Téma 13. Počítačové sítě

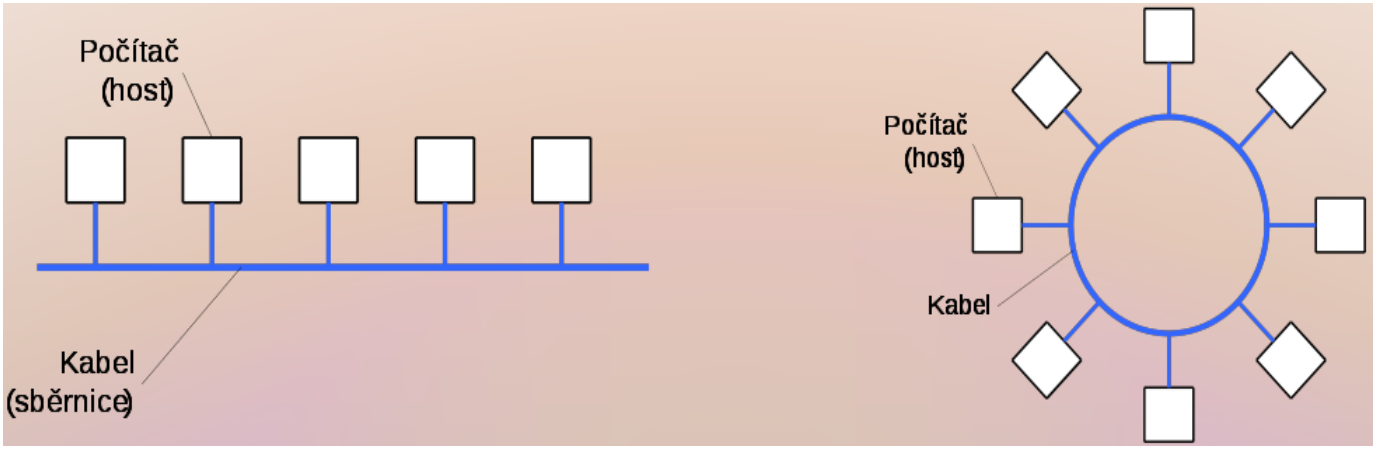
ISO-OSI síťový model

- Vzhledem ke komplexnosti přenosu dat po síti vždy vícevrstvá struktura
- OSI = *Open System Interconnect*
- Model o 7 vrstvách:

Datový element	Vrstva	Účel
Datový tok	Aplikační (*application*)	Koncové aplikace a s nimi spojené komunikační a formátové protokoly včetně síťového API (např. FTP, HTTP, DNS, TELNET, ...)
	Prezentační (*presentation*)	Transformace dat do tvaru, který používají aplikace
	Relační (*session*)	Organizace a synchronizace dialogu mezi spolupracujícími systémy a řízení výměny dat
Segment	Transportní (*transport*)	Pravidla pro přenos dat mezi dvěma počítači (koncovými body) včetně zabezpečení kvality přenosu (nesmí se nic ztratit, jindy jsou ale ztráty přípustné)
Paket	Síťová (*network*)	Tvorba a přenos logických jednotek (paketů), logické adresování, určování přenosových cest
Rámec	Spojová (*link*)	Tvorba fyzických přenosových jednotek (rámců), fyzická (hardwarová) adresace, LAN
Bit	Fyzická (*physical*)	Popis fyzického média (kabelů apod.), signálových úrovní, konektorů a dalších technických parametrů, reprezentace logických signálů

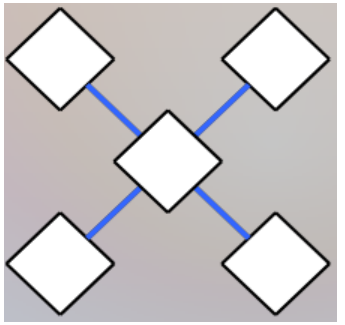
Základní struktury LAN

- Lokální sítě „s vysíláním“ (*broadcast networks*)
  - Sběrnicová a prstencová struktura
  - Každý počítač (uzel) je schopen oslovit všechny ostatní uzly v LAN



- Další možnosti užívané zejména pro propojování LAN
  - dvoubodové spoje (point-to-point)

  - hvězdicová struktura (point-to-multi-point)



Např. připojení strojů k „přístupovému bodu“ WiFi. WiFi však simuluje sběrnicovou strukturu (umí „broadcast“)

Základní technologie LAN

- Technologie Token Ring (dnes již téměř historická technologie IBM)
  - TokenRing 4 Mbit/s TokenRing 16 Mbit/s
  - Prstencová topologie, předávání „tokenu“, 8-mi bitové adresy
  - Formát rámce TokenRing 4 Mbit

Začátek zprávy	Adresa odesilatele	Adresa příjemce	Typ rámce	Datový rámec	Konec zprávy	Parita	Odmítnutí
10 bitů	8 bitů	8 bitů	24 bitů	0-16352 bitů	9 bitů	1 bit	1 bit

- Technologie ethernet (nečastější LAN)
  - Časový multiplex CSMA/CD (= *Carrier Sense Multiple Access with Collision Detection*)
    - Každý uzel začne vysílat, kdykoliv potřebuje a poslouchá, zda slyší to, co říká. Pokud ne, došlo ke kolizi. Oba pak „zmlknou“ a za náhodnou dobu to zkusí znovu.
  - Adresování v ethernetu: jedinečné 48-bitové hardware adresy
  - Formát ethernetového rámce

Preambule	Adresa příjemce	Adresa odesilatele	Typ rámce	Datový rámec	Kontrolní součet
64 bitů	48 bitů	48 bitů	16 bitů	368-12000 bitů	32 bitů

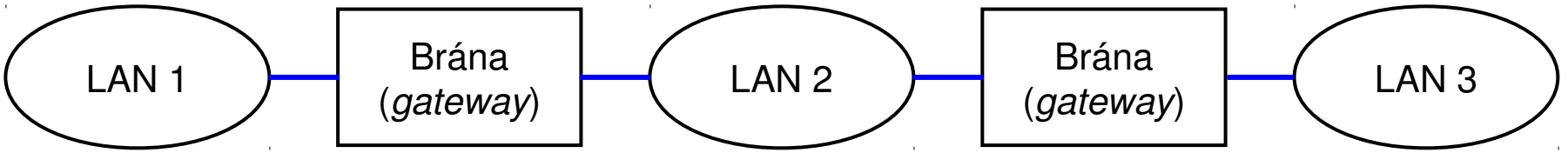
- Obě technologie podporují tzv. "broadcast"
  - tj. oslovení všech zařízení v lokální síti – NÁKLADNÉ

ARP protokol

- Pro doručení Ethernetového datagramu je nutné znát MAC adresu prvního počítače na cestě
- Je nutný převod IP na MAC adresu
- K tomu je protokol ARP, který slouží k přenosu informace o MAC adrese
- Standardní průběh komunikace:
  - Počítač vysílá broadcast s ARP dotazem Who has IP? Dotaz obsahuje MAC a IP adresu vysílajícího počítače.
  - Přijímající počítač si uloží získané informace MAC a IP od vysílajícího počítače pro případné další použití do ARP tabulky MAC↔IP.
  - Počítač s hledanou IP adresou odpoví ARP datagramem obsahující jeho MAC adresu.
  - Vysílající počítač si získanou MAC adresu uloží do ARP tabulky MAC↔IP.
  - Počítač může vyslat data na cestu internetem.

Architektura Internetu

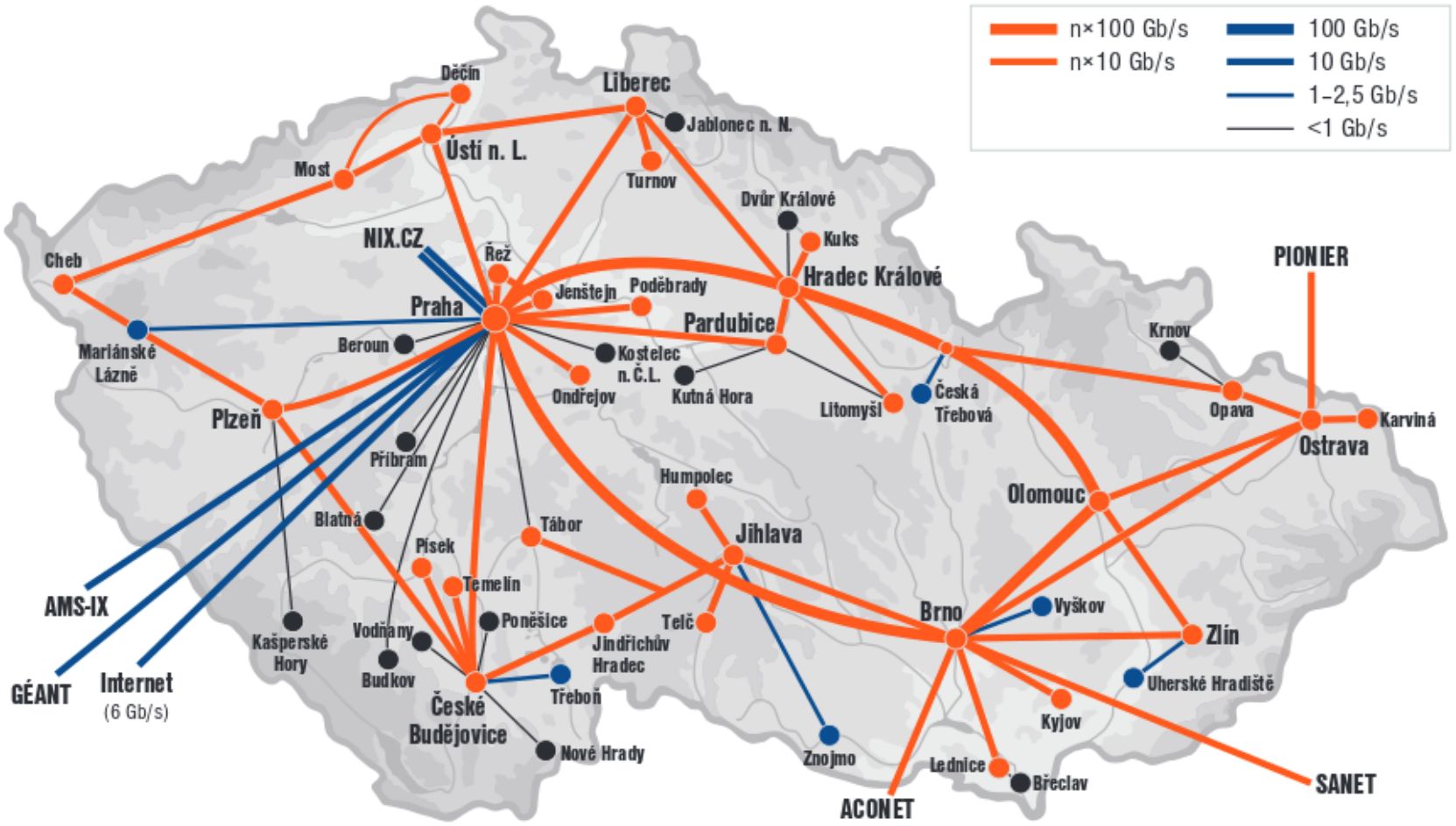
- Základní architektura Internetu (i internetů)
  - internet(working) s malým „i“ = obecné propojení několika LAN



