

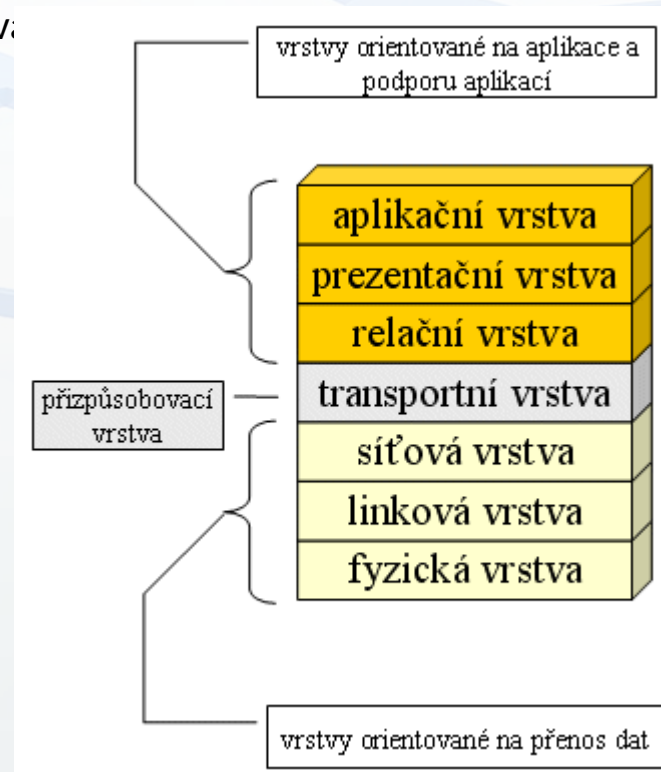
Úvod do počítačových sítí

Přednáška 10
(2021/2022)
ver. 2021-12-07-01



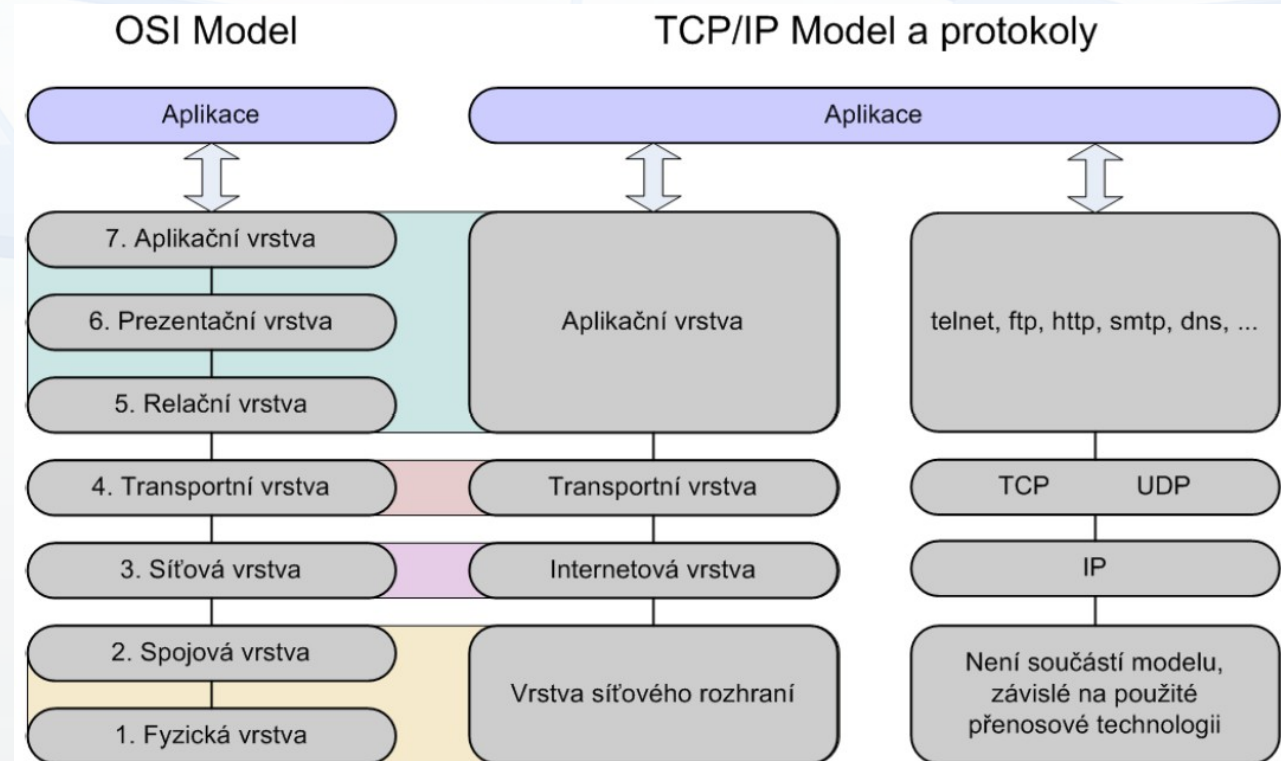
L4 – Transportní vrstva

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



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

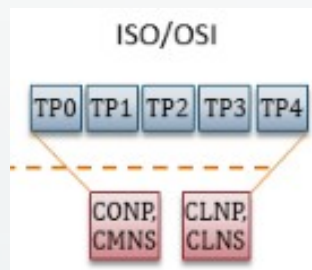
- V obou modelech je transportní vrstva zastoupena a na stejné úrovni - L4
- Provedení L4 se mezi ISO/OSI a TCP/IP výrazně liší
 - Provedením se myslí původní plán / návrh
- Dnes už se reálně používá jen TCP/IP varianta
- ISO/OSI nabízí vyšší vrstvám 5 variant přenosů
 - T0, T1, T2, T3, T4 – transportní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



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

L4 protokoly v ISO/OSI

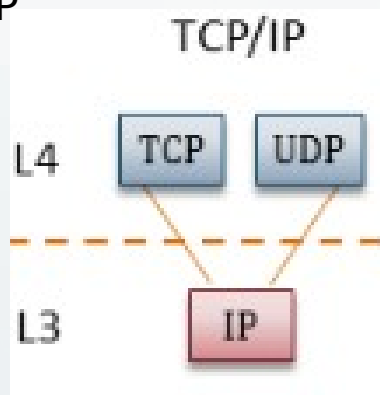
- Už v původním provedení počítala s 5 druhy služeb poskytovaných vyšším vrstvám
- Vychází z původního návrhu L3 ISO/OSI, které může fungovat :
 - Spolehlivě a spojovaně
 - CONP – Connection Oriented Network Protokol
 - Přenosový protokol
 - CMNS – Connection Mode Network Service
 - Služba pro vyšší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



