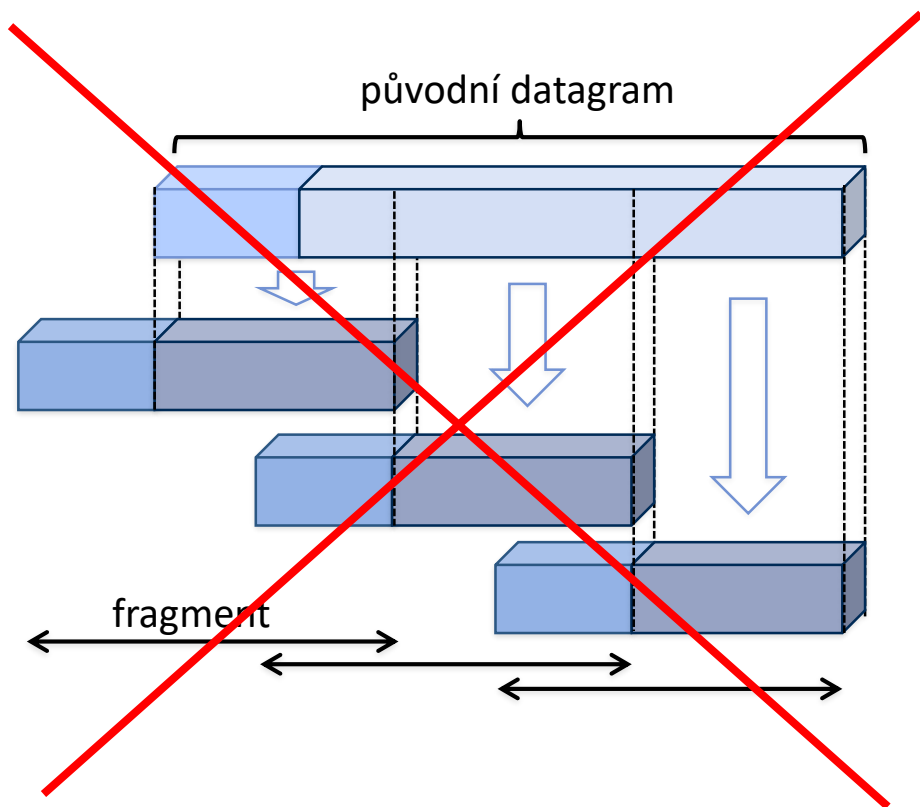


- podmínkou pro fragmentaci je podpora v protokolu IP
 - IPv6 ji řeší pomocí rozšiřujících hlaviček
 - které připojuje k základní hlavičce až v případě potřeby
 - až když se skutečně fragmentuje
 - IPv4 ji řeší položkami, které jsou přítomné v každé hlavičce
 - a tudíž jsou zbytečné (a zabírají místo)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