  - Mosty (*bridges*), brány (*gateways*) a směrovače (*routers*) propojují fyzické lokální sítě
  - Mosty propojují segmenty LAN stejných fyzických technologií
    - Mnohdy oddělují provoz na segmentech adaptivním přeposíláním rámců na základě "naučených" fyzických adres
  - Brány propojují LAN s různými technologiemi
    - Velmi často jsou fyzicky integrovány se směrovači
  - Směrovače pracují s datovými jednotkami "vyšší úrovně"
    - Mosty i brány pracují na úrovni spojové (linkové) vrstvy ISO-OSI
    - Směrovače znají informace o sítích a posílají pakety (datagramy ➔) na základě "vyšších" (logických) adres.
    - Např. v IP staví na znalosti o cílové síti (nikoliv o cílovém stroji)
- IP protokoly považují všechny sítě za rovnocenné bez ohledu na jejich fyzickou technologii

CESNET

- CESNET znamená Czech Education and Scientific NETwork
- Cílem je výzkum a vývoj informačních a komunikačních technologií, budování a rozvoj e-infrastruktury CESNET určené pro výzkum a vzdělávání



Internet a jeho řízení

- Historické poznámky
  - ARPA/DARPA – projekt z počátku sedmdesátých let 20. stol.
  - BSD UNIX a systém symbolického adresování strojů prostřednictvím tzv. domén (*Domain Name System* = DNS) – 1984
  - Profesionalizace Internetu (devadesátá léta 20. stol.)
- Řízení Internetu
  - **IAB** = *Internet Architecture Board* – celková architektura
  - **IETF** = *Internet Engineering Task Force* – technologie, protokoly
  - **IANA** = *Internet Assigned Numbers Authority* – přidělování adres, čísel portů, atd.
  - ISOC = *Internet Society* – sdružení profesionálních firem
  - **IESG** = *The Internet Engineering Steering Group* – technická standardizace
- Dokumentace
  - RFC (*Request for Comment*) šířené volně po síti
    - např. http://www.ietf.org/rfc.html

Internetové adresy

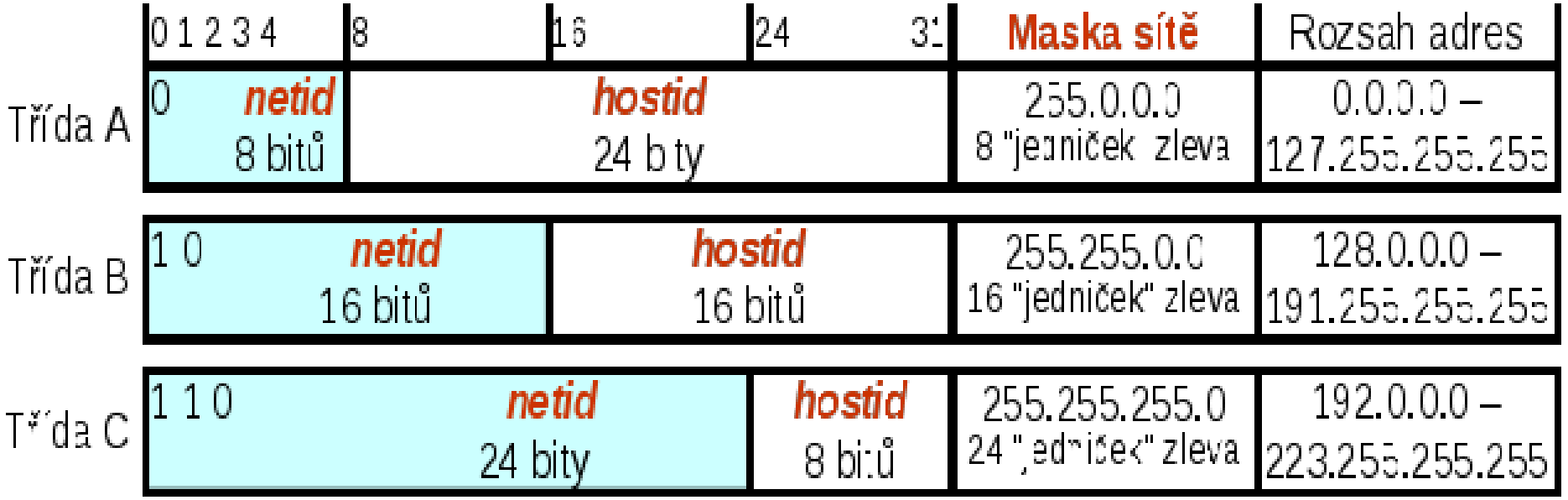
- Základní adresování v Internetu
  - Každý stroj má svoji jednoznačnou identifikaci: tzv. IP adresu
- Současný Internet – v. 4 používá adresy 32 bitů
  - Konvence: 4 dekadická čísla à 8 bitů – 147.32.85.1
- Internet v. 6 má adresy 128 bitů
  - Proč rovnou v 6 – verze 5 byla v roce 1979 použita k definici Internet Stream Protocol, který se moc neprosadil
  - Dataily později
- IP adresa
  - Identifikuje každý jednotlivý síťový adapter
    - Stroj může mít i více adapterů („*multihomed*“ *host*)
    - Identifikace nemusí být jednoznačná: jeden adapter může mít více IP adres
  - Skládá se ze dvou částí
    - Identifikace (adresa) sítě – *netid* (bity vlevo)
    - Identifikace (adresa) stroje v síti – *hostid* (bity vpravo)

Internetové adresy

- Primární třídy IP adres

Třída	Bitový prefix	Počet bitů *čísla sítě*	Počet bitů *čísla stroje*	Počet sítí	Počet adres v síti	Rozsah adres
A	0	8	24	128 = 2^7	16.777.216 = 2^{24}	0.0.0.0 – 127.255.255.255
B	10	16	16	16 384 = 2^{14}	65 536 = 2^{16}	128.0.0.0 – 191.255.255.255
C	110	24	8	2 097 152 = 2^{21}	256 = 2^8	192.0.0.0 – 223.255.255.255
D	1110	nedefinováno		„Multicast" adresy (➔)		224.0.0.0 – 239.255.255.255
E	1111	nedefinováno		Experimentální rozsah		240.0.0.0 – 255.255.255.255

Jiný pohled



Internetové adresy

- Konvence:
  - Adresa sítě je plná IP adresa s *hostid* = 0
  - Adresa tvořená číslem sítě a částí *hostid* tvořenou samými "1" je adresa oslovující všechny stroje v síti (*broadcast address*)
- Maska sítě:
  - „Adresová aritmetika“ $IP\_Adresa \wedge Maska\_Sítě = netid$ $IP\_Adresa \wedge \neg(Maska\_Sítě) = hostid$
    - Nutno znát binární reprezentace dekadických čísel a operace s binárními čísly!
- Adresování CIDR (= *Classless Inter-Domain Routing*)
  - Adresová aritmetika umožňuje efektivnější členění párů *netid* | *hostid* – hranice částí IP adresy může být kdekoliv
  - Maska sítě dána *n* (*n*=0 až 32) jedničkovými bity zleva
  - CIDR notace:
    - IP_Adresa/*n*; příklad: 147.32.85.128 – 147.32.85.191 = 147.32.85.128/26
      ale též = 147.32.85.183/26
    - LAN 192.168.200.64/30 obsahuje 4 adresy: 192.168.200.64 = netid, 192.168.200.65=stroj$_1$, 192.168.200.66=stroj$_2$, 192.168.200.67 = LAN broadcast

Internetové adresy

- Rezervované rozsahy IPv4 adres

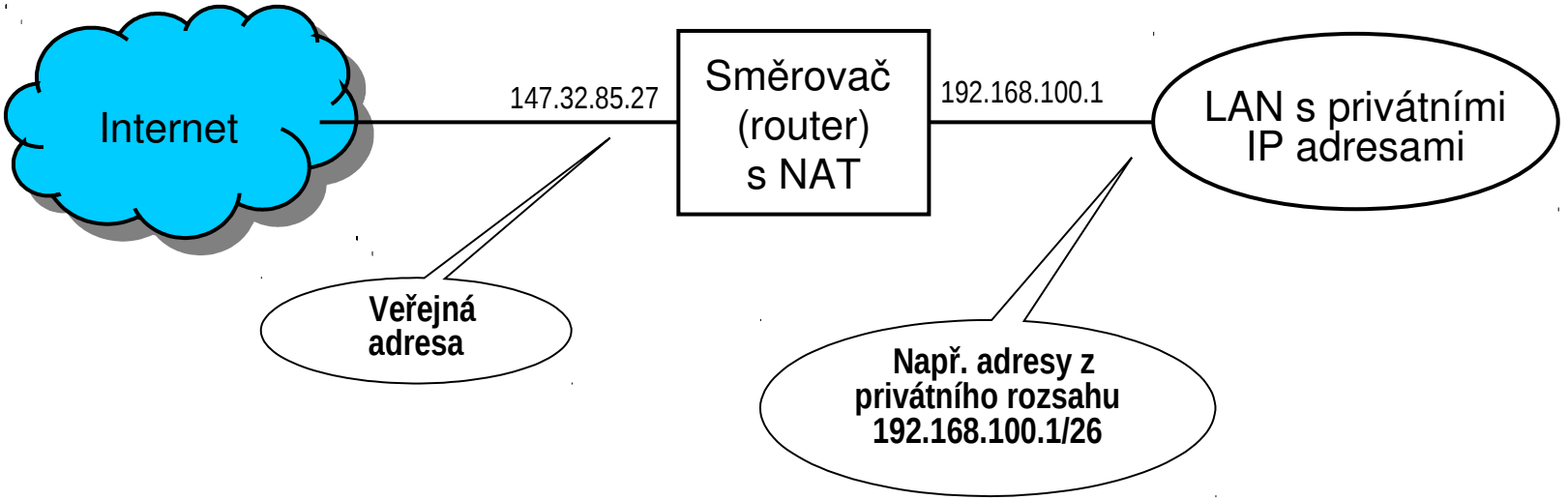
CIDR notace	Rozsah adres	Počet adres	Účel
0.0.0.0/8	0.0.0.0 – 0.255.255.255	16.777.216	„Broadcast" v rámci dané (this) sítě (RFC 1700)
10.0.0.0/8	10.0.0.0 – 10.255.255.255	16.777.216	Privátní rozsah adres (RFC 1918)
127.0.0.0/8	127.0.0.0 – 127.255.255.255	16.777.216	„Loopback" adresy, stroj oslovuje „sám sebe" (obvykle se používá jen **127.0.0.1**)
169.254.0.0/16	169.254.0.0 – 169.254.255.255	65.536	Autokonfigurační rozsah, kdy stroj potřebuje zjistit svoji adresu, obvykle pomocí **DHCP** (➔) protokolu
172.16.0.0/12	172.16.0.0–172.31.255.255	1.048.576	Privátní rozsah adres (RFC 1918)
192.88.99.0/24	192.88.99.0–192.88.99.255	256	Pro mechanismus přechodné migrace mezi IPv4 a IPv6 (RFC 3068)
192.168.0.0/16	192.168.0.0–192.168.255.255	65.536	Privátní rozsah adres (RFC 1918)
198.18.0.0/15	198.18.0.0–198.19.255.255	131.072	Pro testování „inter-network" komunikací (RFC 2544)
224.0.0.0/4	224.0.0.0–239.255.255.255	268.435.456	Viz třída D – „Multicast", tj. komunikace 1:N (RFC 3171)
240.0.0.0/4	240.0.0.0–255.255.255.255	268.435.456	Viz třída E – Rezervováno pro experimentální vývoj protokolů dle zvláštního povolení IANA