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

L4 protokoly v TCP/IP

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



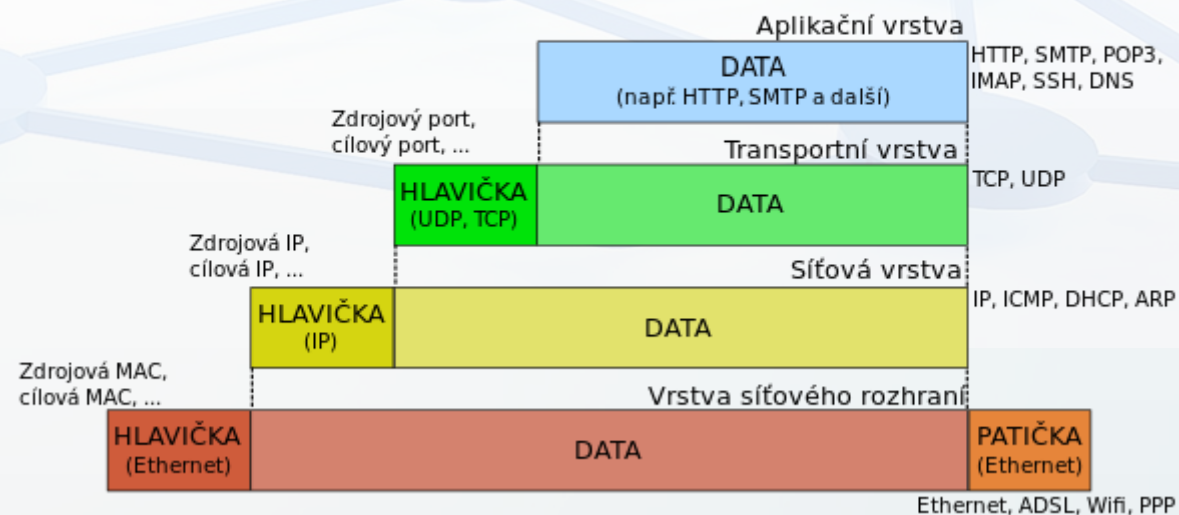
<i>Protokol UDP</i>	<i>Protokol TCP</i>
Mezi počítači není třeba vytvořit relaci.	Mezi počítači se vytváří relace.
Nezaručuje doručení dat, nezajišťuje potvrzování ani správné pořadí dat.	Zaručuje dodání paketů s využitím potvrzování a správného pořadí při přenosu dat.
Programy používající UDP musí zajistit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

Základní funkce transportní vrstvy:

Zapouzdření dat

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

ZAPOUZDŘENÍ DAT V SÍTI TCP/IP



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

Základní funkce transportní vrstvy:

Zapouzdření dat- TCP a UDP hlavičky

TCP Segment Header Format

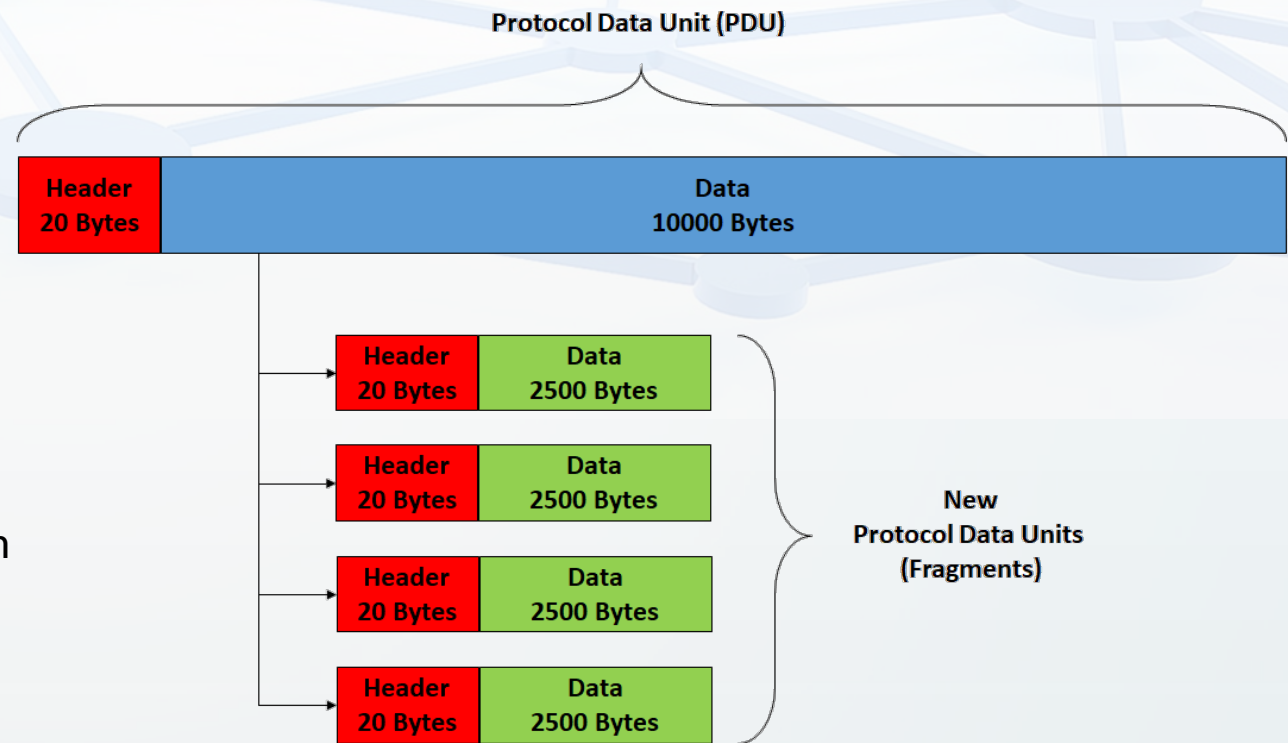
Bit #	0	7	8	15	16	23	24	31
0	Source Port				Destination Port			
32	Sequence Number							
64	Acknowledgment Number							
96	Data Offset	Res	Flags		Window Size			
128	Header and Data Checksum				Urgent Pointer			
160...	Options							

UDP Datagram Header Format

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

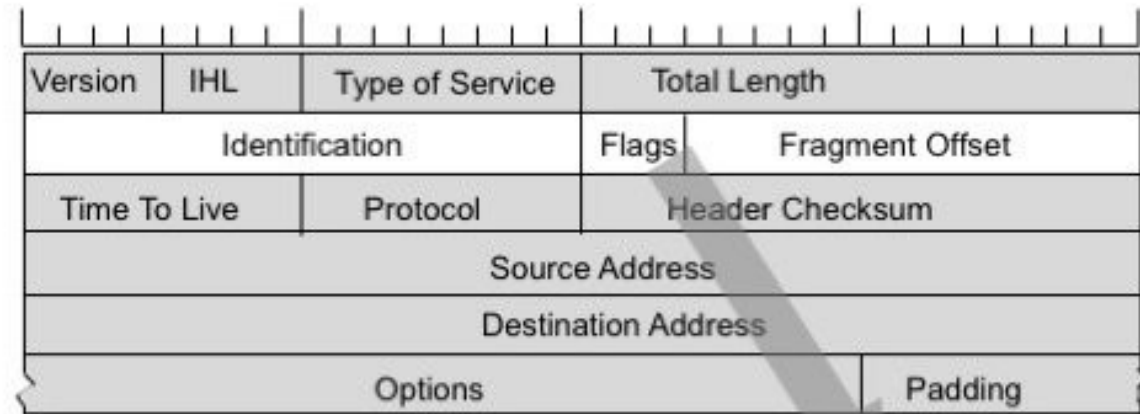
Segmentace a fragementace dat: Základní princip