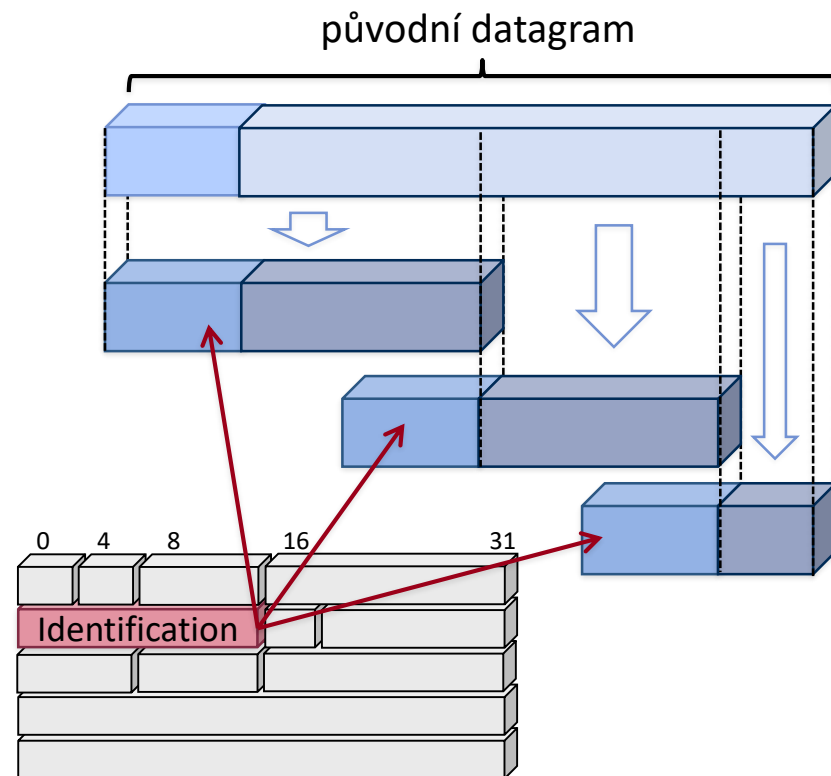


způsob fragmentace v IPv4

- fragmentace nepředstavuje zapouzdření původního IP datagramu

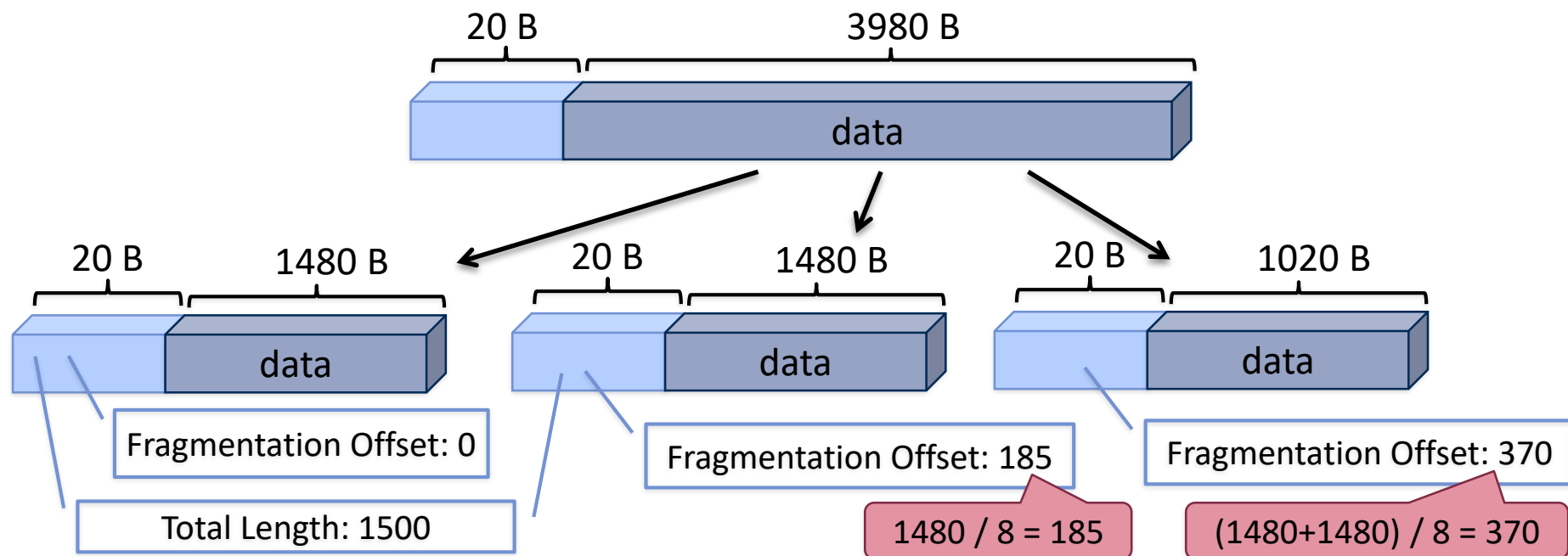


- ale překlad (transformaci) původního datagramu



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

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



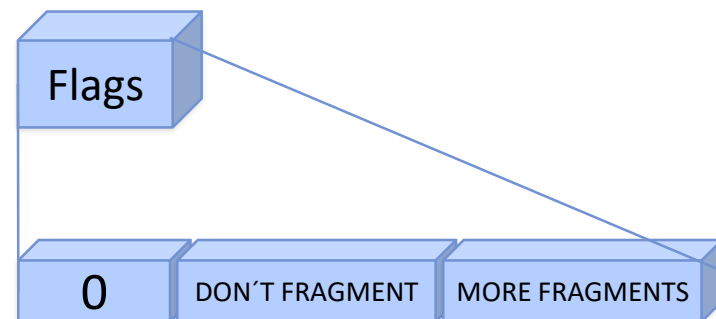
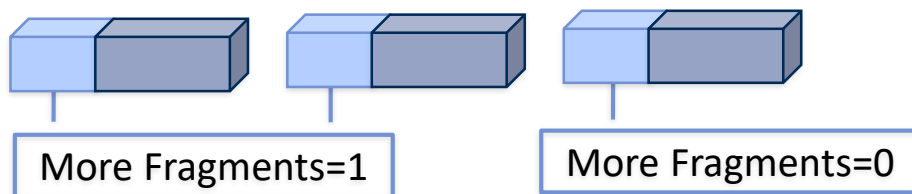
- příznaky fragmentace (Flags):

- **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“)

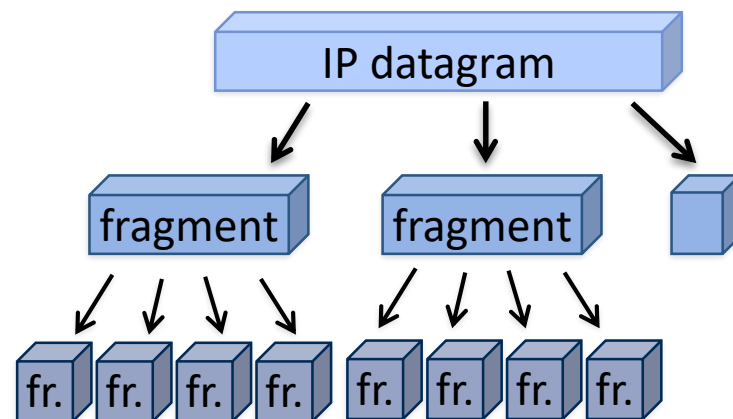
- **More Fragments**, 1 bit

- příznak, udávající zda jde o poslední fragment
 - 1 = nejde o poslední
 - 0 = jde o 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 zkopírován i do jednotlivých fragmentů, nebo nikoli
- připomenutí:
 - (u IPv4): fragmentovat může jak odesílající 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 neprijdou (do zvoleného časového limitu) všechny fragmenty, musí všechny dosud přijaté fragmenty zahodit
 - a odesílateli pošle **ICMP** zprávu **Time Exceeded**

- protokol IP je velmi jednoduchý a přímočarý
- postrádá:
 - mechanismy pro signalizaci (hlášení) chyb a nestandardních situací
 - například zahození 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
 -

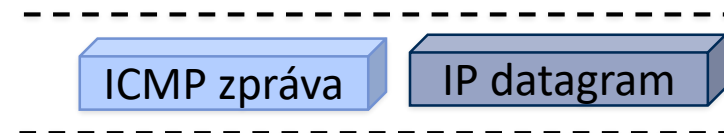
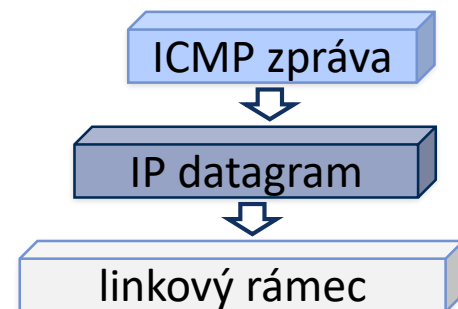
jak se přenáší ICMP zprávy?

- 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 !!



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 vrstevných 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



- lze je dělit na:

- chybové zprávy

- informují o chybách, nejčastěji při zpracování IP datagramů

- dotazy, výzvy, odpovědi a informační zprávy

- informují o význačných skutečnostech, vznášejí dotazy/podněty a reagují na ně

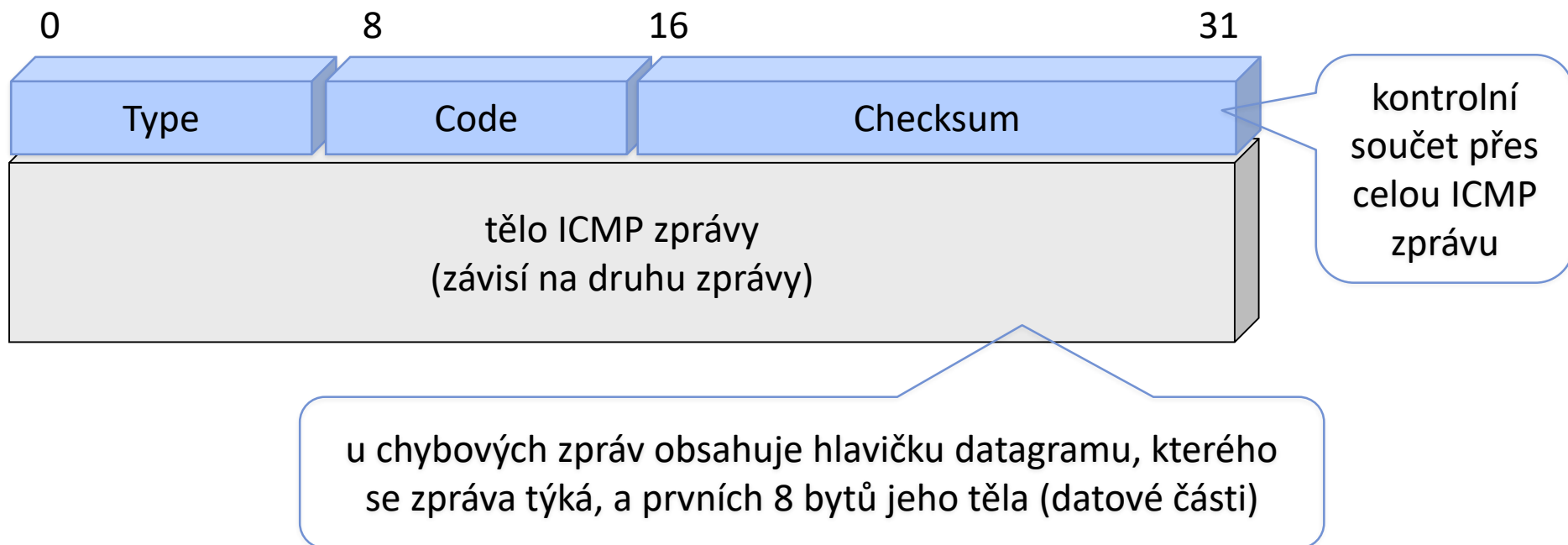
- rozlišují se podle:

