

Lekce 9: Transportní vrstva

Jiří Peterka

úkoly „vyšších“ vrstev

- **L7: aplikační vrstva**

ve smyslu: fungující dle standardů

- běží v ní jen „standardizované části“ aplikací (nikoli jejich uživatelské rozhraní)
 - obecně: běží zde jen ty části aplikací, které implementují příslušné aplikační protokoly

- **L6: prezentační vrstva**

- zajišťuje to, aby obě (všechny) strany interpretovaly přenášená data stejně
 - v praxi: zajišťuje konverze a převod dat z/do formátů, vhodných pro přenos

viz lekce
3

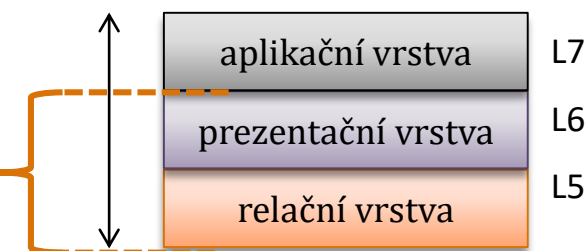
- **L5: relační vrstva**

- zajišťuje „podporu relací“
 - vedení/udržování spojení
 - průběh komunikace
 -

viz lekce
3

jen v ISO/OSI,
nikoli v TCP/IP

vrstvy orientované na aplikace



- **L4: transportní vrstva**

- zajišťuje „přizpůsobení“
 - mezi možnostmi nižších vrstev a požadavky vyšších vrstev
- zajišťuje multiplexing/demultiplexing
- zajišťuje end-to-end komunikaci
- může zajišťovat „další“ úkoly
 - podporu QoS, řízení toku, předcházení zahlcení,



vrstvy orientované na přenos

jeden z úkolů transportní vrstvy

- **přizpůsobovat požadavky vyšších vrstev možnostem nižších vrstev**

- síťová vrstva (L3) může fungovat jedním způsobem, nebo několika (málo) různými způsoby

- TCP/IP: L3 funguje jen 1 způsobem:

1. nespojovaně, nespolehlivě, best effort, po blocích (paketech)

- zajišťuje protokol IP

- OSI/OSI: L3 může fungovat 2 různými způsoby

1. spojovaně, spolehlivě, s podporou QoS, po blocích (á la X.25)

- zajišťuje protokol **CONP** (Connection Oriented Network Protocol), a

- **CMNS** (Connection Mode Network Service) – služba pro vyšší vrstvy

2. nespojovaně, nespolehlivě, best effort, po blocích (á la IP)

- zajišťuje protokol **CLNP** (Connectionless Network Protocol), a

- **CLNS** (Connectionless Network Service)

- požadavky vyšších vrstev se mohou týkat:

- spojovaného/nespojovaného způsobu přenosu

- spolehlivosti/nespolehlivosti přenosu

- podpory QoS (místo best effort)

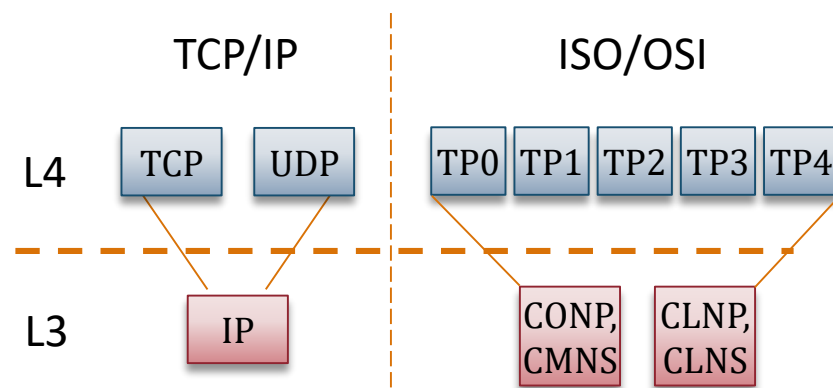
- proudového přenosu (místo po blocích)

-

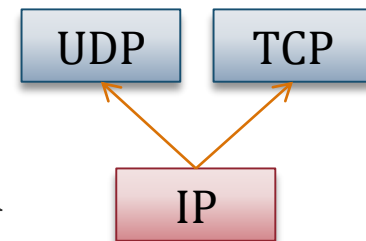
- transportní vrstva (L4) může fungovat více různými způsoby:

- TCP/IP: L4 funguje 2 různými způsoby

- ISO/OSI: L4 funguje 5 různými způsoby



transportní vrstva TCP/IP



- vyšším vrstvám nabízí 2 varianty „přizpůsobení“

- 2 varianty transportních služeb: transportní protokoly TCP a UDP

a) minimální změna

– transportní protokol UDP

- nespojovaný a nespolehlivý
 - stejně jako protokol IP
- velmi jednoduchý protokol
 - stejně jako protokol IP
- funguje stylem best effort, bez QoS
 - stejně jako protokol IP
- nezajišťuje řízení toku ani nepředchází zahltění
 - stejně jako protokol IP
- přenáší data po blocích (datagramech)
 - stejně jako protokol IP

b) „maximální“ změna

– transportní protokol TCP

- spojovaný a spolehlivý
 - na rozdíl od protokolu IP
- velmi složitý a komplexní protokol
 - na rozdíl od protokolu IP
- funguje stylem best effort, bez QoS
 - stejně jako protokol IP
- zajišťuje řízení toku a předchází zahltění
 - na rozdíl od protokolu IP
- přenáší data jako proud bytů (stream)
 - na rozdíl od protokolu IP

- postupně vznikla poptávka po dalších transportních protokolech

- **SCTP** (Stream Control Transmission Protocol): spolehlivý, spojovaný (ale jinak než TCP)
- **DCCP** (Datagram Congestion Control Protocol): nespolehlivý (jako UDP), spojovaný



zatím se ale moc nepoužívají

transportní vrstva ISO/OSI

- vyšším vrstvám nabízí 5 variant „přizpůsobení“

- 5 různých transportních protokolů

- TP0, TP1, TP2, TP3, TP4

- liší se v tom, zda:

- dokáží fungovat nad spojovanou L3 (CONP/CMNS)
- dokáží fungovat nad nespojovanou L3 (CLNP/CLNS)
- zajišťují spolehlivost na L4
- umožňují více L4 spojení po jednom L3 spojení
- zajišťují řízení toku
- zajišťují zotavení po chybě
- zajišťují obnovu spojení po přerušení
-

TP0	TP1	TP2	TP3	TP4
+	+	+	+	+
-	-	-	-	+
-	+	-	+	+
-	-	+	+	+
-	-	+	+	+
-	+	-	+	+
-	+	-	+	-

- transportní protokol TP4 je „podobný“ TCP

- ale ne zcela identický

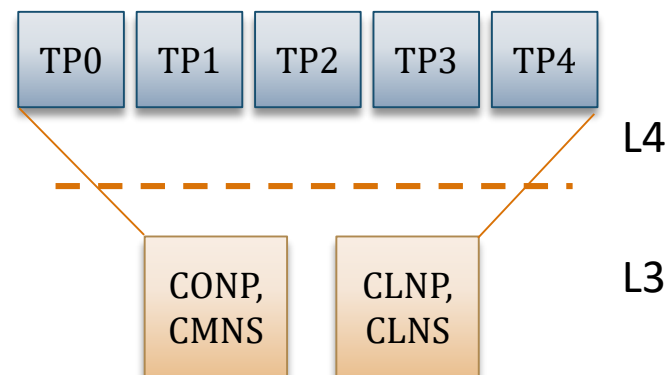
- srovnání s TCP/IP

- TCP/IP: málo variant, postupně přidávání dalších

- ISO/OSI: již od počátku hodně variant

- relativně komplikovaných

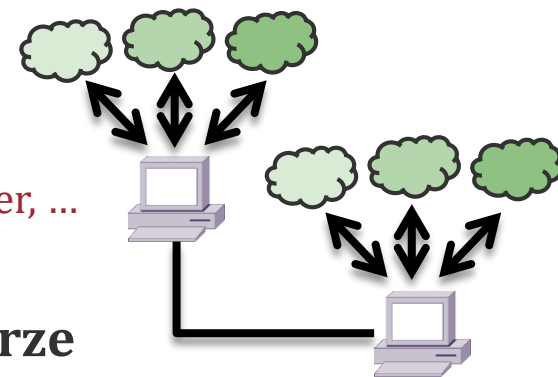
dnes se protokoly ISO/OSI již nepoužívají