Speciální adresy

- Privátní adresy
  - nesmí se šířit po Internetu – směrovače nesmí propustit datagramy s těmito adresami
- "Multicast" adresy
  - jeden stroj rozesílá informace více zaregistrovaným strojům (např. internetová televize)

Internetové adresy (5)

- Šetření IP adresami
  - Užívání privátních adres a jejich překlad na adresy veřejné (NAT = *Network Address Translation*)
    - Množina privátních adres je překládáno na jedinou veřejnou adresu
    - Na privátním rozsahu (za NAT směrovačem) je problém se servery
    - Způsob práce NAT souvisí s IP protokoly, zejména pak s tzv. porty
    - Vrátíme se k tomuto problému a jeho řešení později



Internetové datagramy

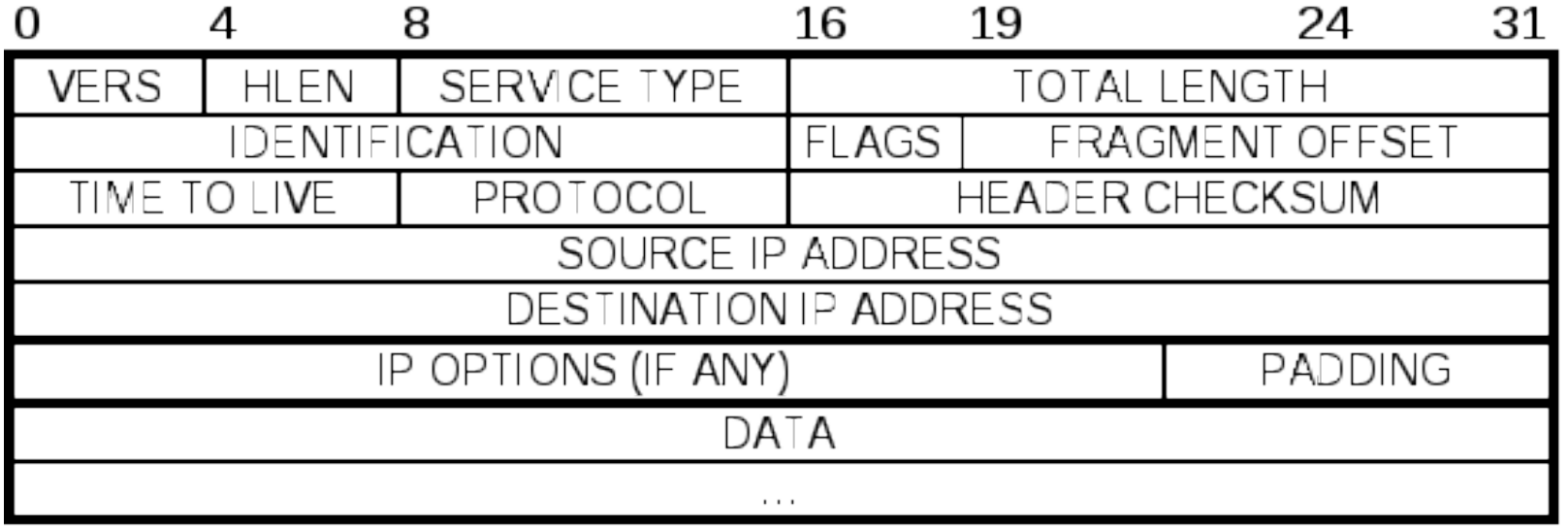
- Internet vytváří virtuální síť a přenáší tzv. IP datagramy
  - Síť představuje systém „s nejlepší snahou o doručování“

    (*best effort delivery*)
  - Datagramy putují po různých fyzických sítích majících různou strukturu a velikost rámců
- Formát IP datagramu

Zapouzdření IP datagramu do rámce fyzické sítě

Hlavička IP datagramu

- Každý IP datagram má hlavičku nesoucí informace důležité pro přenos datagramu od odesilatele k adresátovi



Formát IP datagramu

Význam položek

- VERS: Verze IP protokolu – pro IP v. 4 VERS = 4
- HLEN: Délka hlavičky ve 32-bitových slovech (standardně 5).
- TOTAL LENGTH: Celková délka datagramu v bytech (oktetech) včetně hlavičky – max. 65535 bytů.
- SOURCE IP ADDRESS: IP adresa odesilatele
- DESTINATION IP ADDRESS: IP adresa adresáta

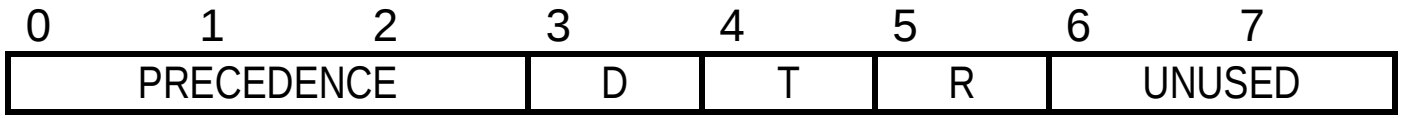
Hlavička IP datagramu (pokračování)

- IDENTIFICATION: obvykle sekvenční nebo náhodné číslo vygenerované odesilatelem datagramu.
- PROTOCOL: Identifikace protokolu IP datagramu (ICMP=1, UDP=17, TCP=6, ...). Definováno v RFC 1060

FLAGS, FRAGMENT OFFSET: Informace o fragmentaci datagramu

TIME TO LIVE (TTL): Určuje jak dlouho smí datagram putovat po Internetu. Každá brána dekrementuje tuto hodnotu; je-li TTL=0 odstraní datagram a pošle ICMP zprávu odesilateli

SERVICE TYPE: Osmibitové pole obsahující pokyny pro směrování paketu



Precedence datagramu: 0=normální, 7=řízení sítě

Malé zpoždění

Vysoká propustnost

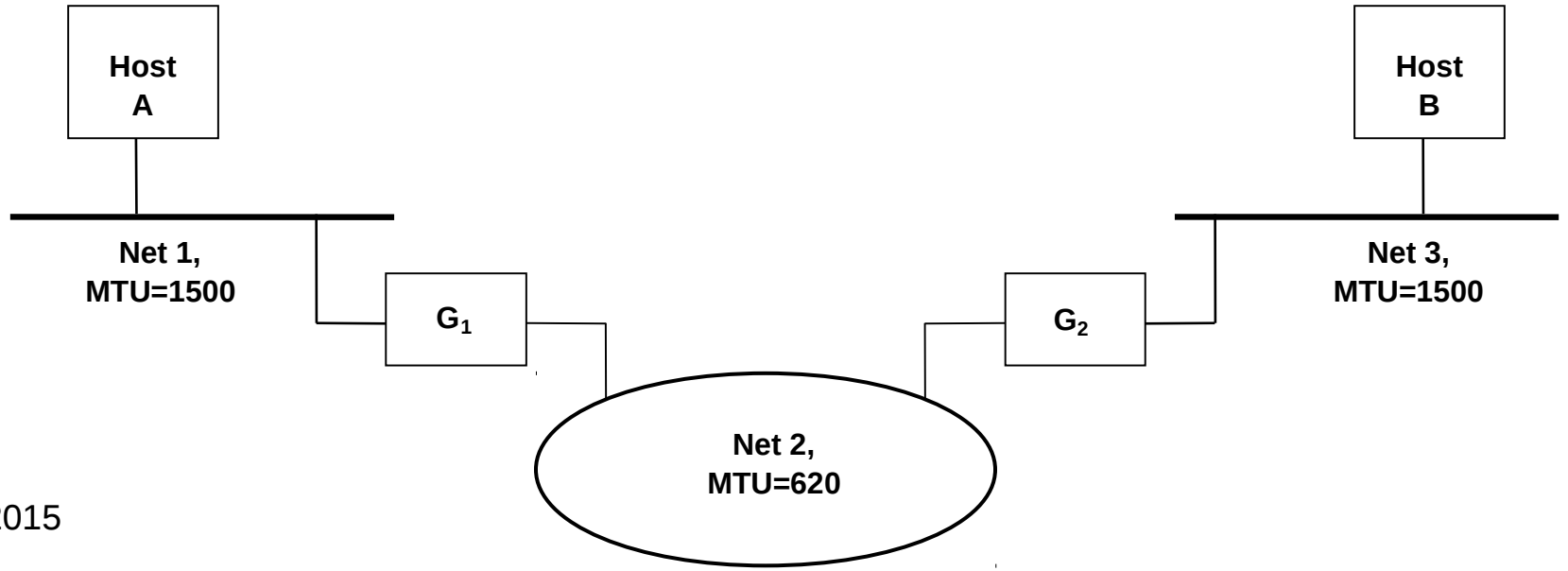
Vysoká spolehlivost

Fragmentace datagramů

- MTU: (*Maximum Transmission Unit*) určuje maximální velikost datagramu, kterou lze přenést po LAN s určitou technologií

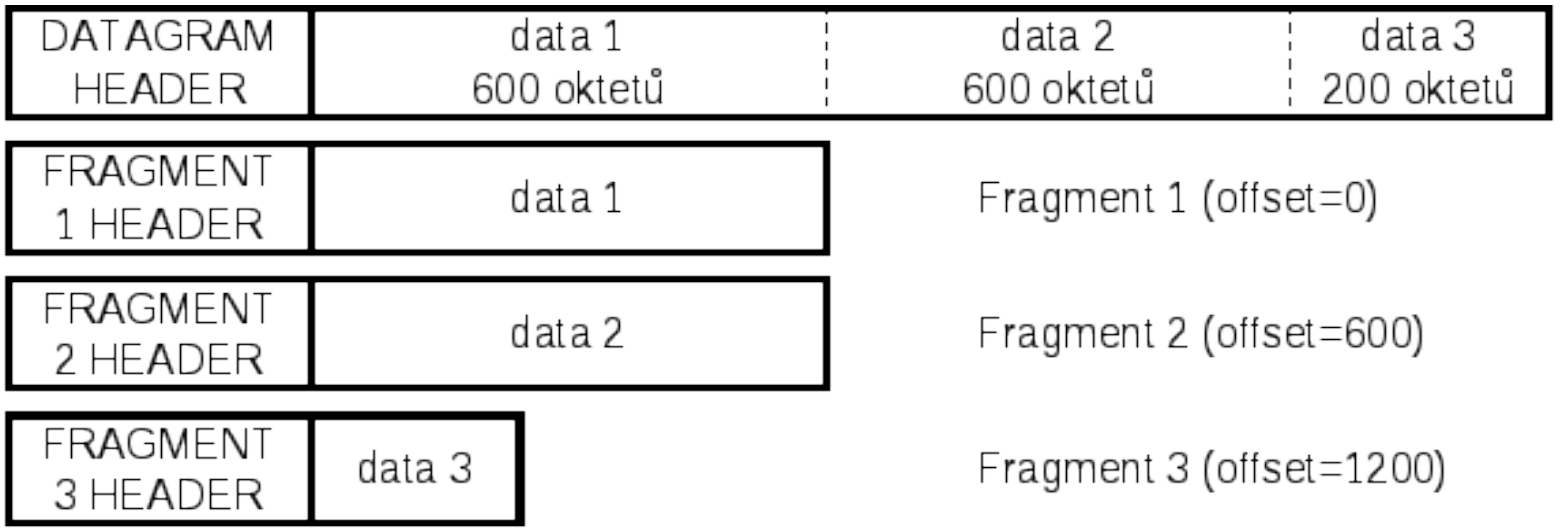
Síť	Implicitní MTU	Síť	Implicitní MTU
PPP	296	X.25	576
Ethernet	1 500	WiFi (IEEE 802.3)	1 492
TokenRing 4Mb	4 464	TokenRing 16Mb	17 914