- ICMP Message Type, 8 bitů

- (hlavní) typ ICMP zprávy

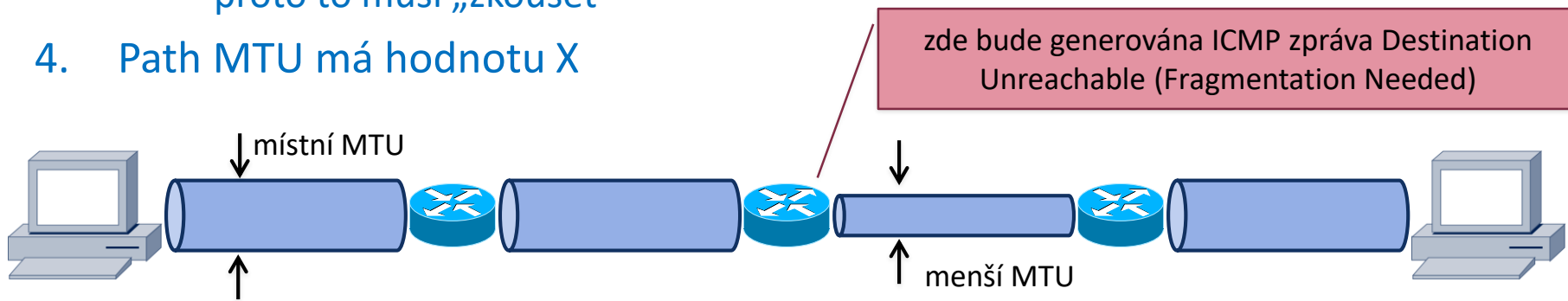
- ICMP Message Code, 8 bitů

- podtyp, upřesňuje druh zprávy



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 hodnotě MTU mělo dojít k fragmentaci
 - proto to musí „zkoušet“
 4. Path MTU má hodnotu X



- **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

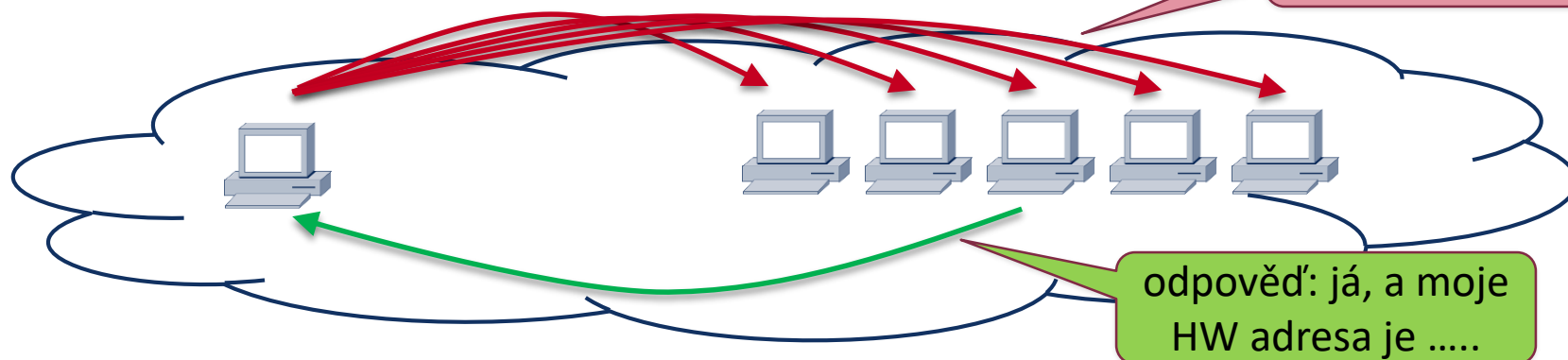


HW adresa

(např. Ethernetová
adresa)

broadcast musí být k dispozici,
jinak ARP nelze použít

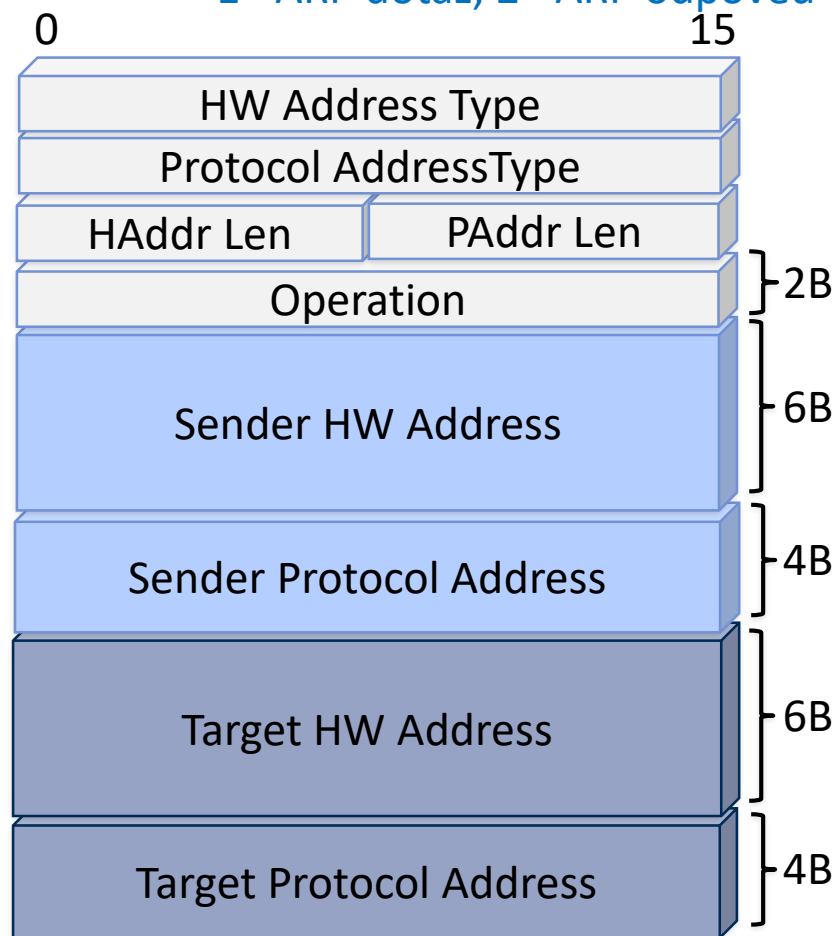
dotaz: kdo z vás má
tuto IP adresu?