další úkol transportní vrstvy

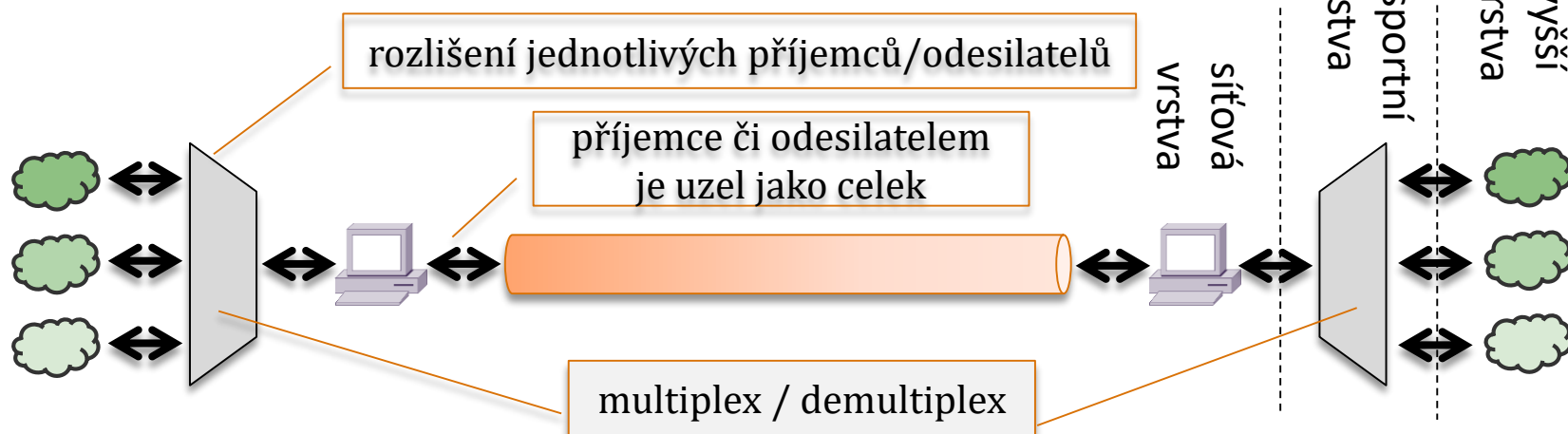
• připomenutí:

- na síťové vrstvě (i na vrstvě síťového rozhraní) se adresují jednotlivé uzly jako celky
 - příjemcem či odesilatelem je uzel jako celek
- ale: v rámci každého uzlu existuje více různých entit, které mohou vystupovat v roli odesílatelů či příjemců dat
 - například: instance (okna) browseru, emailový klient, web server, ...
 - a je třeba je rozlišit



• rozlišení se provádí na úrovni transportní vrstvy, skrze

- „sloučení“ několika samostatných přenosů do jedné společné přenosové cesty
 - **multiplex**
- „zpětné rozložení“ na odpovídající samostatné přenosy
 - **demultiplex**



porty a adresování

• co je nutné pro korektní rozlišení jednotlivých entit v rámci uzlů?

– existence mechanismu, který umožní předávat/přebírat data selektivně od jednotlivých entit

• řešení:

– více přechodových bodů mezi transportní vrstvou a bezprostředně vyšší vrstvou

- idea: pro každou jednotlivou entitu bude určen jiný takovýto přechodový bod

– možnost vhodného adresování

- otázka: mají být adresovány přímo příslušné entity, nebo pouze přechodové body?

– adresování entit: bylo by problematické

- protože entity vznikají i zanikají dynamicky
- protože na různých platformách mohou mít entity různou podobu (procesy, úlohy, ...)
- a také různé identifikátory

– navíc:

- v praxi nejde o to, o kterou konkrétní entitu jde – ale co dělá
- je potřeba adresovat „toho, kdo poskytuje takovou a takovou službu“