- Internet – soustava LAN s různými MTU
  - Pokud je datagram větší než MTU, musí se fragmentovat



Fragmentace datagramů

- Fragmentace nastává kdekoliv po cestě datagramu
  - Je-li datagram fragmentován, neskládá se cestou, ale rekonstrukce datagramu je úkolem cílového stroje
  - Každý fragment putuje jako samostatný datagram:
    - Z hlavičky původního datagramu se okopírují pole: VERS, HLEN, SERVICE TYPE, IDENTIFICATION, PROTOCOL, SOURCE IP ADDRESS, DESTINATION IP ADDRESS
    - TOTAL LENGTH se změní na délku fragmentu a položka FRAGMENT OFFSET určuje polohu (offset) fragmentu v původním datagramu
    - Pole FLAGS obsahuje bit: "*more fragments*". Je-li tento bit 0, pak cílový stroj ví, že obdržel poslední fragment, a pomocí polí FRAGMENT OFFSET a TOTAL LENGTH může sestavit originální datagram

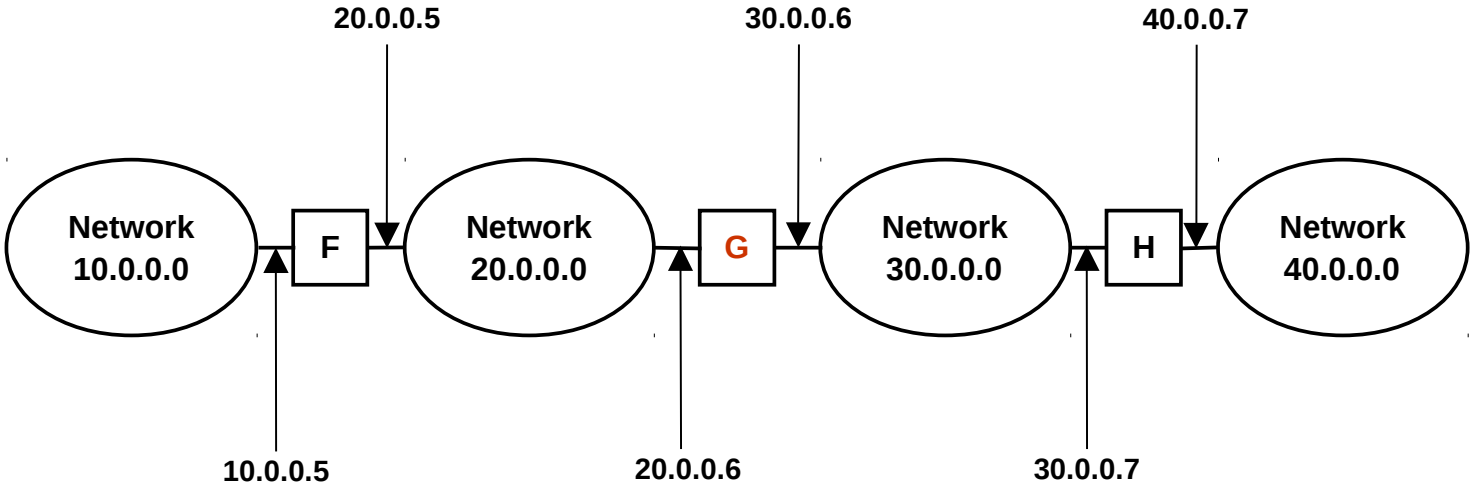


Fragmentace datagramu délky 1400 oktetů při průchodu sítí s MTU = 620

Směrování datagramů

- Směrování (*routing*) je proces rozhodování o cestě, kudy poslat datagram (nebo jeho fragment) k cíli
  - Za směrovač se považuje libovolný stroj schopný přijímat takové rozhodnutí
  - Směrování může být přímé nebo nepřímé
    - Přímé směrování nastává, když je cílový stroj součástí lokální sítě bezprostředně spojené se směrovačem
    - Jinak jde o směrování nepřímé
  - Směrovače v Internetu tvoří kooperativní propojenou strukturu. Datagramy putují od jednoho směrovače k druhému dokud nedosáhnou směrovače, který umí zaslat datagram přímo cílovému stroji
  - Tabulkou řízené směrování
    - Každý směrovač obsahuje tzv. směrovací tabulku tvořenou dvojicemi
      (*N*, *G*), kde *N* je *netid* cílové sítě a *G* je IP adresa "příštího" směrovače podél cesty k cílové síti *N*. „Příští směrovač“ musí být dosažitelný přímo.

Směrování datagramů



Při zasílání stroji na síti	Směruj na adresu
20.0.0.0	Adresuj přímo cílový stroj
30.0.0.0	Adresuj přímo cílový stroj
10.0.0.0	20.0.0.5
40.0.0.0	30.0.0.7

Tabulka směrovače G

- Implicitní směry (*default routes*)
  - Velmi často jsou LAN propojeny se "zbytkem Internetu" prostřednictvím jediného směrovače. Pak tento směrovač představuje pro tzv. default gateway, tj. adresu, kam všechny stroje v LAN posílají datagramy adresované vně LAN

Směrování datagramů

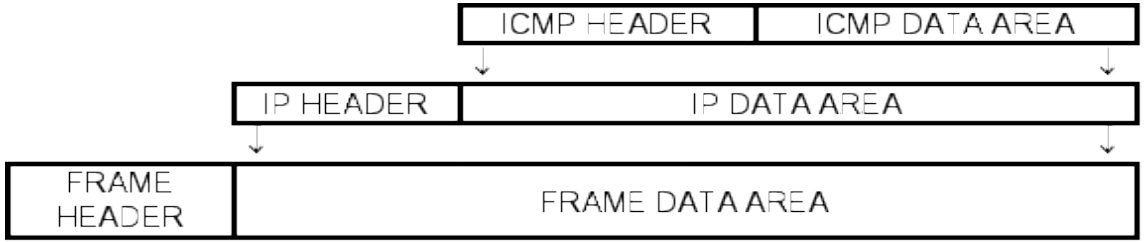
- Specializované směry ke strojům (*Host-Specific Routes*)
  - Někdy je výhodné přiřadit jednomu nebo několika strojům speciální směrovací informaci. Důvody mohou být bezpečnostní, administrativní i technické. Technickým důvodem je např. připojení samostatného stroje po point-to-point spoji (Internetový PPP protokol)
- Směrovací algoritmus:
  1. Vyjmi z datagramu cílovou IP adresu *ID* a s použitím síťové masky urči *netid* cílové sítě
  2. Pokud *ID* odpovídá některému spec. směru (*host-specific route*), pak pošli datagram přímo tomuto stroji
  3. Pokud *netid* se shoduje s některou přímo připojenou sítí, směruj přímo
  4. Pokud *netid* se nachází ve směrovací tabulce, pošli datagram odpovídajícímu směrovači
  5. Pokud bylo specifikováno implicitní směrování (*default route*), pošli datagram na "*default gateway*"
  6. Jinak oznam chybu směrování zasláním ICMP zprávy odesilateli (*Destination unreachable*)

Lokální doručení datagramu

- Přímé směrování musí doručit datagram lokálně
  - Totéž se děje při předání datagramu přímo dostupnému směrovači připojenému přes LAN (nikoliv při *point-to-point* spoji)
  - Datagram obsahuje IP adresu, avšak doručit je nutno na fyzickou adresu uvnitř LAN
- Mapování IP adres na fyzické adresy
  - ARP (= *Address Resolution Protocol*) – dynamické mapování
  - Řešení v "broadcast" LAN – zaslání datagramu strojem A s IP adresou I_A stroji B, který má IP adresu I_B
    - Odesilatel zná svoji IP adresou I_A a i fyzickou adresou F_A, a potřebuje zjistit fyzickou adresu F_B k jemu známé IP adrese I_B
    - Vyšle „ARP broadcast“ rámec, v jehož datové části bude vedle I_A i I_B. Tento rámec přijmou všechny stroje v LAN.
    - Stroj, který rozpozná svoji adresu I_B, na tuto „všeobecnou výzvu“ odpoví a sdělí tak odesilateli svoji fyzickou adresu F_B.
    - "Broadcast" však zatěžuje LAN, proto si tazatel získanou F_B jistou dobu (standardně 5 minut) pamatuje.
    - Vzhledem k tomu, že se dá očekávat brzká odpověď $B \rightarrow A$, stroj B získá a zapamatuje si z ARP rámce i adresy I_A a F_A.

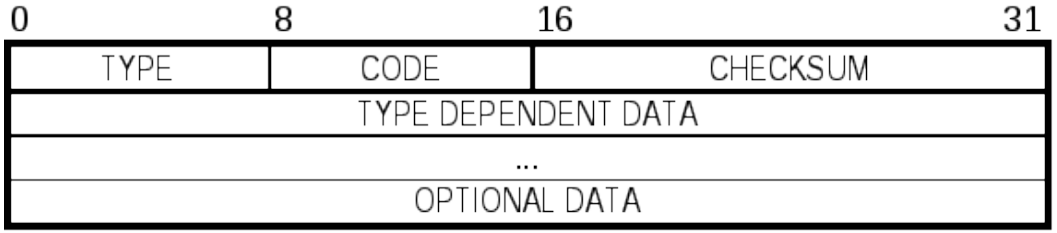
Protokol ICMP

- ICMP (= *Internet Control Message Protocol*)
  - Nejjednodušší protokol pro řízení sítě a předávání chybových hlášení



Zapouzdření ICMP v IP datagramu na fyzické síti

  - Hlavička ICMP datagramu nemá (kromě prvních 4 bytů) pevnou strukturu



  - Pole TYPE udává účel ICMP zprávy a určuje i formát a význam dalších polí – některé typy ICMP datagramů:
    - Standardizovaných typů je mnohem více (cca 40)

TYPE	Účel	TYPE	Účel
0	Echo reply	9	Router advertisement
3	Destination unreachable	10	Router discovery
5	Redirect (route change)	11	Datagram TTL exceeded
8	Echo request	12	Datagram parameter problem

Protokol ICMP – základní užití

- Operátorské použití
  - "Utilita" ping k testování dostupnosti cílového stroje je postavena na ICMP
    - "Náš" systém vyšle ICMP "Echo request" s cílovou adresou testovaného stroje. Navíc ping umí nastavit velikost zasílaného paketu a další příznaky v záhlaví datagramu (např. "don't fragment").
    - Dorazí-li ICMP datagram k cílovému stroji, ten odpoví pomocí ICMP "Echo reply", a když tento paket dorazí "k nám", víme, že cesta je OK.
  - "Utilita" traceroute (ve Windows tracert) dovolí trasovat cestu od "našeho" stroje k cíli
    - Využívá fakt, že každý směrovač po cestě datagramu dekrementuje pole TTL, a klesne-li hodnota tohoto pole na nulu, informuje zdrojový systém ICMP zprávou "Datagram TTL exceeded" (typ 11).
    - Posíláme tedy sérii datagramů ICMP "Echo request", kde první datagram má pole TTL=1, druhý TTL=2, atd. Tím se nám vrací datagramy ICMP type 11 od všech směrovačů po cestě "od nás" k cíli. Dosažení cíle je indikováno návratem ICMP "Echo reply".
    - Existují varianty traceroute užívající i jiných protokolů, ale princip s proměnným TTL je týž.
- ICMP se užívá i pro zjištění lokálního směrovače
    - Stroj na lokální síti vyšle ICMP 10 (Router discovery) s cílovou adresou 0.0.0.0 (*broadcast*) a směrovač odpoví ICMP 9 (Router advertisement)