odpověď: já, a moje
HW adresa je

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:

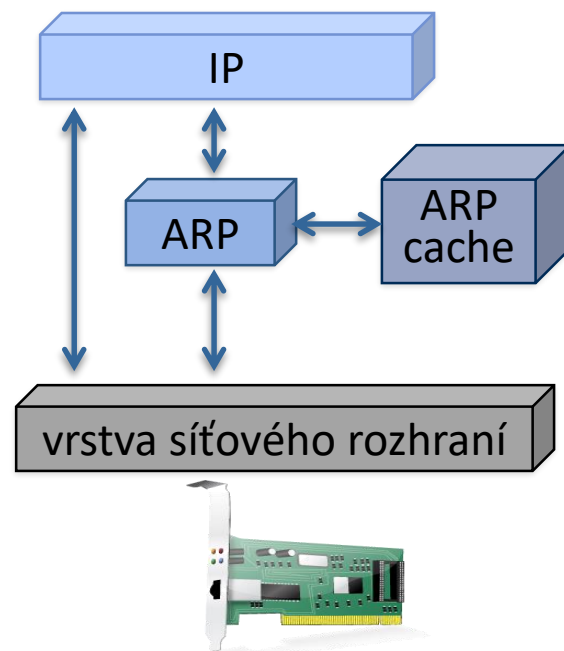
- Sender HW Address = jeho HW adresa
- Sender Protocol Address = jeho IP adresa
- Target Protocol Address = IP adresa, na kterou se ptá

vyplní uzel, který odpovídá na dotaz:

- Target HW Address = jeho HW adresa

- protokol ARP má nezanedbatelnou roli
 - 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



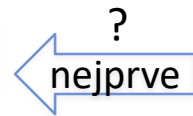
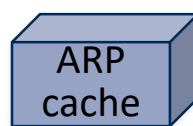
zpracování ARP dotazu

- 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

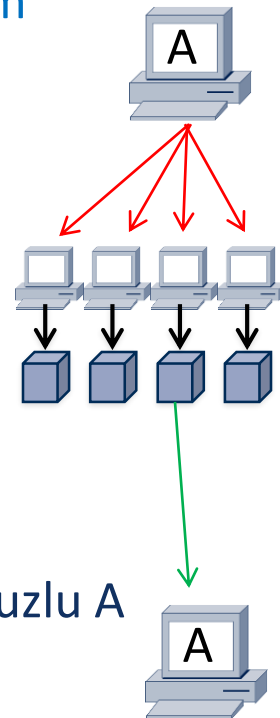
- vyplní v ní svou IP a HW adresu, a IP adresu uzlu B

- 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 uzlem B (zda IP adresa uzlu B je jeho IP adresou)
 - pokud ne, ARP zprávu zahodí a končí

- uzel B:

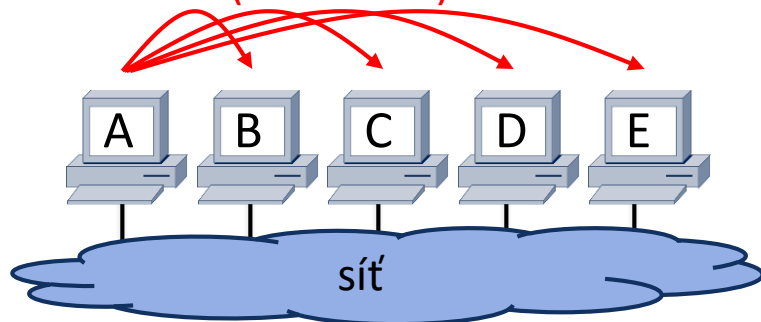
- si do své ARP cache zanesou vazbu (binding) mezi IP a HW adresou uzlu A
 - nebo ji osvěží, pokud ji ve své ARP cache již měl
- sestaví ARP zprávu s odpovědí a odešle ji cíleně (unicast-em) uzlu A
 - v dotazu přehodí Sender/Target, příznak Dotaz/Odpověď a vyplní svou HW adresu



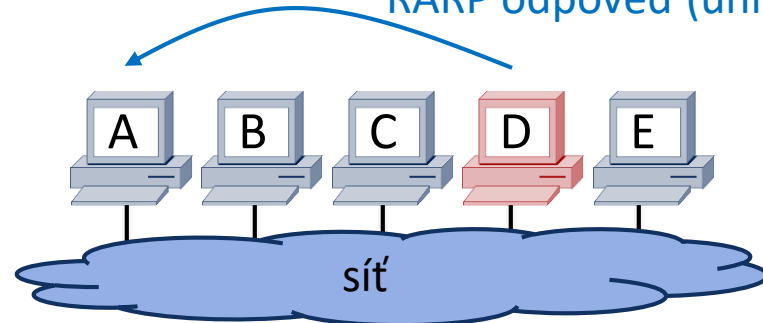
reverzní ARP (RARP)

- 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 uzlům
- 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 uzlů více, může odpovědět kterýkoli z nich

RARP dotaz (broadcast)

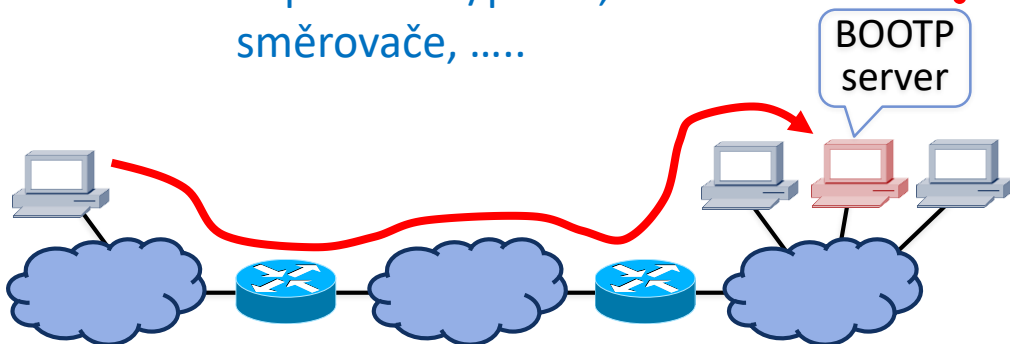


RARP odpověď (unicast)