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



Segmentace a fragementace dat: Povolení a zákaz fragementace

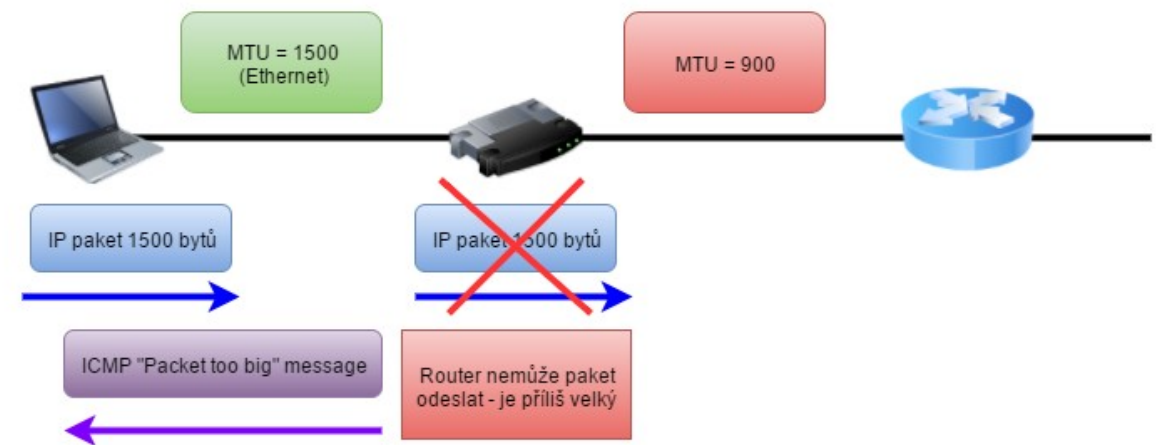
- K fragmentaci dojde, pokud potřebuji větší data vyšší vrstvy přenést v menší jednotce nižší vrstvy
 - MTU je 1500, potřebuji přenést 6000
 - Segment bude přenesen na 4 části po 1500 bytech
- Pokud má k fragmentaci dojít, musí být podporována – především na L3 a L4
- Zda je fragementace povolena či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



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

Segmentace a fragementace dat: PMTUD

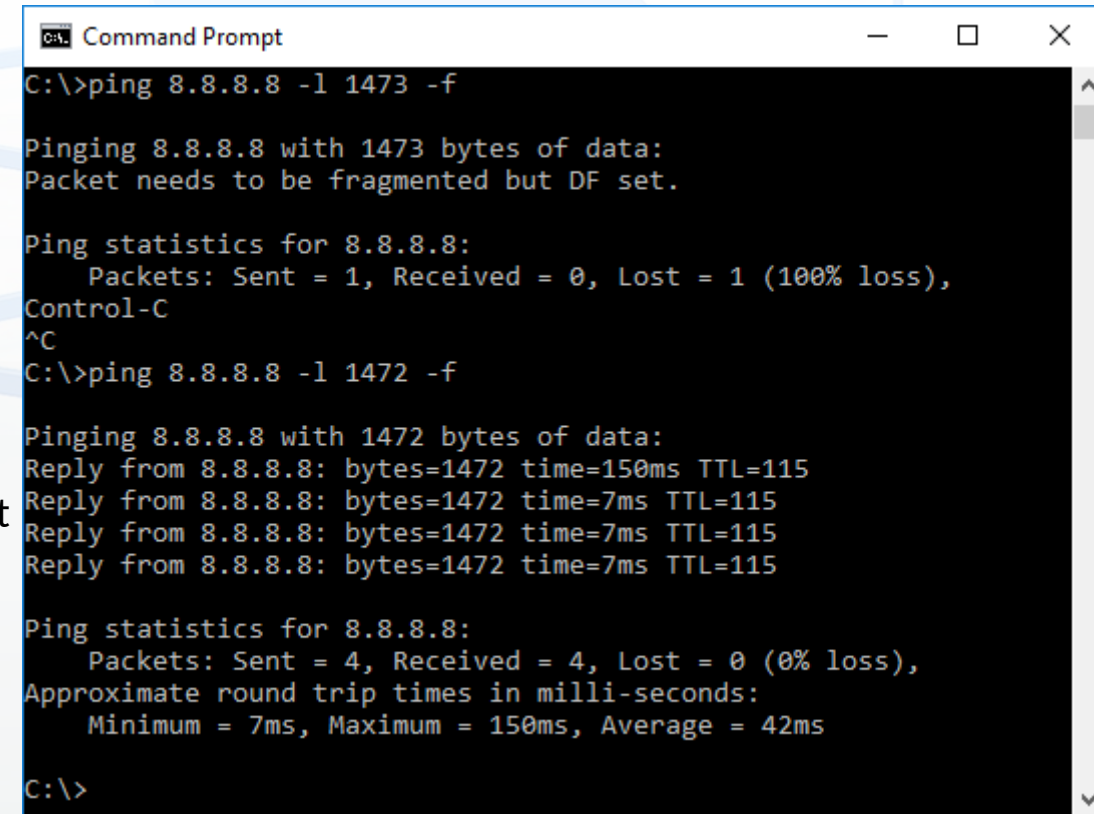
- Na příkladu z obrázku se snažíme poslat data větší než mohou routerem projít
- Zde funguje PMTUD – Path MTU Discovery
 - Snažím se najít optimální velikost dat
 - Na prvním skoku nastavení MTU znám, ale dále nevím
 - Zním, protože zde je přímé spojení
 - POZOR i zde mohou mít obě proti strany různé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



Segmentace a fragementace dat:

Problémy s fragmentací

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



```
Command Prompt
C:\>ping 8.8.8.8 -l 1473 -f

Pinging 8.8.8.8 with 1473 bytes of data:
Packet needs to be fragmented but DF set.

Ping statistics for 8.8.8.8:
    Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),
Control-C
^C
C:\>ping 8.8.8.8 -l 1472 -f

Pinging 8.8.8.8 with 1472 bytes of data:
Reply from 8.8.8.8: bytes=1472 time=150ms TTL=115
Reply from 8.8.8.8: bytes=1472 time=7ms TTL=115
Reply from 8.8.8.8: bytes=1472 time=7ms TTL=115
Reply from 8.8.8.8: bytes=1472 time=7ms TTL=115

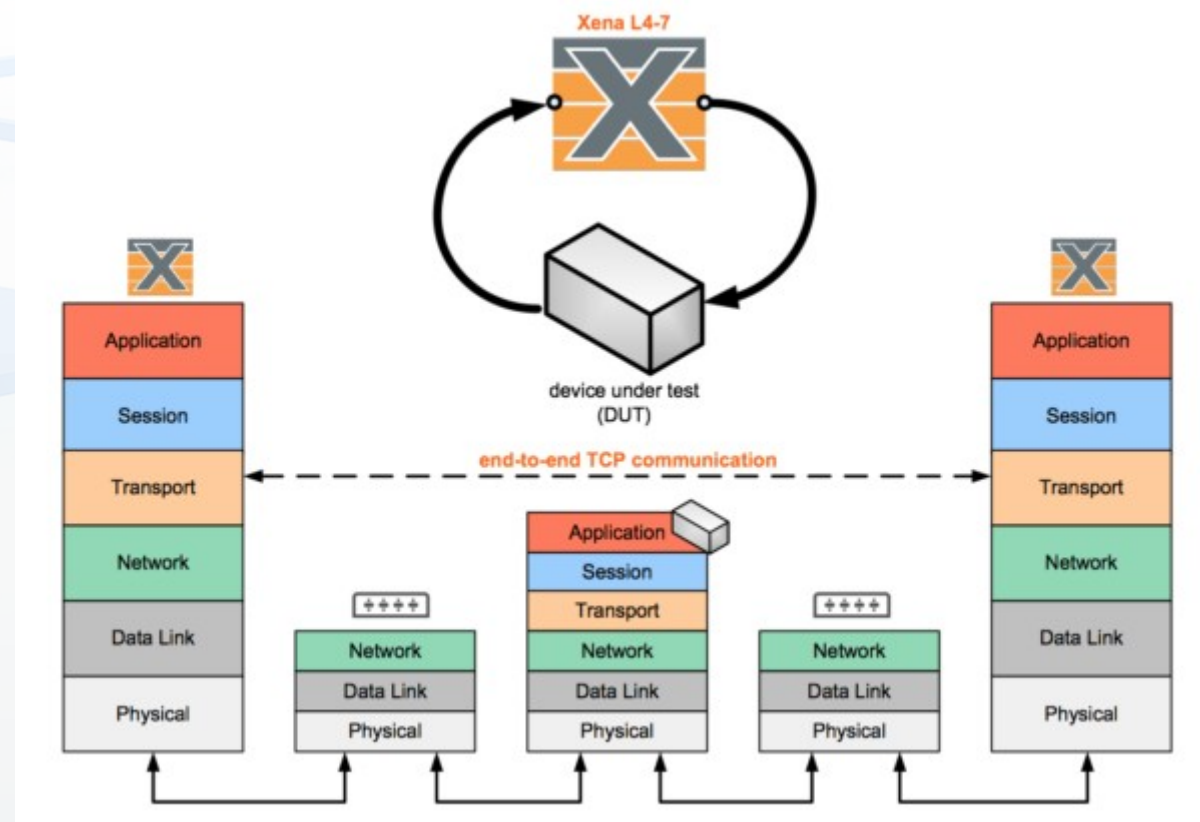
Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 7ms, Maximum = 150ms, Average = 42ms

C:\>
```

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

End-to-end komunikce

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

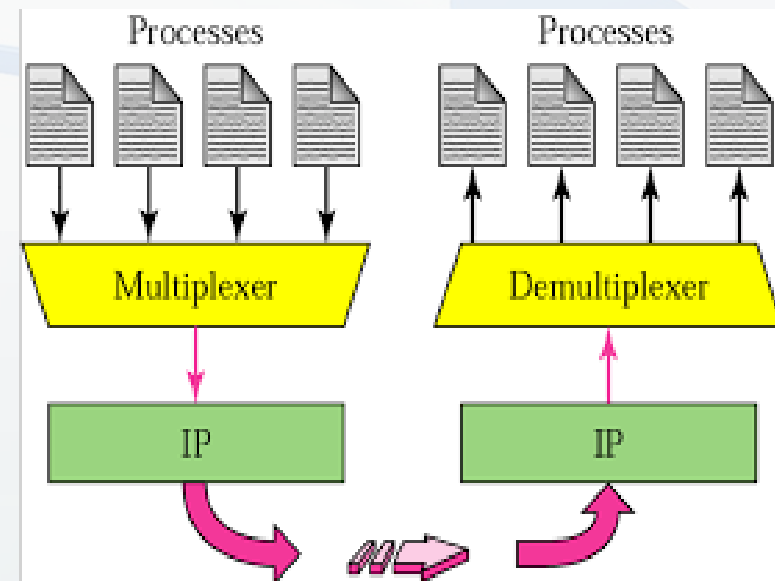


zdroj:

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

Multiplex a demultiplex na L4

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



Multiplex a demultiplex na L4: Naslouchání aplikací

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


Adresace v transportní vrstvě

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

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

Adresace v transportní vrstvě:

Kategorizace portů

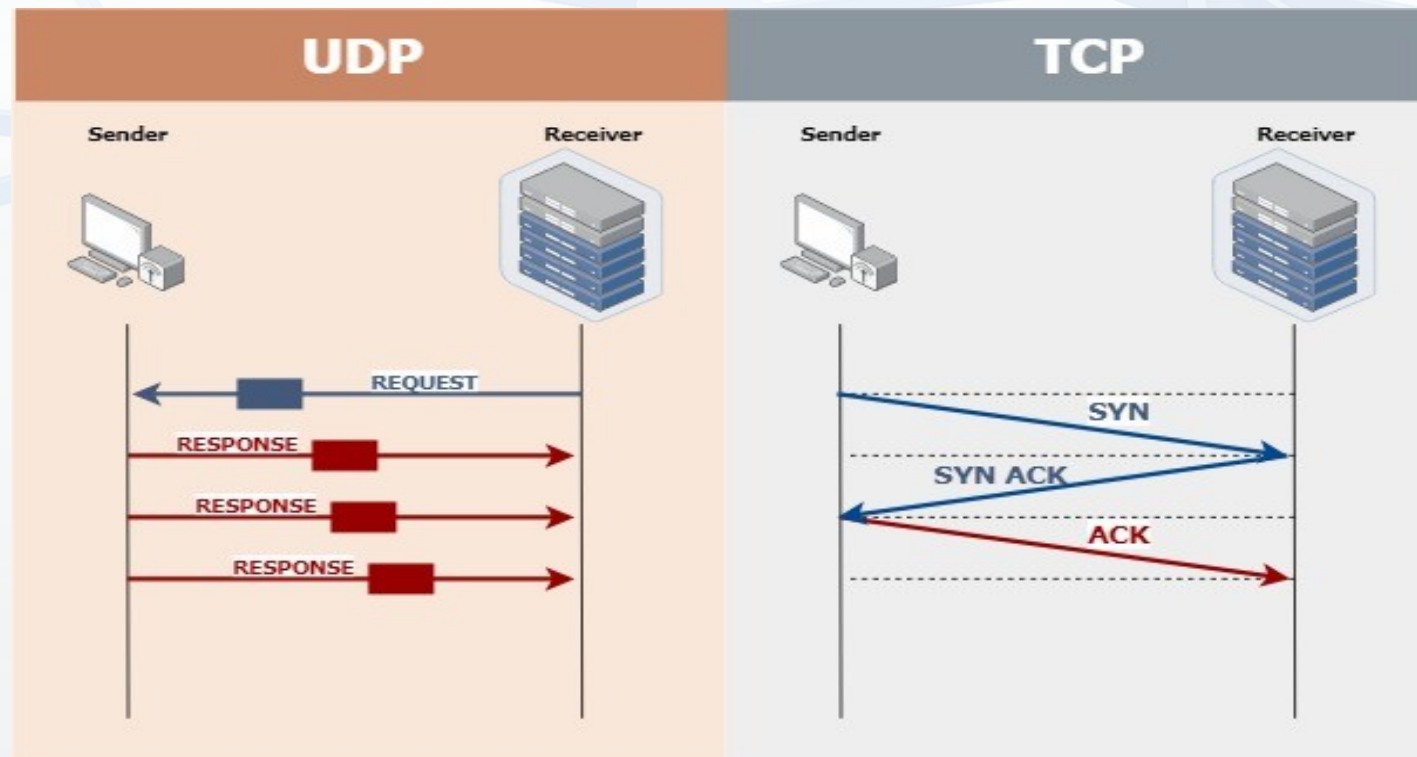
- Obecně může jakýkoliv proces fungovat na jakémkoliv portu – s výjimkou portu 0
- „0“ se chová jinak než ostatní port – v rámci přenosu se nula nahrazuje za jakýkoliv volný z dynamických portů
- Rozlišujeme tři skupiny portů
 - Dobře známé porty 0 – 1023
 - Na těchto portech běží typicky vždy stejné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