IPv6

- IPv6 má adresu danou 128 bity (IPv4 pouze 32 bit)
- IPv6 vylepšuje některé vlastnosti IPv4, ale stará se pouze o vrstvu síťování
- Proč 6? Internet Stream Protocol z roku 1979 používá IP hlavičku s číslem verze 5. Další volné číslo je 6.
- Vylepšení
  - Velký prostor adres – přibližně 3.4×10^{38} různých adres
  - Multicasting – vyslání jednoho paketu na více různých počítačů
  - SLAAC – Stateless address autoconfiguration – automatická konfigurace za použití algoritmu prohledávání okolí Neighbor Discovery (lze použít i DHCPv6 nebo statické nastavení adresy)
  - Použití síťové bezpečnosti (šifrování a ověřování) je povinné v IPv6
  - Mobilita – zabraňuje různému směrování paketů při cestě ze zařízení a zpět (avoid triangular routing)
  - Jumbogram – datagram s velikostí až $2^{32} - 1$ (v IPv4 max 65535)

Adresy IPv6

Zápis 8 skupin s 16 bity – každá skupina má 4 hexadecimální čísla

- 2001:0db8:85a3:0000:0000:8a2e:0370:7334
- 0 nejsou důležité, pokud nemají žádný význam
- 2001:db8:85a3::8a2e:370:7334

Formát adresy:

- 48 nebo více – routing prefix - směrování
- 16 nebo méně – subnet-id (subnet-id+routing prefix = 64 bitů)
- 64 – identifikace zařízení (může to být MAC-adresa síťové karty, identifikátor od DHCPv6, náhodně vygenerované číslo, ručně zadané číslo)

Multicast – datagram pro více počítačů:

- 8 bitů prefix – začínající FF
- 4 bity příznaků
- 4 bity rozsah – scope 16 předefinovaných rozsahů pro multicast
- 112 bitů číslo skupiny

IPv6 směrování

- Zjednodušené zpracování pro směrovače (routery)
- Hlavička IPv6 je jednodušší
- IPv6 směrovače neprovádějí fragmentaci
  - Minimální MTU (Maximal transmission unit) je 1280
  - Směrovače umožňují detekci MTU na specifikované cestě
- IPv6 nemá kontrolní součet (to zajišťuje transportní a linková vrstva)
- TTL – (Time To Live) je nahrazen Hop Limit – maximální počet směrovačů na cestě, není třeba měřit čas strávený v bufferech
- Směrovací prefix obsahuje veškeré informace potřebné k směrování
  - RFC 3177 (2001) navrhuje směrování všech počítačů na základě /48 lokace
  - RFC 6177 (2011) zredukovalo lokaci na /56

Od IPv4 k IPv6

- IPv4 s CIDR směrováním a použití NAT zpomalily nutnost přechodu na IPv6
- Jak přejít od IPv4 k IPv6
  - Dual-stack – asi nejčastější současné řešení, směrovač podporuje současně obě verze IP protokolu
  - Tunneling – zapouzdření telegramů IPv6 do IPv4
  - Použití proxy a překlad – pro počítače s IPv6, který chce využít služeb IPv4 serveru je nutné zajistit převedení a překlad IPv6 na IPv4
- Zkontrolujte si své připojení na:
  - www.test-ipv6.cz
  - www.test-ipv6.com
  - ipv6test.google.com

Transportní vrstva

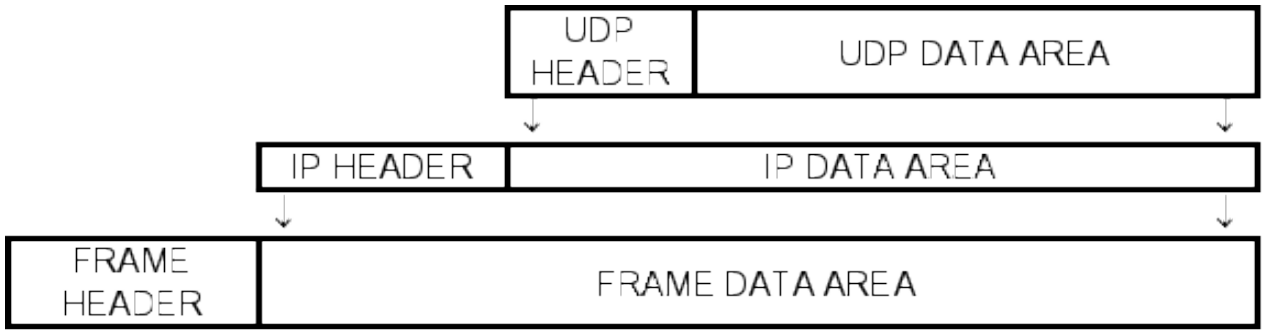
- Cíle transportní vrstvy
  - Zajistit komunikaci mezi procesy
  - Rozlišit různé adresáty na jednom počítači
  - Zajistit spojovaný přenos dat
  - Zvýšit spolehlivost
  - Zvýšit kvalitu služby (QoS Quality of Service)
  - Kontrolovat přenos dat
- Rozlišují se 3 typy sítí
  - Kategorie A – sítě bez ztrát paketů a bez chyb spojení – lokální sítě
  - Kategorie B – sítě bez ztrát paketů s možností chyb spojení – privátní sítě
  - Kategorie C – sítě s možností ztrát paketů i chyb spojení - internet

Transportní vrstva

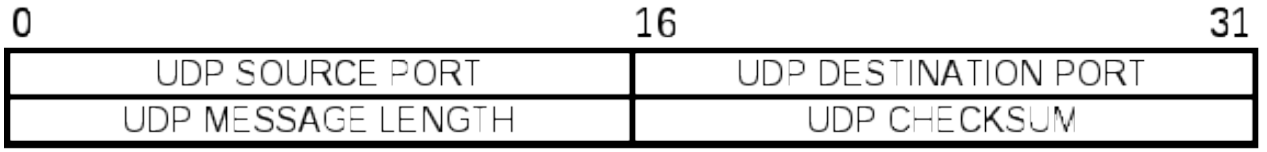
- 5 tříd transportní vrstvy
  - TP0 – jednoduchá vrstva pro sítě kategorie A
  - TP1 – vrstva řešící rozpojení pro sítě kategorie B
  - TP2 – vrstva pro sítě kategorie A s použitím portů
  - TP3 – vrstva řešící rozpojení pro sítě kategorie B s použitím portů
  - TP4 – transportní vrstva pro sítě kategorie C s použitím portů a spolehlivého doručení dat s potvrzováním
- Příklady
  - UDP – transportní vrstva třídy TP2
  - TCP – transportní vrstva třídy TP4

Protokol UDP

- UDP (= *User Datagram Protocol*)
  - jeden z nejjednodušších transportních protokolů.
  - Poskytuje tzv. nespojovanou a nezabezpečenou službu doručování uživatelských datagramů
  - Oproti ryzím IP datagramům má schopnost rozlišit mezi různými cílovými procesy na adresovaném počítači pomocí položky port



Zapouzdření UDP v IP datagramu na fyzické síti



UDP Header

  - Položky PORT se používají k rozlišení výpočetních procesů čekajících na cílovém stroji na UDP datagramy.
    - Položka SOURCE PORT je nepovinná; není-li použita, musí být 0.
    - Jinak označuje číslo portu, na nějž má být zaslána případná odpověď.

Protokol UDP

- Aby se zajistilo, že různé stroje na Internetu si budou rozumět, IANA vydává závazný seznam tzv. obecně známých čísel portů
- Některá vybraná čísla UDP portů:
  (viz http://www.iana.org/assignments/port-numbers)

Port	Keyword	Význam
0	-	Reserved
7	echo	Echo the datagram
13	daytime	Time of the day
53	dns	Domain Name Service
67	bootps	Bootstrap Protocol Server, DHCP Server
68	bootpc	Bootstrap Protocol Client, DHCP Client
69	tftp	Trivial File Transfer
123	ntp	Network Time Protocol
137	netbios-ns	NETBIOS Name Service

Protokol UDP

- UDP protokol nezabezpečuje, že:
  - datagram se během přenosu neztratí
  - datagram nebude doručen vícekrát
- Potřebné zabezpečení musí řešit aplikace, které UDP používají
- Příklad použití UDP
  - DNS (překlad mezi symbolickými jmény strojů a jejich IP adresami) – realizuje komponenta lokálního OS zvaná *resolver*
  - Utilita **nslookup** slouží k explicitnímu použití (a testování) DNS

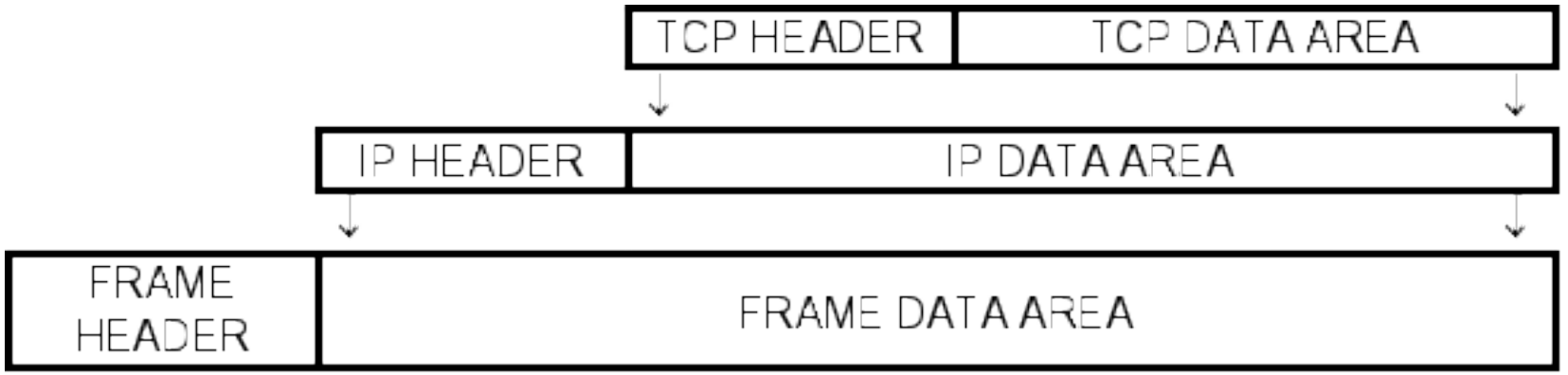
```
zubrina > nslookup proxy.felk.cvut.cz
IP(udp) 147.32.85.46:41245 > 147.32.80.9:53  11033- A? proxy.felk.cvut.cz. (36)
IP(udp) 147.32.80.9:53 > 147.32.85.46:41245  11033* 1/2/3 A 147.32.80.13 (146)

zubrina> nslookup 147.32.80.13
IP(udp) 147.32.85.46:38523 > 147.32.80.9:53  2- PTR? 13.80.32.147.in-addr.arpa. (43)
IP(udp) 147.32.80.9:53 > 147.32.85.46:38523  2* 1/2/4 PTR proxy.felk.cvut.cz. (197)
```