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)



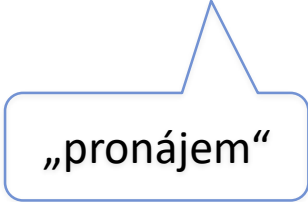
- protokol BOOTP přiřazuje IP adresy trvale (staticky)
 - jakmile uzel dostane přidělenou IP adresu, může ji používat libovolně dlouho
- dnešní sítě potřebují přidělovat IP adresy dočasně (**dynamicky**)
 - na omezenou dobu – s možností následně přidělit stejnou IP adresu jinému uzlu
 - kvůli tomu, že 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

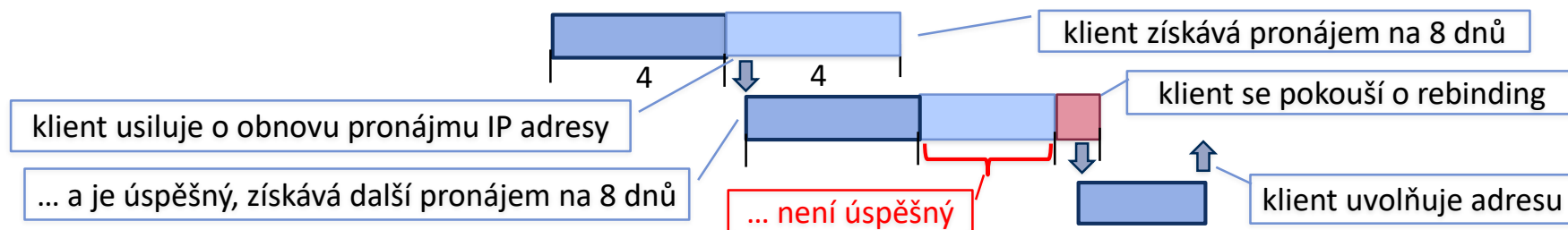
DHCP lease („pronájem“)

- 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 pronajatu (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“

- 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 T1 = 50% 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 T2 = 87,5% l.t.)
 - **uvolnění**: klient „vrací“ pronajatou IP adresu ještě před koncem jejího pronájmu



Rodina protokolů TCP/IP verze 3

Téma 8: Protokol IPv6

Jiří Peterka

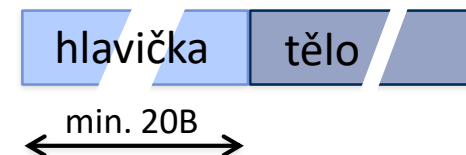
- připomenutí:
 - blokům, které IPv6 přenáší, se častěji říká **pakety** (než datagramy, jako u IPv4)
 - ale nepanuje v tom konsensus, i některá novější RFC hovoří o IPv6 datagramech
 - protipříkladem je možnost použití velkých paketů, tzv. jumbogramů
 - fakticky v tom není rozdíl, protože i IPv6 přenáší své pakety nespojovaným způsobem
 - takže je na místě i jejich označení jako datagramy
- skutečné rozdíly oproti IPv4:
 - jednodušší formát paketu
 - jiný význam položek hlavičky
 - rozšiřující hlavičky
 - jsou úspornější a efektivnější
 - než volitelné položky
 - povinná podpora multicastu
 - u IPv4 je dobrovolná
 - jiný přístup k fragmentaci
 - fragmentovat může jen odesílající uzel
 - „modernější“ směrování
 - lepší podpora hierarchického směrování
 - zabudovaná podpora bezpečnosti
 - povinný IPSEC
 - podpora pro alokaci zdrojů a QoS
 - podpora mobility
 - možnost velkých paketů
 - tzv. jumbogramů (nevyužíváno)
 -

a samozřejmě se používají
větší (128-bitové) IPv6 adresy

- více hlaviček místo jedné

- IPv4 datagram má jen jednu hlavičku proměnné velikosti

- minimálně (a obvykle) 20 bytů, včetně 2 IPv4 adres



- IPv6 paket má více hlaviček

- (povinnou) **základní hlavičku** (main header): vždy 40 bytů, včetně 2 IPv6 adres
- další **rozšiřující hlavičky** (extension headers): proměnné velikosti
 - připojují se pouze v případě, že jsou skutečně zapotřebí !!!

- dále:

- méně položek v základní hlavičce

- „méně potřebné“ položky byly přesunuty do rozšiřujících hlaviček
 - např. položky sloužící potřebám fragmentace mají vlastní rozšiřující hlavičku

- přejmenování některých položek

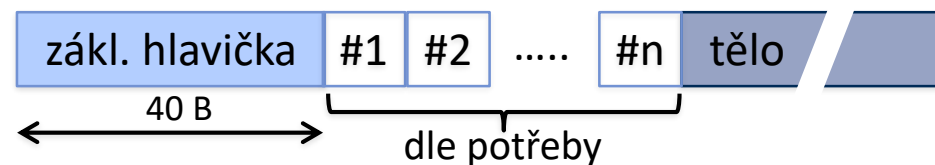
- aby lépe odpovídalo jejich skutečnému významu

- odstranění položky pro délku hlavičky (není potřeba) a kontrolního součtu

- který se musel přepočítávat při každém přeskočení (u IPv6 není co přepočítávat)

- lepší podpora QoS

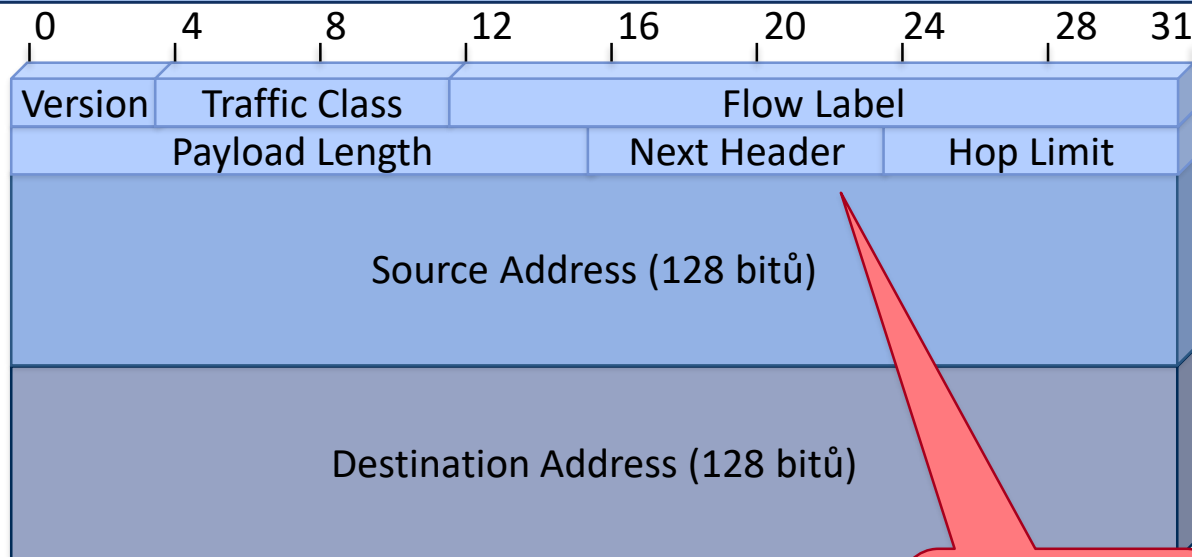
- skrze položky Traffic Class u Flow Label v základní hlavičce