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

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

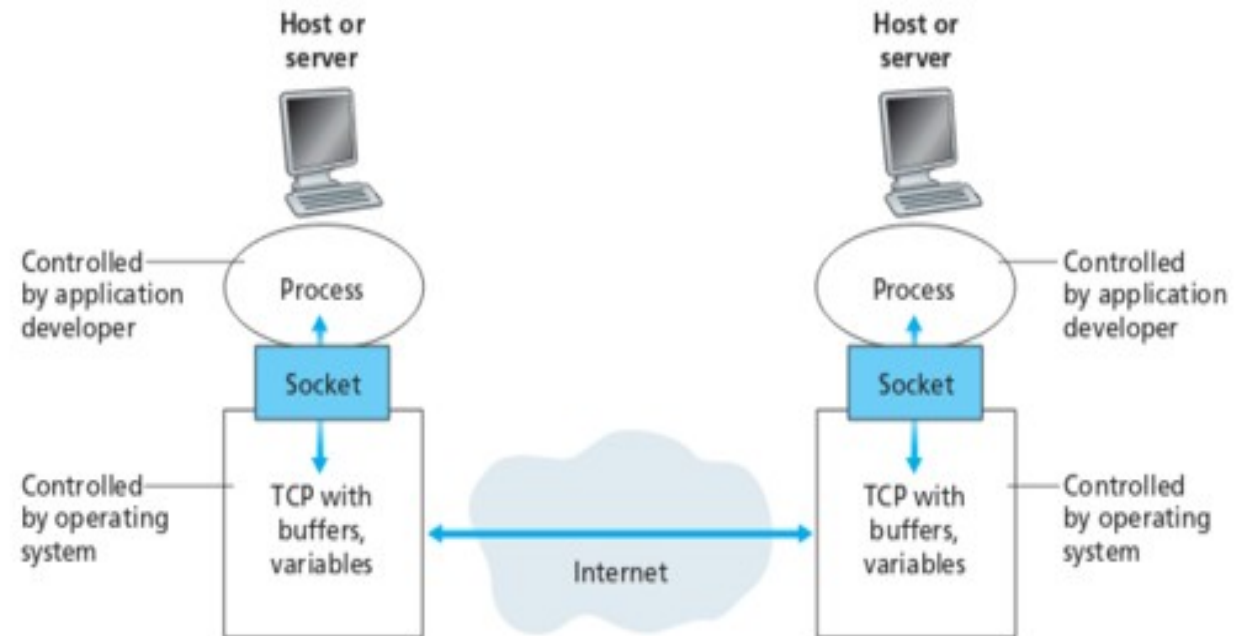
Protokoly v L4 pro TCP/IP

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



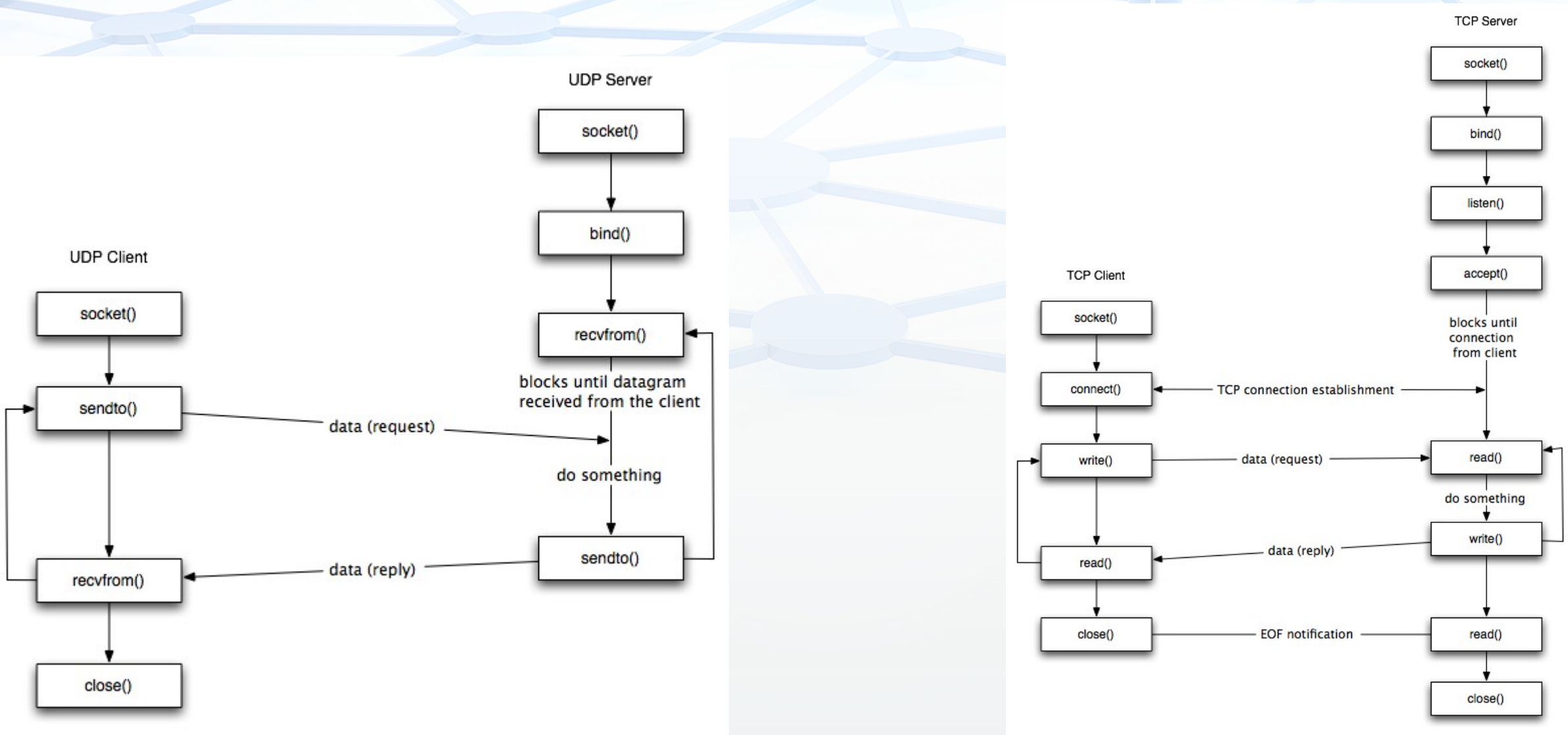
Berkeley Sockets

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



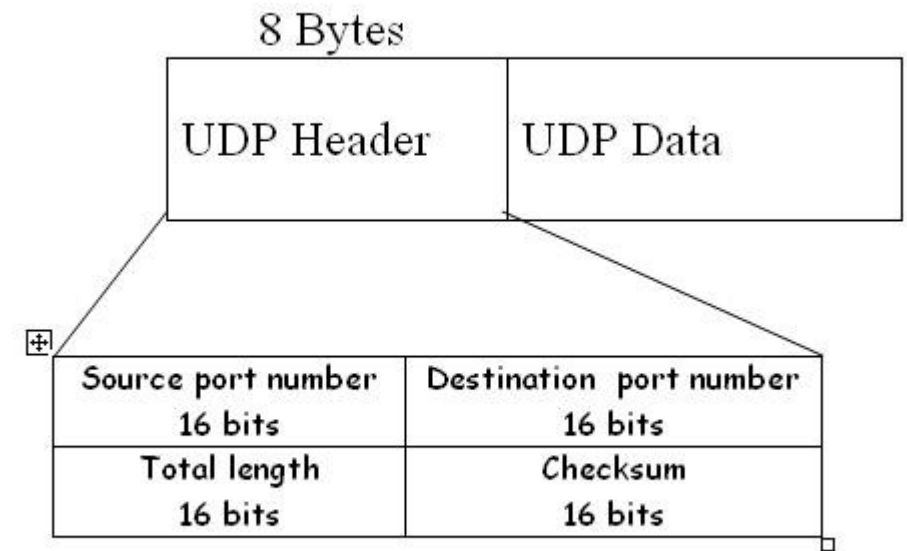
zdroj: <https://stackoverflow.com/questions/152457/what-is-the-difference-between-a-port-and-a-socket/152863>

Sockety: Porovnání UDP a TCP volání



Prokoly v L4: UDP