Protokol zabezpečeného datového toku TCP

- TCP je nejdůležitější obecná zabezpečená služba realizující přímé spojení mezi dvěma počítači
  - TCP/IP je Internetová implementace této služby
- Vlastnosti TCP
  - Datový tok
    - Aplikace komunikující po TCP/IP spoji považují komunikační kanál za tok bytů (oktetů) podobně jako soubor
  - Virtuální spoj
    - Před začátkem přenosu dat se komunikující aplikace musí dohodnout na spojení prostřednictvím síťových komponent svých operačních systémů
    - Protokolový software v operačních systémech obou počítačů se dohodne zasíláním zpráv po síti a ověří, že spojení lze spolehlivě navázat a že oba koncové systémy jsou připraveny ke komunikaci
    - Poté jsou aplikace informovány o ustaveném spojení a datová komunikace může být zahájena
    - Přeruší-li se komunikace spojení během, obě strany jsou o tom informovány
    - Termín virtuální spoj je používán k vyjádření iluze, že aplikace jsou propojeny vyhrazeným spojem. Spolehlivosti je dosaženo plně vázanou komunikací po spoji (úplný "*handshake*")

Protokol zabezpečeného datového toku TCP

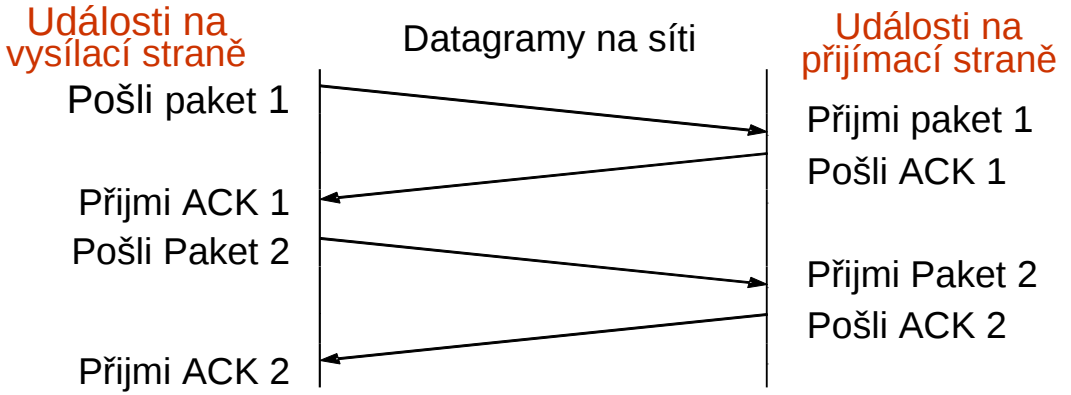
- Přenos s vyrovnávací pamětí
  - Pro zlepšení efektivity přenosu skládá protokolový modul v OS data tak, aby se po síti posílaly pakety rozumné velikosti. Pokud to není žádoucí (např. TELNET), je TCP/IP vybaveno mechanismem, který vynutí přednostní přenos i velmi krátkého datagramu „mimo pořadí“
- Plně duplexní spojení
  - Aplikační procesy vidí TCP/IP spojení jako dva nezávislé datové toky běžící v opačných směrech bez zjevné interakce. Protokolový software potvrzuje (ACK) data běžící v jednom směru v paketech posílaných spolu s daty ve směru opačném



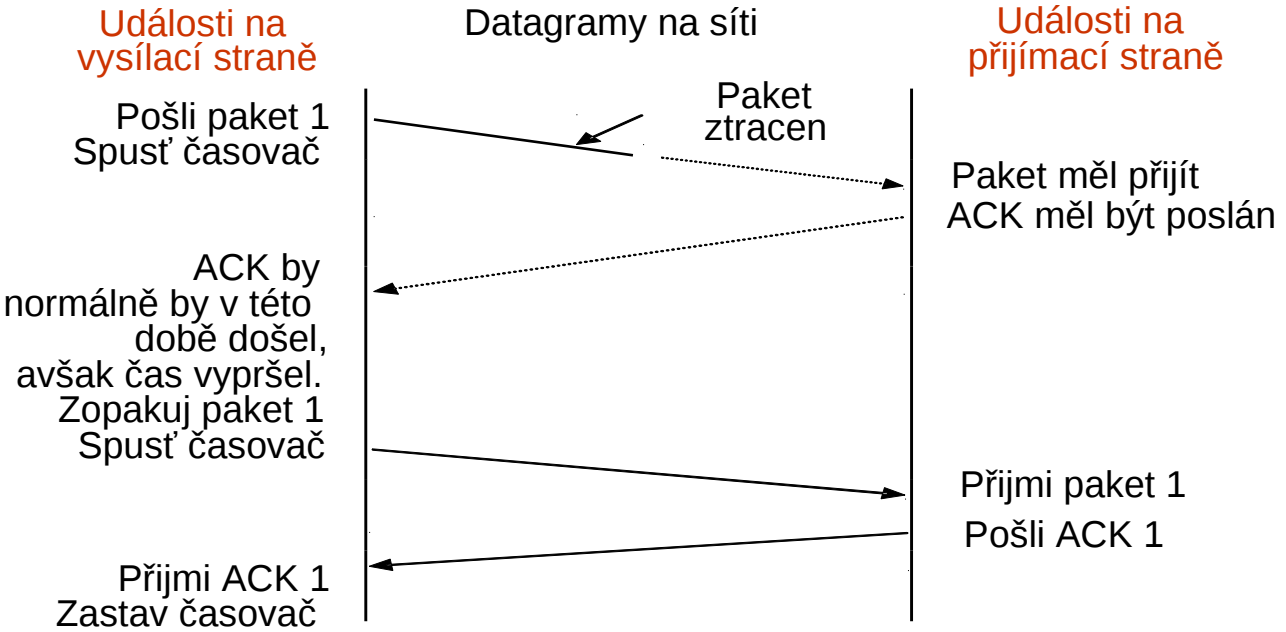
Zapouzdření TCP/IP v IP datagramu na fyzické síti

Řešení spolehlivosti TCP

- Zajištění spolehlivého přenosu
  - pozitivní potvrzování došlých dat spolu s opakováním přenosu

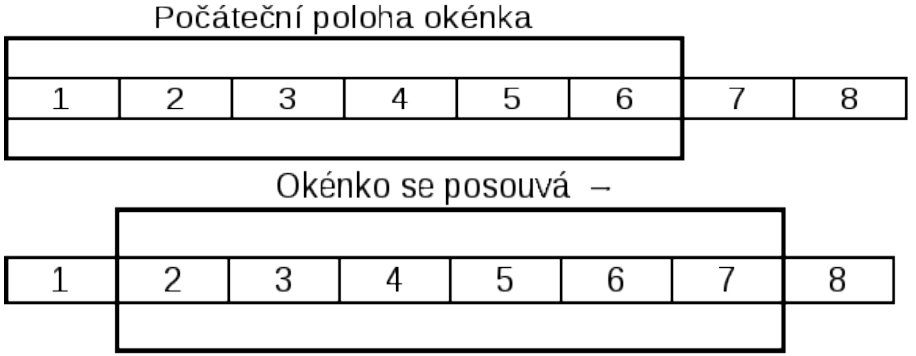


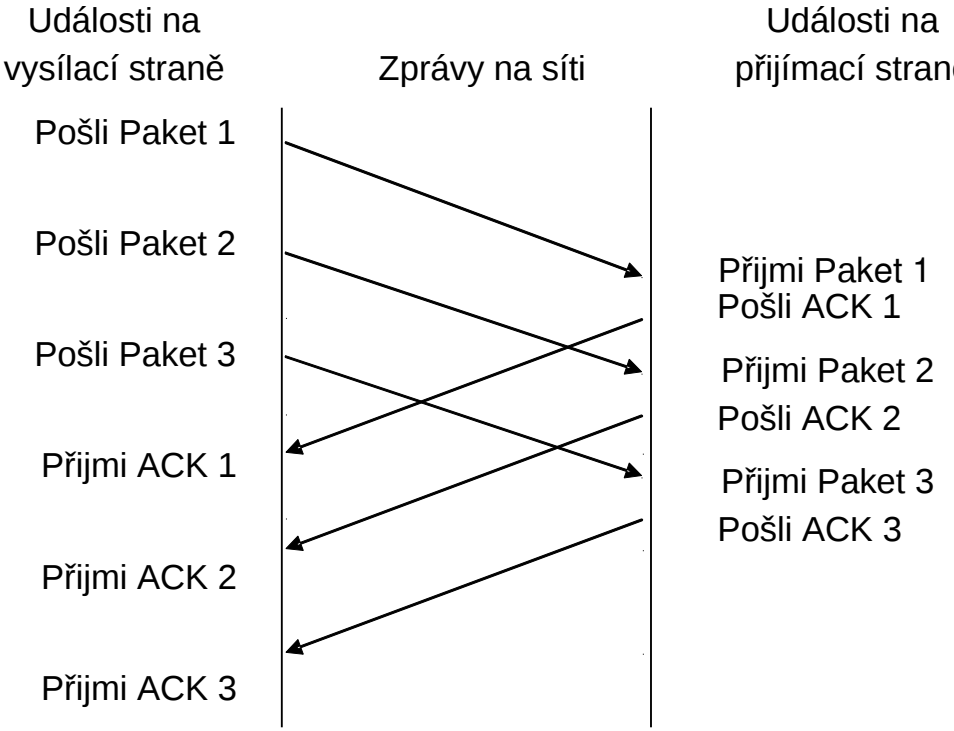
  - Ztracené pakety budou zopakovány na základě vhodných časových prodlev



Řešení spolehlivosti TCP

- Datagramy se mohou po cestě i duplikovat (data i ACK).
  - Tento problém se řeší pomocí sekvenčního číslování datagramů
- Problém efektivity pozitivního potvrzování
  - Čekání na potvrzení každého paketu je časově nákladné
  - Metoda posouvajícího se okna



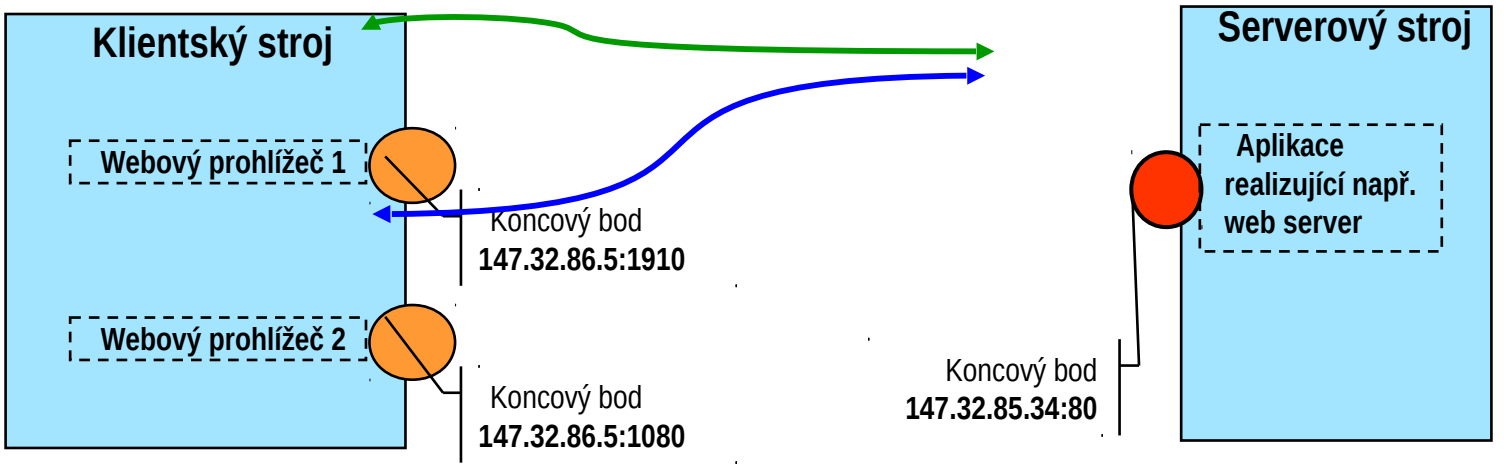


Porty, spojení a koncové body TCP

- TCP rovněž používá porty k rozlišení cílové aplikace na spojených počítačích
  - Čísla portů pro TCP mohou být stejná jako pro UDP, neboť protokoly jsou rozlišeny na obou koncích spoje automaticky
    - Stroj přijímající datagram se napřed "podívá" na pole PROTOCOL v hlavičce datagramu a podle něj předá zpracování buď UDP nebo TCP "větvi" v síťové komponentě OS
  - Aby bylo možno využívat téže služby počítače (serveru) větším počtem jiných počítačů (klientů), TCP/IP zavádí tzv. virtuální spojení (virtuální kanály).
    - Tyto virtuální kanály jsou vlastně spojení mezi tzv. koncovými body, což jsou IP adresy s připojeným číslem portu, např. 147.32.85.34:80.
    - TCP/IP virtuální spojení je pak identifikováno dvěma koncovými body tohoto spojení
      - v IP v4 je to vlastně 12 bytů – tj. 2 x (4 byty adresy + 2 byty port)
    - Pár koncových bodů "celosvětově" odlišuje existující TCP/IP spoj
    - Různé spoje mohou mít na jednom konci týž "koncový bod", avšak na druhém konci musí být různé koncové body.