základní hlavička IPv6 paketu

- význam položek:

- Version: 6
- Traffic Class
 - nahrazuje byte ToS v IPv4 datagramu
 - pro Differentiated Services
- Flow Label
 - další podpora QoS a přenosům v reálném čase
 - datagram se přihlašuje k určitému proudu (flow) pro potřeby QoS
- Payload Length
 - velikost nákladu (16 bitů, maximální velikost paketu je tedy $2^{16} = 64$ kB)
 - nezahrnuje základní hlavičku (ale zahrnuje rozšiřující hlavičky, pokud jsou použity)
- Next Header:
 - nejsou-li přítomny rozšiřující hlavičky: udává typ nákladu (jako Protocol u IPv4)
 - jsou-li přítomny rozšiřující hlavičky, udává typ první z nich
- Hop Limit
 - jako TTL u IPv4, jen lépe vystihuje skutečný význam

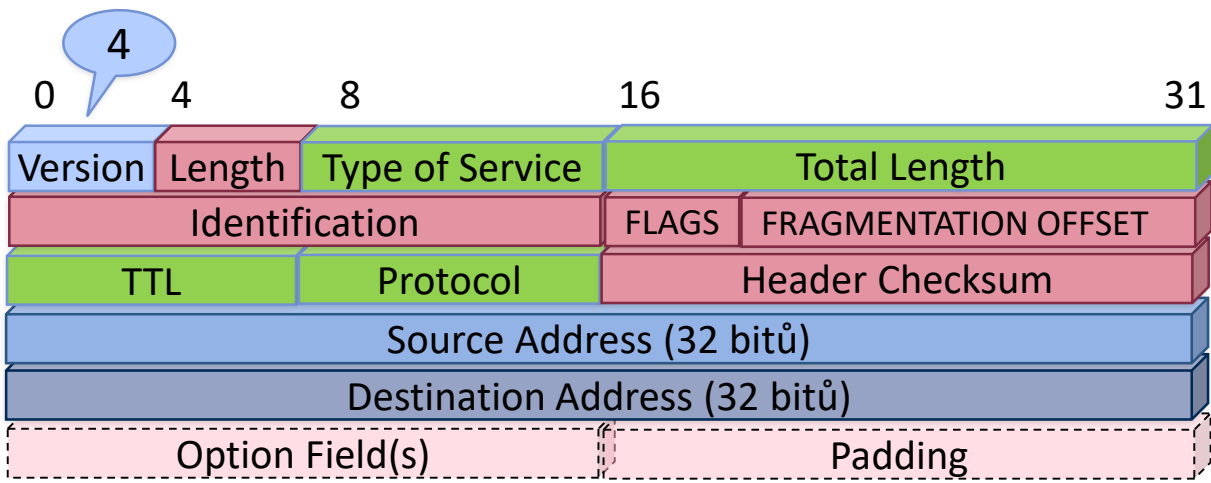


jen 8 bytů navíc
ke 2 IPv6
adresám
(u IPv4 to je 12)

srovnání hlaviček IPv4 a IPv6

- **beze změny:**
 - Version
- **zvětšeno (32 na 128 bitů):**
 - Source Address
 - Destination Address
- **přejmenováno:**
 - ToS na Traffic Class
 - Total Length na Payload Length
 - TTL na Hop Limit
 - Protocol na Next Header
- **přidáno:**
 - Flow Label
- **odstraněno:**
 - Header Length
 - Header Checksum
 - Identification, Flags, Fragmentation Offset
 - Option Field(s), Padding

nemusí se
přepočítávat



u IPv6 jde pouze o základní hlavičku !!!

toky (flow)

- tok = skupina paketů, které spolu „nějak souvisí“
 - mají stejného odesilatele i příjemce, a navíc i nějaký stejný význam/smysl
- v IPv4 by toku odpovídala posloupnost datagramů se stejnou pěticí:

- transportní protokol,
- zdrojová IP,
- zdrojový port,
- cílová IP,
- cílový port

jenže toto jsou údaje, dostupné až na transportní vrstvě

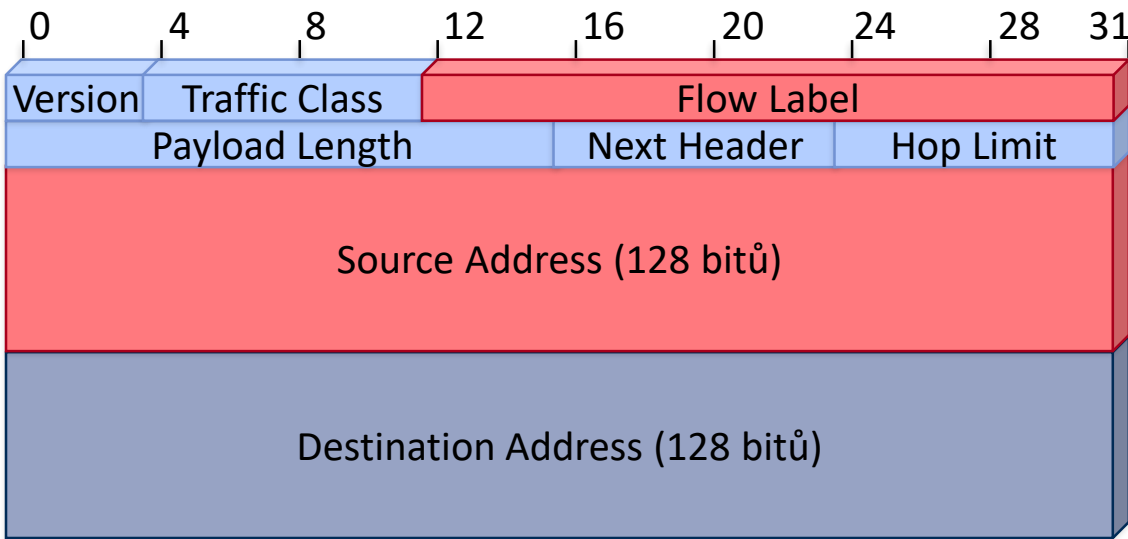
směrovač, který by je chtěl identifikovat (a zpracovávat) jako proud, by musel analyzovat nákladovou část datagramu