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

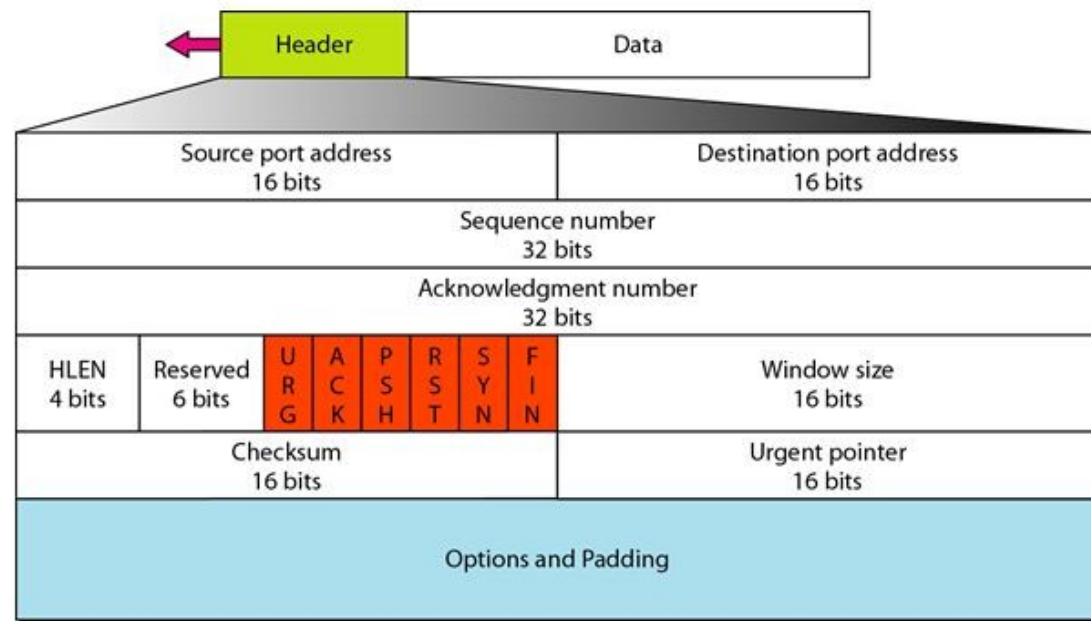


zdroj:

https://en.wikibooks.org/wiki/Communication_Networks/TCP_and_UDP_Protocols/

Prokoly v L4: TCP

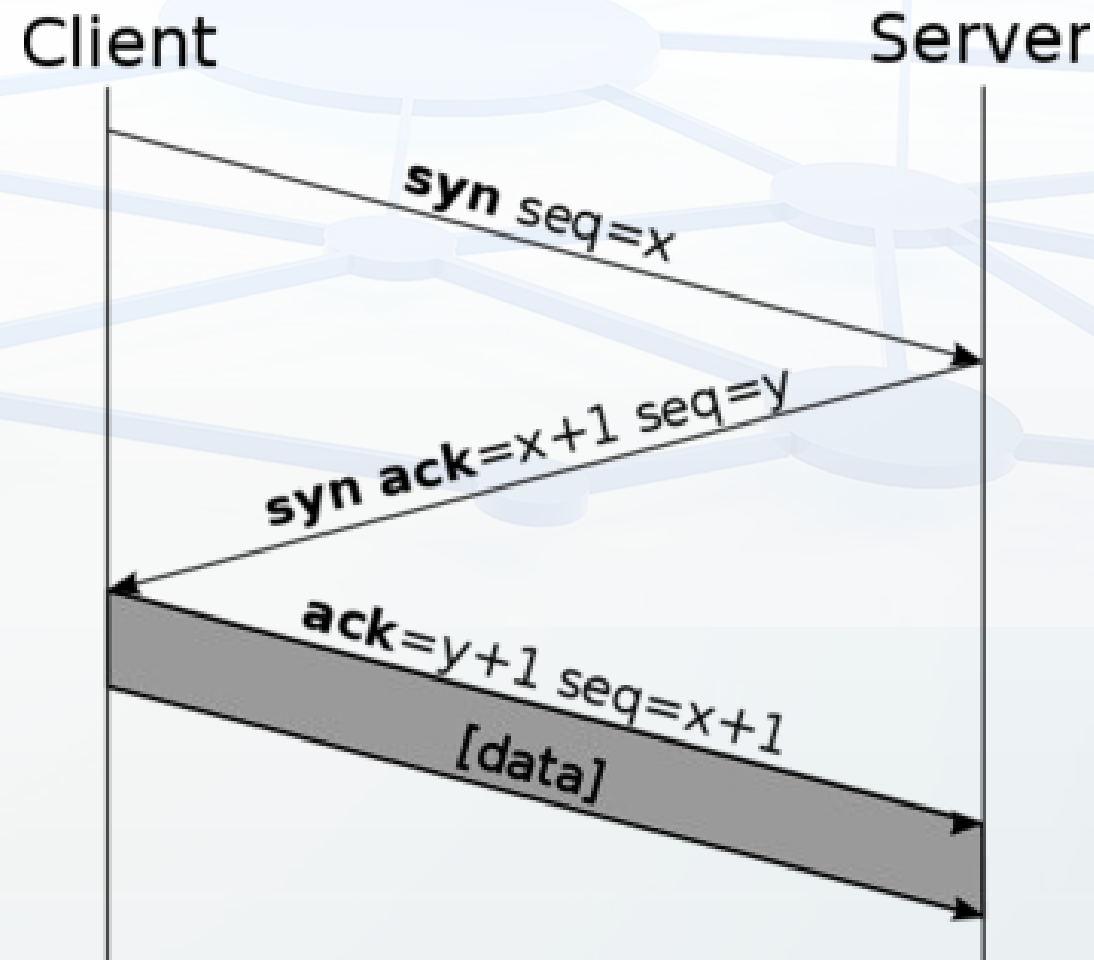
- TCP – Transmission Control Protocol
- Spojovaný
 - Před začátkem samotného přenosu je dohodnuto spojení
 - Příjemce tedy musí s přenosem souhlasit
- Potvrzovaný
 - Během přenosu j potvrzována správnost přenosu dat
 - Pokud během přenosu dojde k chybě jsou data znovu vyžádána
- Používá bufferování data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