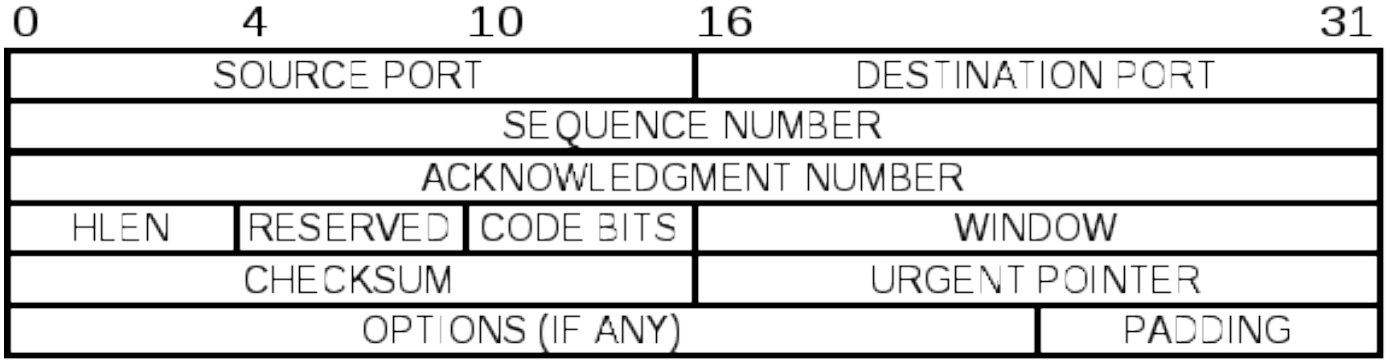
Tvorba TCP spojení

- Pasivní a aktivní otevření
  - TCP vyžaduje, aby se systémy, mezi nimiž se spojení navazuje, předem dohodly o vzniku spojení
  - Aplikace na jednom konci musí požádat svůj lokání OS a uskutečnit tzv. pasivní otevření na daném portu indikující ochotu aplikace přijímat příchozí žádosti o spojení. Tento konec kanálu je obvykle označován jako server.
    - Server "poslouchá" na daném portu
  - Chce-li klientská aplikace se serverem navázat spojení, požádá svůj OS o aktivní otevření, kdy zadá IP adresu serveru a příslušný port. Lokální klientský OS přiřadí navazovanému spojení vhodný volný lokální port (obvykle 1024 – 2047). Oba stroje pak naváží spojení (➔) a mohou spolu komunikovat.



TCP segmenty a jejich formát

- Datový tok TCP se dělí na segmenty
  - Segmenty putují po síti jako IP datagramy
    - Každý byte v datovém toku má své 32-bitové sekvenční číslo v rámci spojení
- Hlavička TCP datagramu (segmentu)



- Význam položek
  - SOURCE PORT, DESTINATION PORT: Identifikace aplikací na obou koncích spojení
  - SEQUENCE NUMBER: Sekvenční číslo bytu v datovém toku
  - ACKNOWLEDGMENT NUMBER: sekvenční číslo bytu v protisměrném toku, který odesilatel očekává v odpovědi od příjemce

    Poznámka: SEQUENCE NUMBER se vztahuje ke směru přenosu, v němž se posílá segment, zatímco ACKNOWLEDGMENT NUMBER se vztahuje ke směru opačnému

TCP segmenty a jejich formát

- Význam položek TCP hlavičky (pokračování)
  - HLEN: Délka hlavičky ve 32-bitových slovech
  - CODE: Pole obsahující 1-bitové příznaky:
    - URG: Pole URGENT POINTER je platné ➔
    - ACK: Datagram nese potvrzení protisměrného datového segmentu
    - PSH: Tento segment požaduje "push", tj. okamžité doručení aplikaci bez použití vyrovnávací paměti na přijímající straně
    - RST: Reset spojení
    - SYN: Aktivní žádost o zřízení spojení (synchronizace sekvenčních čísel)
    - FIN: Ukončení spojení (odesilatel detekoval konec datového toku)
  - WINDOW: Určuje kolik dat je odesilatel ochoten přijmout od příjemce v rámci datového toku běžícího v opačném směru
  - URGENT POINTER: Toto pole je ukazatel na urgentní datový element uvnitř datového úseku segmentu (např. ^C v TELNET-ovém spojení) – platí jen ve spojení s příznakem URG
  - OPTIONS: Volitelné položky používané při vytváření spojení (např. max. velikost segmentu)

Časové prodlevy pro opakování přenosů

- Konstantní hodnota časového zpoždění pro opakované vyslání paketu je nevhodná
  - Internet je příliš různorodý a je složen z mnoha různých LAN a „point-to-point“ spojů založených na různých HW technologiích
- TCP/IP přizpůsobuje časové parametry virtuálního spoje
  - Používá adaptivní algoritmus pro zopakování posílaného paketu.
  - Algoritmus je založen na průběžném sledování tzv. „*round trip time*“ (RTT)
    - Doba mezi odesláním paketu a přijetím jeho potvrzení.
  - Skutečná prodleva pro opakování paketu je určována jako vážený průměr z RTT naměřených v nedávné historii.
  - Strategie se rychle přizpůsobuje okamžité zátěži mezilehlých sítí a směrovačů

Navázání TCP spojení

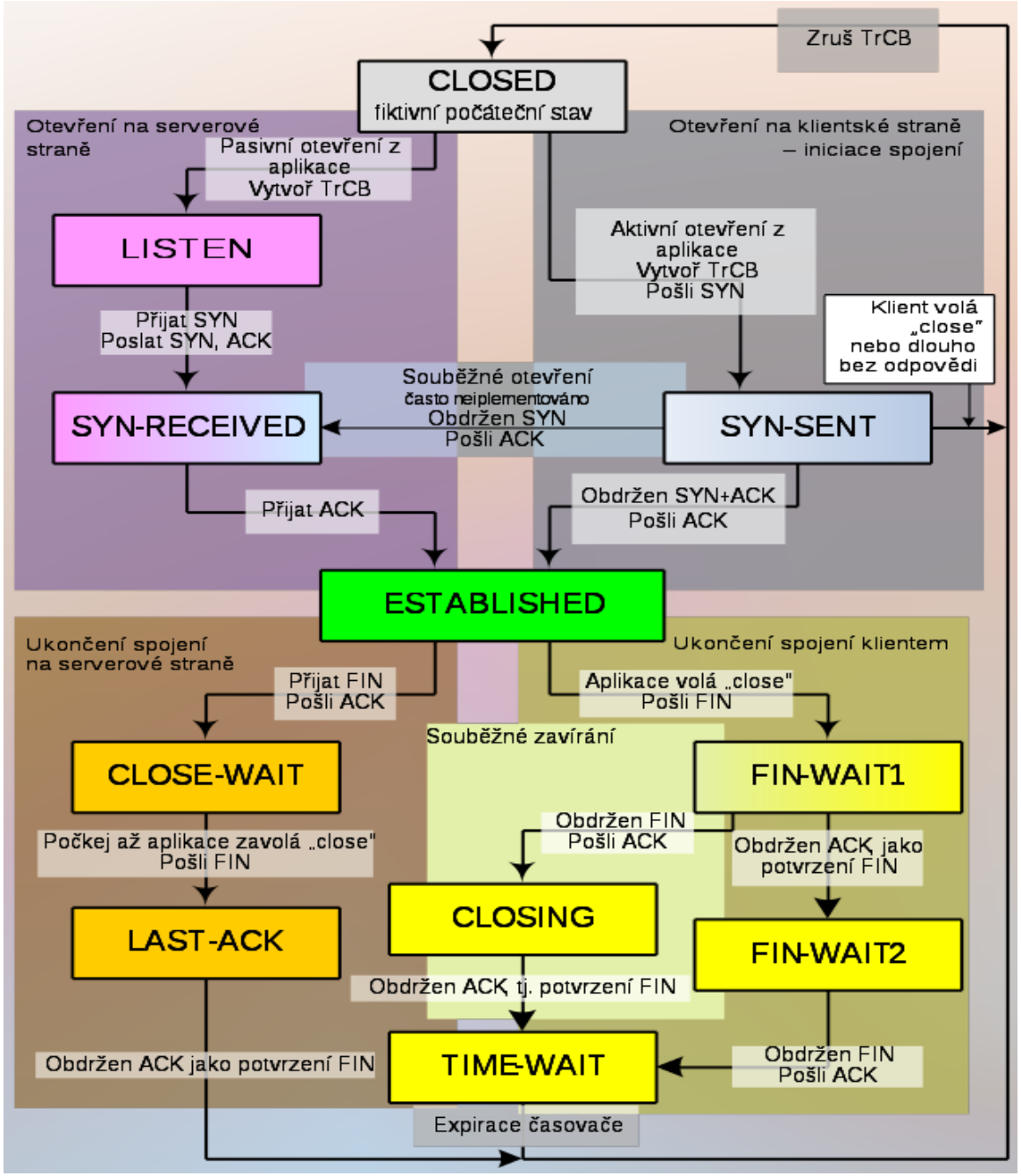
- TCP používá třístupňový postup navazování spojení (předpokládáme, že server „naslouchá“):
  1. V prvním kroku iniciátor spojení (klient) pošle adresátovi (serveru) segment s nastaveným SYN bitem, náhodně vygenerovaným SEQUENCE NUMBER = *x* a prázdnou datovou sekcí
  2. Když server obdrží tento segment, odpoví na něj segmentem s nastaveným SYN a ACK, náhodným SEQUENCE NUMBER = *y* a ACKNOWLEDGMENT NUMBER = *x*+1.
  3. Když klient dostane tento segment, potvrdí jeho přijetí zasláním segmentu s nastaveným ACK, nulovým SYN bitem a ACKNOWLEDGMENT NUMBER = *y*+1.
  - Tak jsou ustaveny počáteční hodnoty SEQUENCE NUMBER a ACKNOWLEDGMENT NUMBER pro dané spojení
  - Sekvenční čísla jsou náhodná
    - Možnost detekovat havárii či restartování strojů na koncích spoje v rámci časové prodlevy