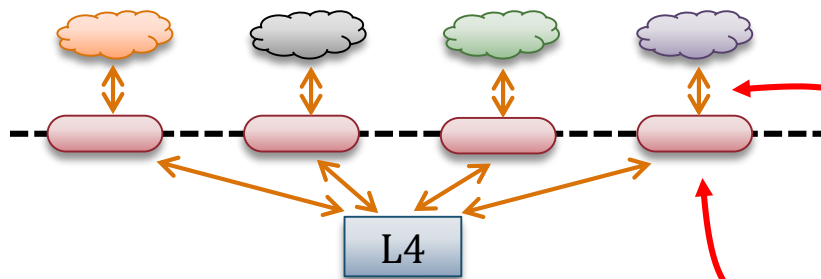
• odpověď:

– adresují se přechodové body

- které mohou být (a jsou) statické
- adresa přechodového bodu reprezentuje určitou službu, která je poskytována

– konkrétní entity se dynamicky asociují s konkrétními přechodovými body

- asociuje se ta entita, která skutečně poskytuje příslušnou službu



– v ISO/OSI

- jde o **body SAP** (Service Access Point)

– v TCP/IP

- jde tzv. **porty**

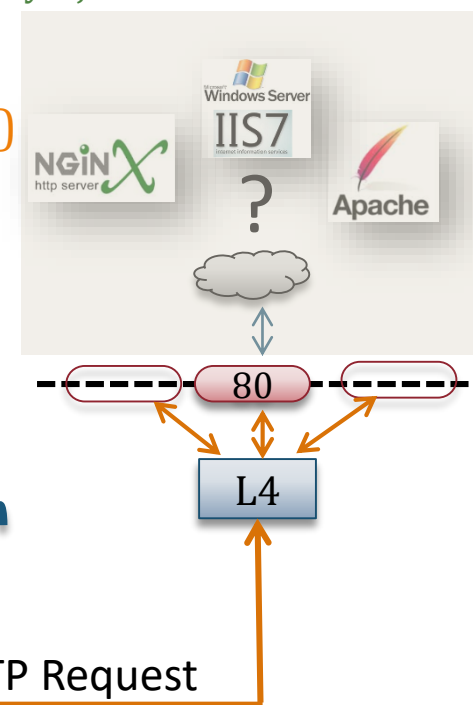
výhody zvoleného řešení

- **rekapitulace:**

- adresují se přechodové body (ISO/OSI: body SAP, TCP/IP: porty)
 - nikoli konkrétní entity, které vznikají a zanikají dynamicky, mohou být na různých platformách různé atd.
- adresy přechodových bodů odpovídají poskytovaným službám
 - k přechodovému bodu se dynamicky „asociuje“ ta entita, která v rámci daného uzlu poskytuje příslušnou službu
 - příklad (TCP/IP): na portu č. 80 je poskytována služba HTTP serveru
 - tj. s tímto portem je dynamicky asociována ta entita, která skutečně poskytuje službu HTTP serveru