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

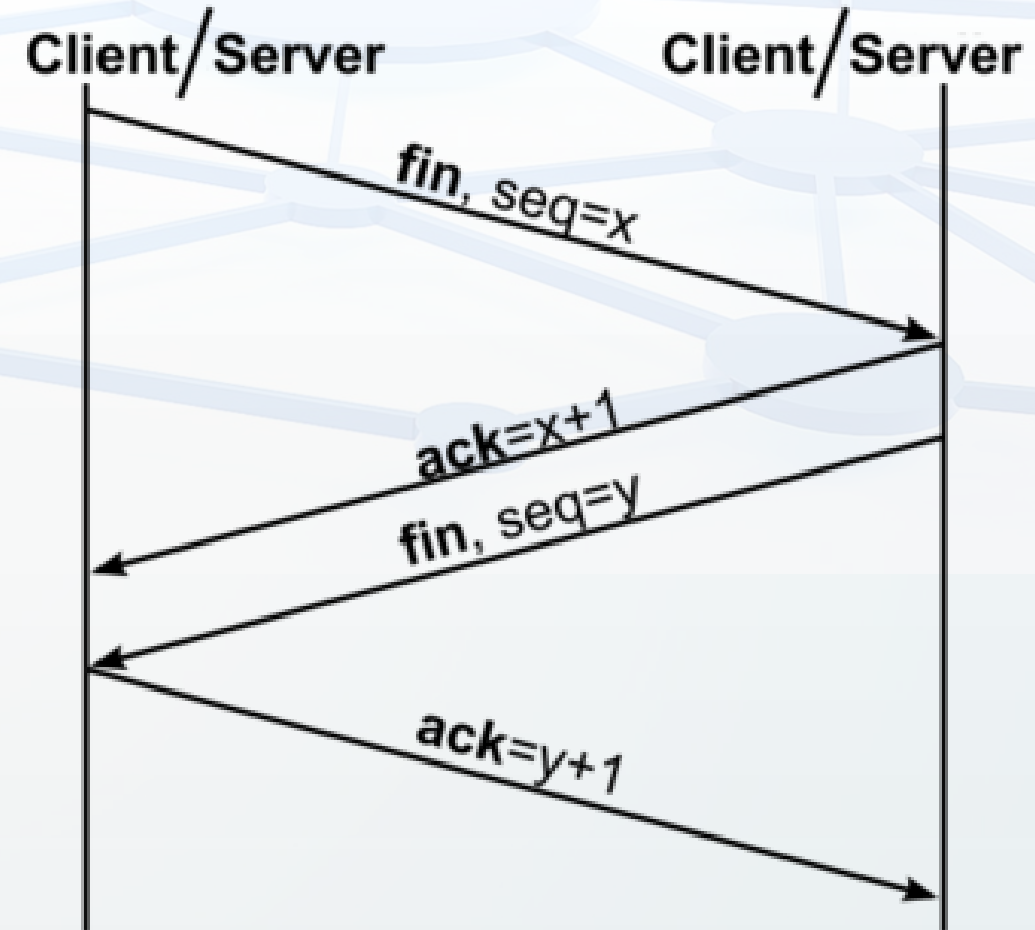
Prokoly v L4: TCP: Navazování spojení

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



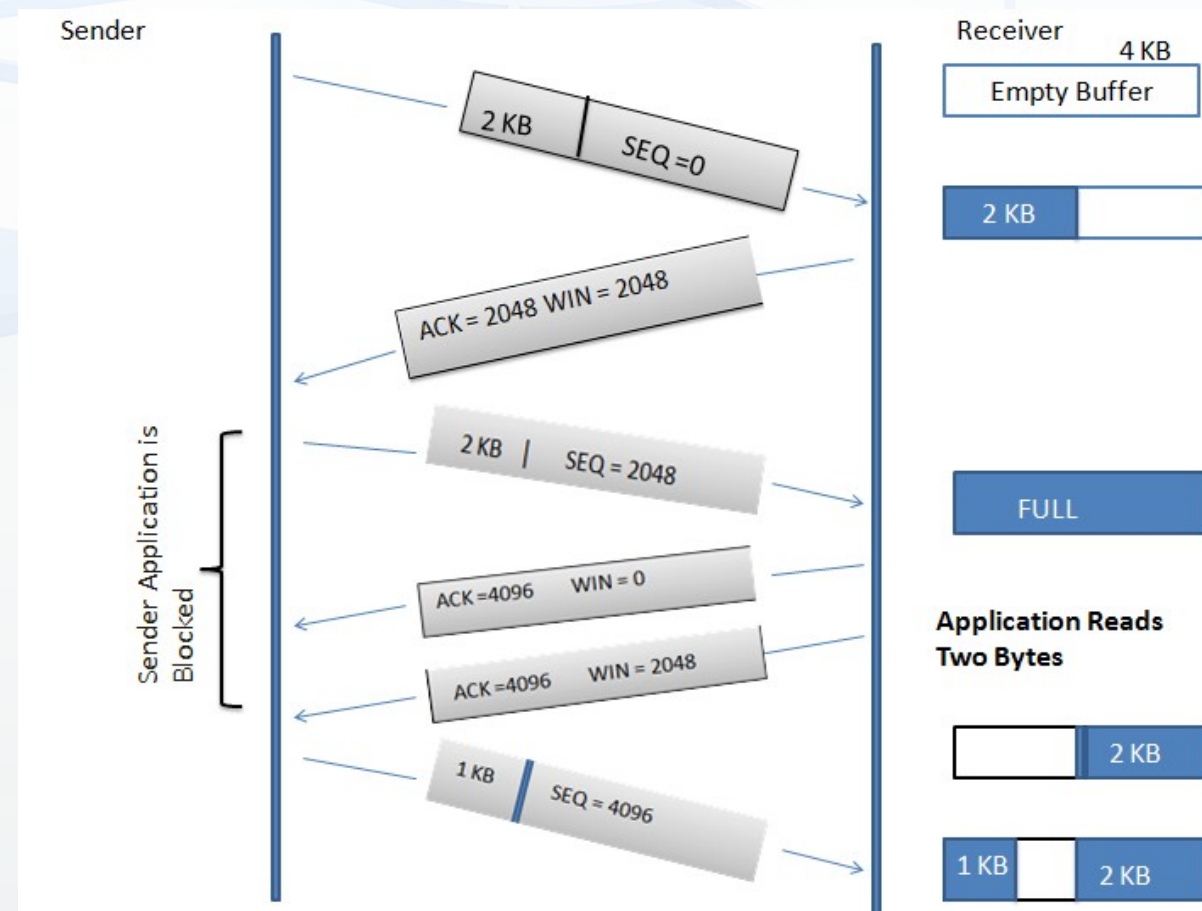
Prokoly v L4: TCP: Ukončování spojení

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



Prokoly v L4: TCP: Řízení toku dat

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



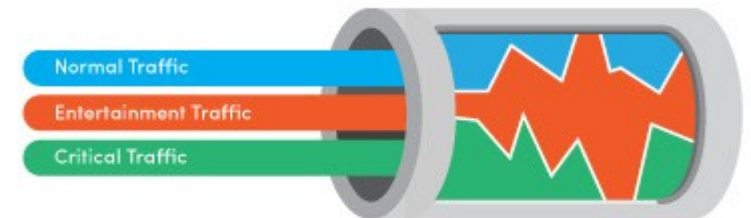
Prokoly v L4: TCP: Předcházení zahlcení

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

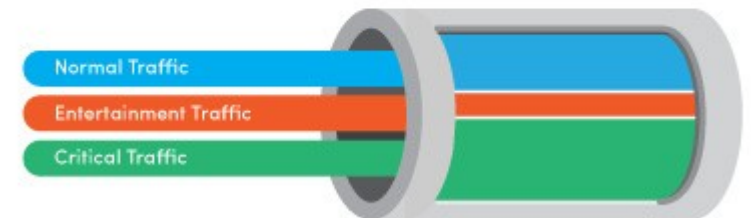
QoS: Quality of Service

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

Bandwidth WITHOUT QoS



Bandwidth WITH QoS



QoS: Možnosti implementace

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

QoS: Principy

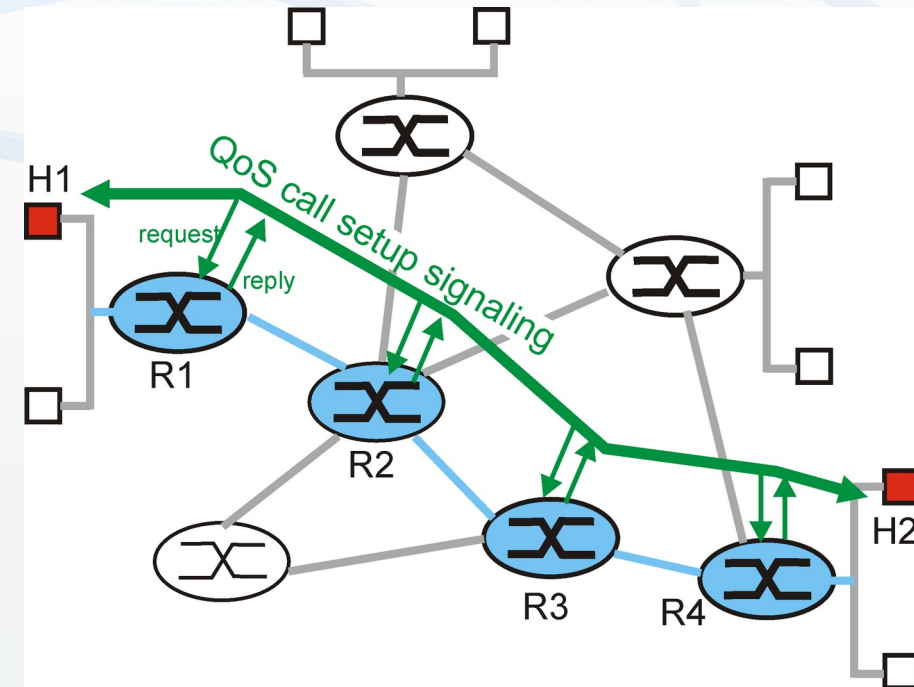
- Značkování paketů
 - Jednotlivé pakety budou rozděleny do tříd dle požadovaných parametrů