- v IPv6 je tok identifikován:

- zdrojovou IPv6 adresou
- identifikátorem toku
 - položkou Flow Label
 - již v základní hlavičce

- není nutné analyzovat data v těle paketu

- potřebné info je dostupné již na úrovni síťové vrstvy

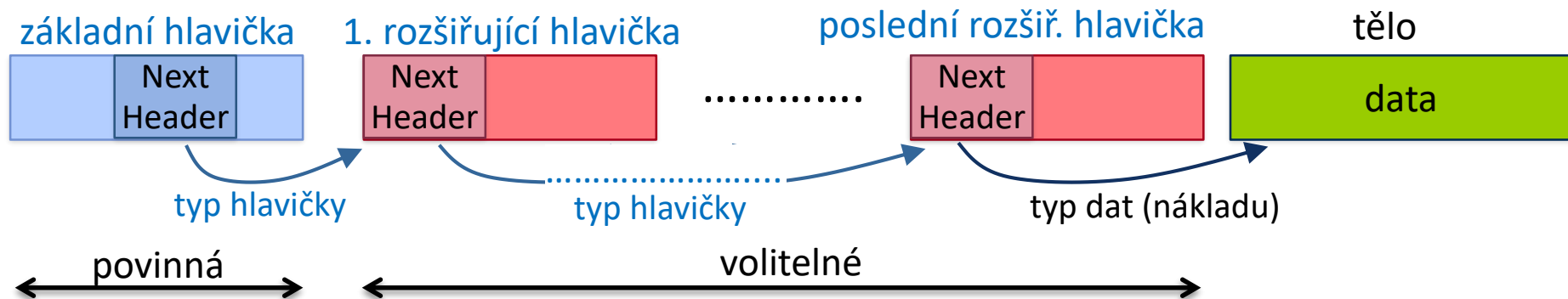


- využití toků (flows)
 - nejde ani tak o zjednodušení směrování
 - resp. jeho nahrazení přepínáním (switchingem), jako např. u MPLS
 - předpokládá se spíše podpora QoS
 - různé nakládání s pakety, podle toho, do jakého patří toku
 - požadavky na QoS mohou být předkládány:
 - stavově: dopředu se sdělí všem směrovačům „po cestě“
 - prostřednictvím „rezervačního“ protokolu, např. RSVP
 - toto „sdělení“ ale musí být časově omezeno
 - bezstavově: každý paket si své požadavky nese v sobě, v rozšiřující hlavičce
 - hlavičce Hop-by-Hop
 - problém:
 - konkrétní fungování dosud není dostatečně rozpracováno, v praxi se nepoužívá
- třída provozu (Traffic Class)
 - 1 byte v základní hlavičce
 - je určen pro vyjádření priority
 - třídy provozu
 - pro Differentiated Services



- DSCP: Differentiated Services Code Point
 - specifikuje prioritu
- ECN: Explicit Congestion Notification
 - určuje, zda směrovač má zahazovat pakety a/nebo posílat info o zahlcení

rozšiřující hlavičky

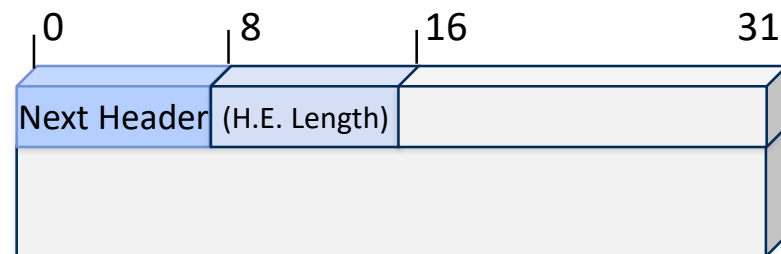


- rozšiřující hlavičky: jsou volitelné (nepovinné)
 - nahrazují doplňky (options) v IPv4, ale šikovněji
 - „odlehčují“ hlavičky IPv6 paketu
 - to, co je v hlavičce IPv4 datagramu využíváno jen občas, se v IPv6 přesouvá do rozšiřující hlavičky
 - nabízí více možností, než má IPv4
- jsou řazeny v sérii za sebou
 - položka Next Header udává **typ další rozšiřující hlavičky**
 - nebo **typ nákladu** v těle paketu (u poslední hlavičky)
 - speciální typ 59: už nenásleduje nic, ani tělo paketu
 - pokud by následovalo, musí být ignorováno
- identifikátory typů v položce Next Header
 - stejně jako u IPv4
 - spravuje je IANA
 - Protocol Numbers
 - například:
 - 0 = Hop-by-Hop options (typ hlavičky)
 - 6 = TCP, 17 = UDP (typ nákladu)
 - 44 = fragmentace
 - 59 = nic nenásleduje

rozšiřující hlavičky

- některé rozšiřující hlavičky mají proměnnou velikost, ostatní pevnou

- 1. položkou je vždy Next Header (1 byte)
 - zbytek se liší
 - podle toho, o jakou rozšiřující hlavičku jde
- u položek s proměnnou velikostí:
 - 2. položkou je údaj o velikosti (Header Extension Length)



výjimka: Destination Options může 2x

- řazení rozšiřujících hlaviček:

- každý typ rozšiřující hlavičky by se (v každém paketu) měl vyskytnout nejvýše 1x
- pořadí není striktně předepsáno, pouze velmi doporučeno
 - cílem je dát na začátek řetězce ty hlavičky, které je třeba zpracovat nejdříve
 - hlavně usnadnit práci směrovačům, aby nemusely analyzovat všechny hlavičky