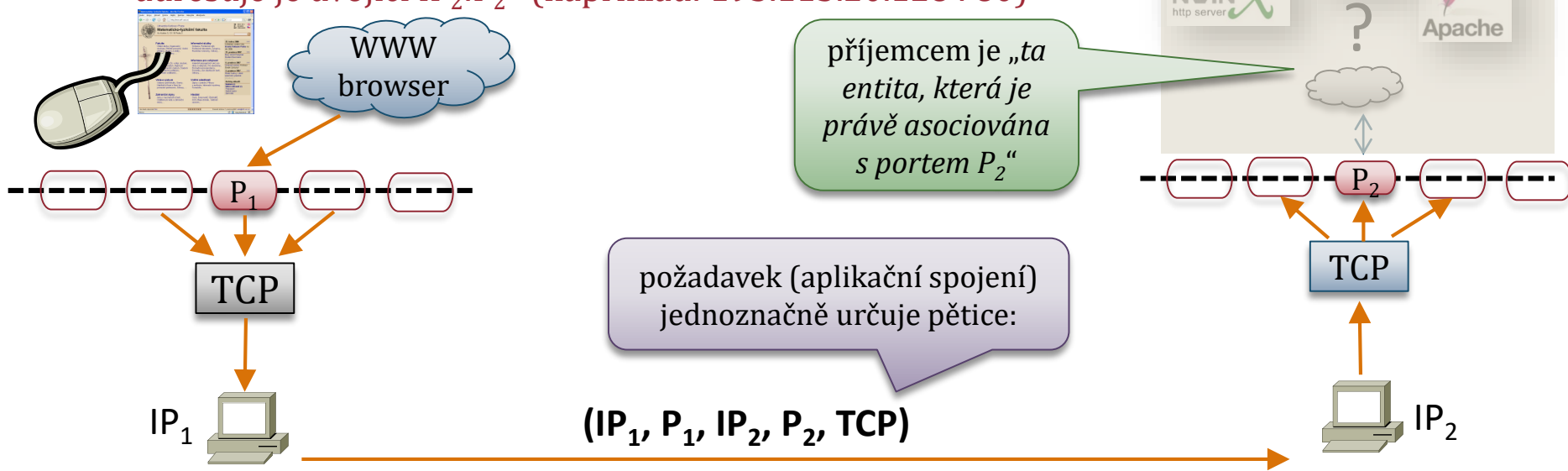
- **výhody zvoleného řešení:**

- přechodové body mohou být statické (existovat apriorně, všude)
 - lze dopředu počítat s jejich existencí
- přechodové body a jejich adresy mohou být abstraktní (všude stejné)
 - zatímco konkrétní entity, které se s nimi asociují, mohou být všude jiné
- „viditelnost“:
 - z vně (od jiných uzlů) jsou „vidět“ pouze adresy přechodových bodů
 - nikoli identifikátory samotných entit
 - které mohou být na různých platformách různé