 - Router pak pracuje s pakety různě na základě značky
 - Jednotlivé třídy musí mít různé priority
 - WWW přenos bude mít nižší prioritu než VOIP, protože web dokáže dočasně vybrat celé pásmo a blokovat hovory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

QoS: Možnosti řešení

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

QoS: Možnosti implementace: Integrated Services

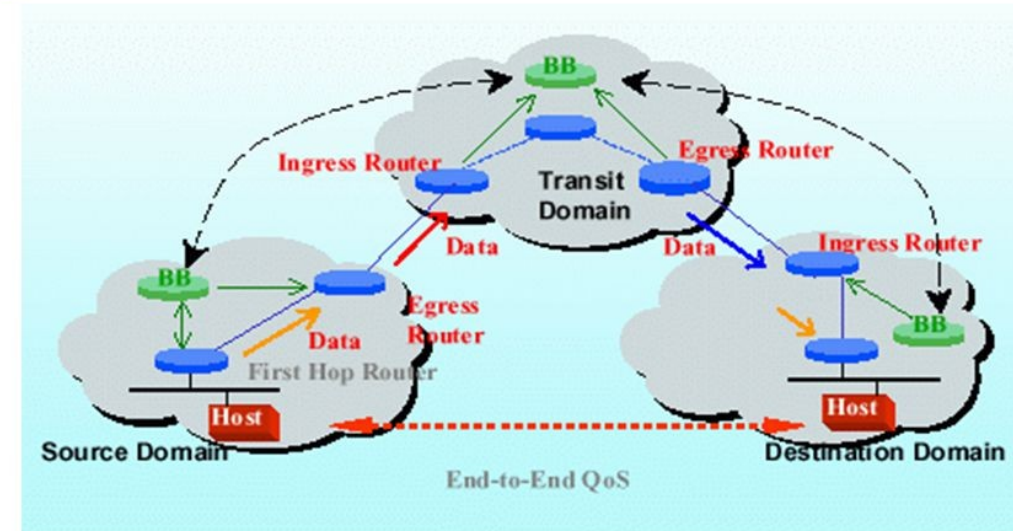
- Integrated Services – jednotné služby
 - Funguje na principu rezervace zdrojů
 - Díky tomu garantuje požadované zdroje
 - Část zdrojů je obecnému IP odebrána a jejich využití / rozdělení řeší QoS
 - To realizuje alokace po celé cestě v síti
 - Defakto vytváří virtuální okruh, které se v přenosech dle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



QoS: Možnosti implementace: Differentiated Service

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

DiffServ - Operation



QoS: Reálné použití

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