1. základní hlavička IPv6

2. Hop-by-Hop Options

- týká se „všech uzlů po cestě“

3. Destination Options

- pro uzly v rámci source routing

4. Routing (směrování)

5. Fragment (fragmentace)

6. Authentication, AH (autentizace)

- pro potřeby IPSEC

7. Encapsulating Security Payload, ESP

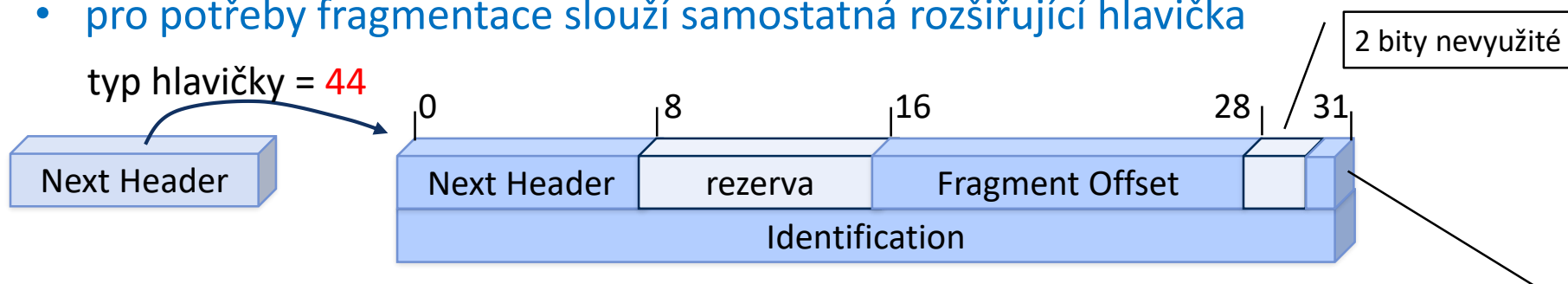
- šifrování obsahu (pro IPSEC)

8. Destination Options

- pro konečného příjemce

9. Mobility

- rozdíly oproti IPv4:
 - v IPv6 může fragmentovat **pouze odesílající uzel !!**
 - pokud se chce „vyhnout problémům“, může:
 1. posílat IPv6 paket do max. velikosti **1280 bytů**
 - v IPv6 je garantováno, že takto velké pakety projdou bez nutnosti fragmentace
 2. zjistit si nejmenší MTU po cestě
 - skrze techniku MTU Path Discovery
 - směrovače v IPv6 už nefragmentují
 - aby nebyly zatěžovány dalšími úkoly
 - pokud směrovač narazí na paket, který by měl být fragmentován, musí ho zahodit
 - pro potřeby fragmentace slouží samostatná rozšiřující hlavička

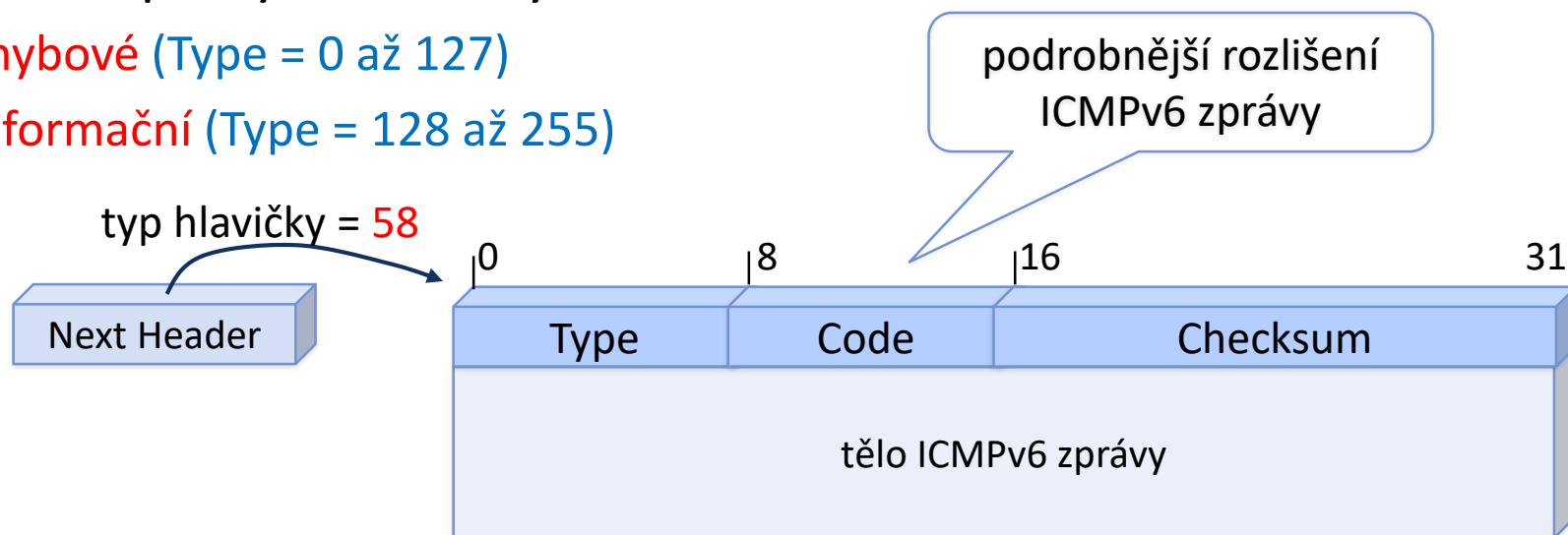


- **Fragment Offset**: stejně jako Fragmentation Offset u IPv4
 - posunutí vůči začátku datové části původního paketu
- **Identification**: stejné jako u IPv4 (ale v rozsahu 32 bitů)

More Fragments Flag:
1: jsou další fragmenty
0: nejsou

- slouží
 - ke stejným účelům, jako ICMP (Internet Control Message Protocol) u IPv4
 - pro signalizaci „nestandardních situací“ v důsledku fungování protokolu IP
- zprávy ICMPv6 se přenáší
 - uvnitř IPv6 paketů (Next Header = 58)
 - stejně jako u IPv4, stejné porušení principů vrstevnatého modelu

- ICMPv6 zprávy mohou být:
 - **chybové** (Type = 0 až 127)
 - **informační** (Type = 128 až 255)



- obsahuje také:
 - kontrolní součet (Checksum), 16 bitů, počítaný včetně tzv. pseudohlavičky

Path MTU Discovery

- postup zjišťování (nejmenšího) MTU po cestě od daného uzlu ke zvolenému cíli
- Path MTU Discovery v IPv4:
 - zdrojový uzel odesílá datagramy určité velikosti (začne s velikostí místního MTU)
 - se zakázanou fragmentací (nastaveným bitem Don't Fragment)
 - pokud datagram kvůli velikosti neprojde, dostane zdroj zpět ICMPv4 zprávu **Destination Unreachable**, s **Code=4** (Fragmentation Needed nad DF Set)
 - z ní se ale **nedozví**, jaké je MTU v dalším úseku, který vyvolal potřebu fragmentace
 - proto cílový uzel odhadne novou (nižší) hodnotu MTU a iteruje
 - pokud znovu neprojde, MTU ještě sníží
 - pokud již projde, MTU zase o něco zvýší
- Path MTU Discovery v IPv6:
 - zdrojový uzel odešle paket o velikosti místního MTU
 - pokud paket kvůli velikosti neprojde, dostane zdroj zpět ICMPv6 zprávu **Packet Too Big**
 - ze které se **dozví**, jaké je MTU v úseku, který vyvolal potřebu fragmentace
 - použije novou (nižší) hodnotu MTU (... a celý postup opakuje)