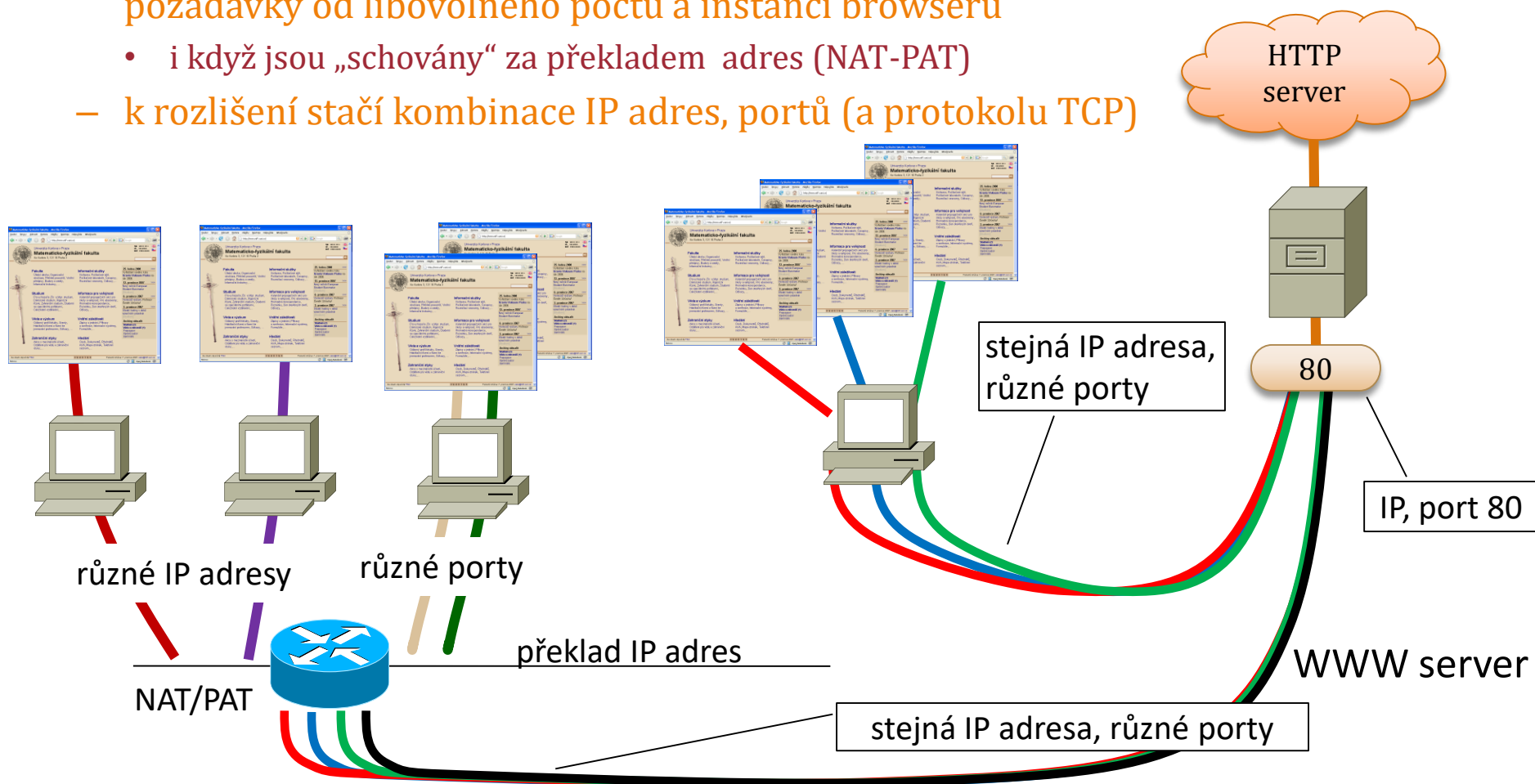
porty v TCP/IP a identifikace spojení

- **porty jsou číslovány (adresovány pomocí čísel)**
 - jde o čísla v rozsahu 16 bitů: s hodnotami od 0 do 65 535
- **číslo portu představuje transportní adresu (adresu na transportní vrstvě)**
 - tato adresa je relativní - je vztažena (relativně) jen k danému uzlu
 - absolutní adresu představuje až dvojice <síťová adresa>:<transportní adresa>
- **princip identifikace (aplikačního) spojení**
 - odesílatel odesílá svá data „z konkrétního portu na konkrétním uzlu“
 - z uzlu se síťovou adresou IP_1 , z portu P_1
 - odesílatel posílá svá data „na konkrétní port na konkrétním uzlu“
 - adresuje je dvojici $IP_2:P_2$ (například: 195.113.20.128 : 80)



rozlišení více (aplikačních) spojení

- jedna entita (proces, ...) může komunikovat s více různými entitami na stejném uzlu, i na různých uzlech
- například (TCP/IP):
 - jeden HTTP server (na portu 80) dokáže „obsloužit“ (korektně rozlišit) požadavky od libovolného počtu a instancí browserů
 - i když jsou „schovány“ za překladem adres (NAT-PAT)
 - k rozlišení stačí kombinace IP adres, portů (a protokolu TCP)



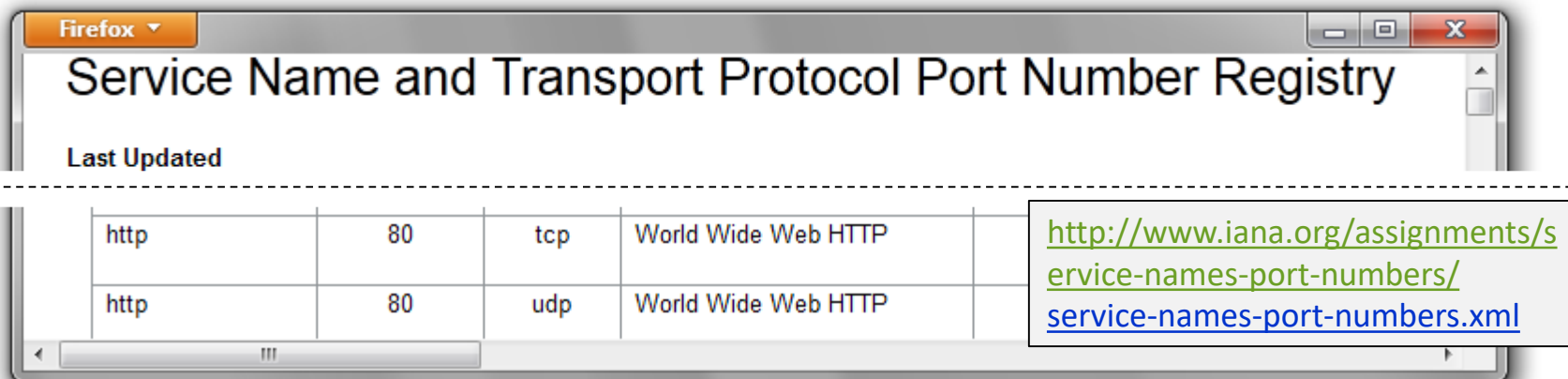
konvence o dobře známých portech

- **otázka:**

- jak webový klient (browser) ví, že má posílat své požadavky na port č. 80?
 - jak ví, že s tímto portem bude asociována entita, poskytující služby HTTP serveru?
- proč nepotřebuje vědět, která konkrétní entita je právě asociována s daným portem?
 - potřebuje pouze, aby to byla „*taková, která poskytuje služby, odpovídající portu*“