Ukončení TCP spojení, TCP porty

- Ukončení spojení nastává obvykle na žádost aplikace, která spojení ustavila
  - Aplikace sdělí TCP, že už nemá další data
  - TCP software uzavře spojení v jednom směru, což se děje zasláním segmentu s nastaveným FIN bitem
  - K úplnému ukončení je třeba spojení zavřít i v opačném směru podobným způsobem (tj. FIN bitem)
  - TCP spojení lze i okamžitě násilně přerušit užitím bitu RST
- Příklady obecně známých čísel TCP portů

Port	Keyword	Význam
20	ftp-data	File Transfer Protocol (data section)
21	ftp	File Transfer Protocol (controls)
22	ssh	Secure Terminal Connection
23	telnet	Terminal Connection
25	smtp	Simple Mail Transport Protocol
53	domain	Domain Name Server Transfer
80	http	World-wide web
110	pop3	Post Office Protocol v.3
143	imap	Internet Message Access Protocol
443	https	Secured www

Konečný automat TCP



Uveden je zjednodušený konečný automat

- předepisuje chování síťové vrstvy OS pro TCP spojení
- jde o „popis implementace“
- každé samostatné spojení může být v daném okamžiku v jiném vývojovém stavu

Každé TCP spojení má svůj řídicí blok TrCB (*Transmission Control Block*)

- je propojen s příslušným socketem
- registruje průběh navazování spojení a jeho "stav"
- odkazuje na vyrovnávací paměti pro přenos dat

API pro síťové služby

- Základním prostředkem pro síťové komunikace je tzv. socket
  - obecný objekt pro meziprocesní komunikaci (IPC)
  - nejčastěji však pro IPC prostřednictvím počítačových sítí
    - tzv. rodina POSIX socketů – zprostředkují IP komunikaci
    - POSIX sockety se vyvinuly z původních BSD socketů
  - z pohledu API se socket jeví jako POSIX „soubor“
- Vytvoření socketu (získání manipulačního čísla „souboru“)

  int sock_fd = **socket**(int domain, int type, int protocol)
  - domain – specifikuje rodinu socketu (pro IP domain = AF_INET)
  - type – určuje způsob komunikace zprostředkované socketem
    - SOCK_STREAM = socket zprostředkuje datový tok (nejčastěji TCP)
    - SOCK_DGRAM = socket zprostředkuje předávání datagramů (např. UDP)
    - SOCK_RAW = socket umožňuje přímý přístup k síťovým službám (užívá se např. pro přístup aplikací k ICMP)
  - protocol – konkrétní protokol (TCP, UDP, ...)
    - IPPROTO_IP = protokol je automaticky zvolen podle parametru type
    - IPPROTO_ICMP = socket zprostředkuje ICMP protokol
    - IPPROTO_UDP = UDP přenos datagramů
    - IPPROTO_TCP = TCP datový tok
    - ... další protokoly viz RFC 1700

Operace se sockety

- Navázání socketu na lokální adresu (pasivní otevření na serverové straně)

  bind(int sock_fd, const struct sockaddr *my_addr, socklen_t addr_len)

  - sock_fd – socket
  - my_addr – lokální adresa, pro AF_INET struktura včetně portu
  - addr_len – délka adresy v bytech
- Čekání socketu na žádost o příchozí spojení (na serverové straně)

  listen(int sock_fd, int backlog)

  - sock_fd – socket
  - backlog – maximální počet čekajících spojení
- Přijetí žádosti klienta o spojení se serverem (včetně identifikace klienta)

  new_fd = **accept**(int sock_fd, struct sockaddr *client_addr,
  socklen_t *client_addr_len)

  - sock_fd – socket
  - client_addr – adresa klienta, pro AF_INET struktura včetně portu
  - addr_len – délka adresy v bytech
  - new_fd je „souborový deskriptor“, jehož prostřednictvím bude probíhat obousměrná komunikace mezi klientem a serverem

  Původní socket zůstává ve stavu „listen“ a je chopen přijímat další příchozí spojení a řadit je do fronty. Existují-li takové žádosti o spojení, další volání accept vrátí ihned další klientské spojení; v opačném případě accept způsobí zablokování volajícího procesu

Operace se sockety

- Připojení na vzdálenou adresu (aktivní otevření klientem)

  connect(int sock_fd, const struct sockaddr *serv_addr, socklen_t addr_len)

  - sock_fd – socket
  - serv_addr –adresa serveru, k němuž se klient připojuje, pro AF_INET struktura včetně portu
  - addr_len – délka adresy v bytech

- K přenosům dat mezi klientem a serverem poté, kdy příchozí žádost byla akceptována (spojení bylo úspěšně navázáno)

  send(), **recv**()
  sendto(), **recvfrom**()
  write(), **read**()
  sendmsg(), **recvmsg**()

  - Prvním parametrem všech těchto funkcí je sock_fd
  - Detaily viz specifikace POSIX

- Ukončení spojení

  close(int sock_fd)

Použití API pro síťové služby

- Server
  1. Vytvoř socket voláním služby `socket()`
  2. Navaž socket na lokální adresu a port voláním `bind()`
  3. Připrav socket na příchod žádostí o spojení voláním `listen()`, vznikne „naslouchající socket“
  4. Voláním služby `accept()` se server zablokuje, dokud nepřijde žádost o spojení. Návratovou hodnotou `accept()` je nový souborový deskriptor (`fd`), otevřený pro komunikaci. Původní socket stále naslouchá a lze znovu volat `accept()`.
  5. Komunikace pomocí `send()`a `recv()`nebo `write()` a `read()`
  6. Případné volání `close()` končícím serverem (passive close)

- Klient
  1. Vytvoř socket voláním služby `socket()`
  2. Volání služby `connect()` naváže spojení se serverem a vrátí souborový deskriptor pro další komunikaci
  3. Komunikace se serverem pomocí `send()`a `recv()`nebo `write()` a `read()`
  4. Volání `close()` k ukončení spojení se serverem (active close)

Základní aplikační IP protokoly

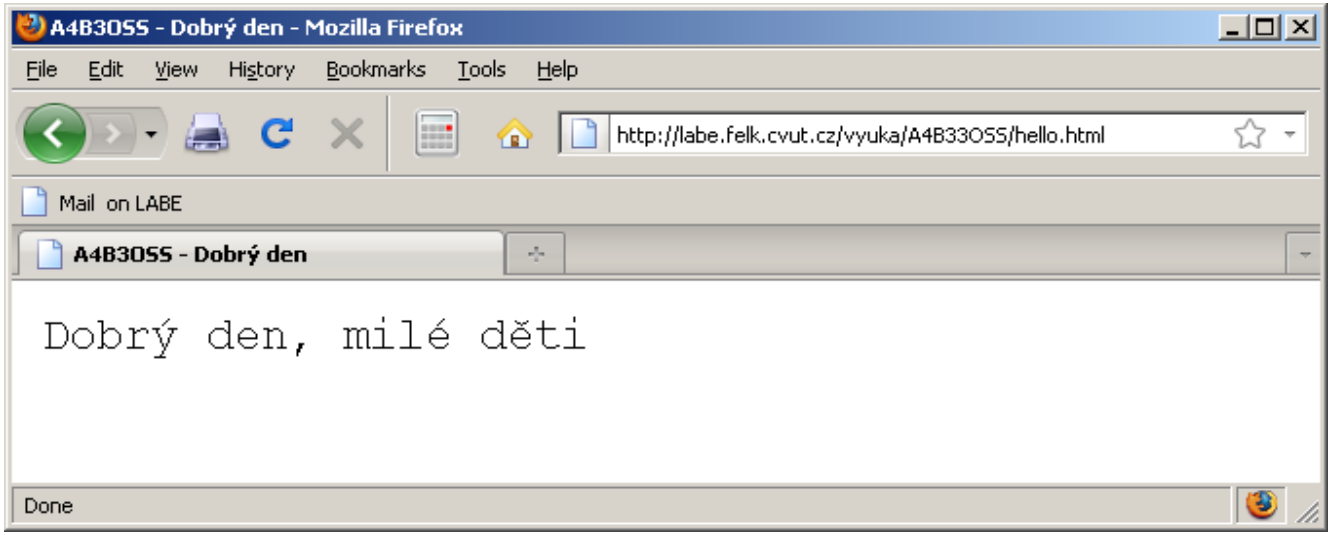
- IP aplikačních protokolů jsou stovky
  - Obvykle se pro uživatelsky orientované protokoly používá zabezpečený transportní protokol TCP/IP
- Většina aplikačních protokolů využívá komunikace „v otevřené řeči“
  - příkazy a reakce na ně jsou „v primitivní angličtině“
  - nebezpečné, proto často existují „zabezpečené“ (zakódované) varianty
- Vyjmenujeme jen pro ukázku některé základní aplikační protokoly
  - SMTP (*Simple Mail Transfer Protocol*), TCP port 25
    - Protokol je určen pro zasílání e-mailů klientem nebo „serverem“ (který se při tomto přenosu chová jako klient) na cílový server.
    - Základní příkazy zadávané klientem jsou MAIL, RCPT, DATA
  - POP3 (*Post Office Protocol*), TCP port 110
    - Protokol pro stahování e-mailů ze serveru, kam byl e-mail doručen pomocí SMTP, do pracovní stanice (např. do aplikace MS-Outlook)
    - Základní příkazy klienta: USER, PASS, LIST, RETR, DELE
  - FTP (*File Transfer Protocol*), TCP porty 20 a 21
    - Velmi komplexní protokol založený na dvou TCP spojích (řídicí a datový); existuje mnoho „klonů“ (např. pasivní FTP) a zabezpečených variant (např. SFTP)
    - Příkazů je asi 50

Protokol HTTP – reálná ukázka

- Web server pro protokol HTTP (*HyperText Transfer Protocol*) poslouchá na TCP portu 80
- Příklad komunikace
  - Na serveru *labe.felk.cvut.cz* je soubor hello.html v adresáři /vyuka/A4B33OSS
    - serverový proces httpd obsluhující TCP port 80 považuje z bezpečnostních důvodů jistý konkrétní adresář na serveru jako kořen adresářového stromu pro „webové soubory“ (dáno konfigurací httpd)
  - Obsah souboru /vyuka/A4B33OSS/hello.html je např.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250">
<meta http-equiv="Pragma" content="no-cache">
<title>A4B33OSS - Dobrý den</title>
</head>
<body>
<pre>
Dobrý den, milé děti
</pre>
</body>
</html>
```

Protokol HTTP – reálná ukázka



- Klient Firefox se připojí na TCP port 80 a pošle

```
GET /vyuka/A4B33OSS/hello.html HTTP/1.1
Host: labe.felk.cvut.cz
User-Agent: Mozilla/5.0 (Windows NT 5.1; en-US;) Firefox/3.25
Accept: text/html, Accept-Language: cs,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: windows-1250,utf-8;q=0.7,*;q=0.7
Keep-Alive: 15
Connection: keep-alive
Cache-Control: max-age=0
<prázdný řádek>
```

Povinné komponenty zaslaného příkazu GET jsou **tučně**.
Další jsou doplňkové informace pro server

Protokol HTTP – reálná ukázka

- Web server odpoví

```
HTTP/1.1 200 OK
Date: Sat, 18 Dec 2012 19:47:10 GMT
Server: Apache/1.3.37 (Unix)
Last-Modified: Sat, 18 Dec 2010 19:40:14 GMT
ETag: "2da442-ea-4d0d0e1e"
Accept-Ranges: bytes
Content-Length: 234
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
```

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=windows-1250">
<title>A4B3OSS - Dobrý den</title>
</head>
<body>
<pre>Dobrý den, milé děti
</pre>
</body>
</html>
```

A po 15 sekundách server ukončí spojení

Obsah hello.html

To je dnes vše.

Otázky?