- **odpověď:**

- může vycházet z konvence o tzv. **dobře známých portech** (well-known ports)
 - představa: jde o tabulku, kterou vede a udržuje někdo důvěryhodný
 - organizace IANA, dnes součást ICANN
 - konkrétně jde o porty 0 až 1023
 - dříve byla konvence o dobře známých portech zveřejňována formou RFC dokumentu
 - dnes je (průběžně) publikována on-line



Service Name and Transport Protocol Port Number Registry				
Last Updated				
http	80	tcp	World Wide Web HTTP	http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml
http	80	udp	World Wide Web HTTP	

dobře známé a registrované porty

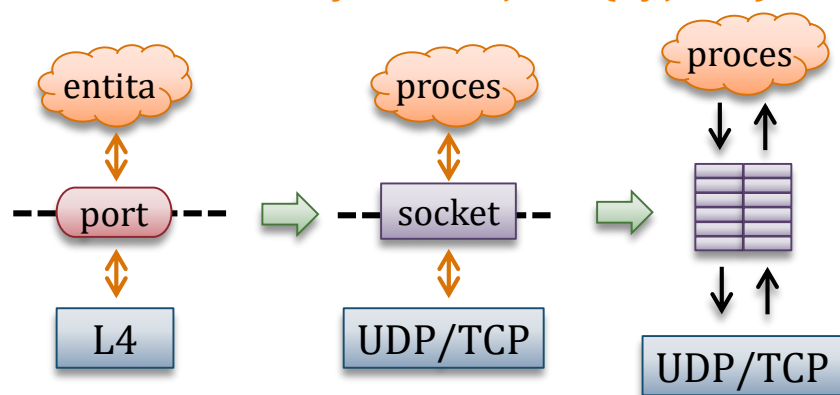
- **dobře známé porty (0 až 1023)**
 - konvence zajišťuje unikátnost účelu
 - stejný port slouží jen jednomu účelu
 - je pro daný účel vyhrazen
 - typicky: „pro systémové věci“
 - **neměl by se používat pro jiné účely**
- **registrované porty (1024 až 49151)**
 - konvence zajišťuje unikátnost účelu
 - každý port je (za)registrován jen pro jeden účel
 - **ale může se používat i pro jiné účely**
 - i pro „uživatelské věci“
 - proto též tzv. **uživatelské porty**
- **dynamické porty (49152 až 65535)**
 - žádná konvence o jejich využití
 - mohou být využity pro jakékoli účely
 - bez potřeby/možnosti registrace
 - alokují se podle potřeby
 - dynamicky, např. pro odchozí spojení

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 !!!

porty vs. sockety (v TCP/IP)

- **porty jsou logickou záležitostí**
 - na všech platformách jsou stejné
 - identifikované svými čísly
 - jejich konkrétní implementace je závislá na platformě
 - nejčastěji je port implementován jako socket
- **socket je datovou strukturou charakteru (obousměrné) fronty**
 - z jedné strany se do něj zapisuje (vkládá), z druhé strany se z něj čte (vyjímá)

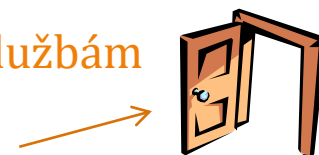


- aplikace si vytváří sockety, které pak asociuje s konkrétním portem

```
proces.newsock = SOCKET(...); BIND(newsock, číslo portu);
```

- **socket vznikl jako abstrakce souboru v BSD Unixu**
 - pro potřeby práce se soubory
 - a také pro vstupy a výstupy
 - pracuje se s ním stylem „(create)-open-read-write-close“
- **sockety byly upraveny i pro potřeby síťování**
 - byly rozšířeny o další možnosti
 - např. o asociaci s porty (BIND)
- **"socketové API"**
 - takové API, které procesům vytváří iluzi, že pracují se sockety
 - např. rozhraní WINSOCK
- **socket si lze představit jako analogii brány**
 - vedoucí k síťovým službám

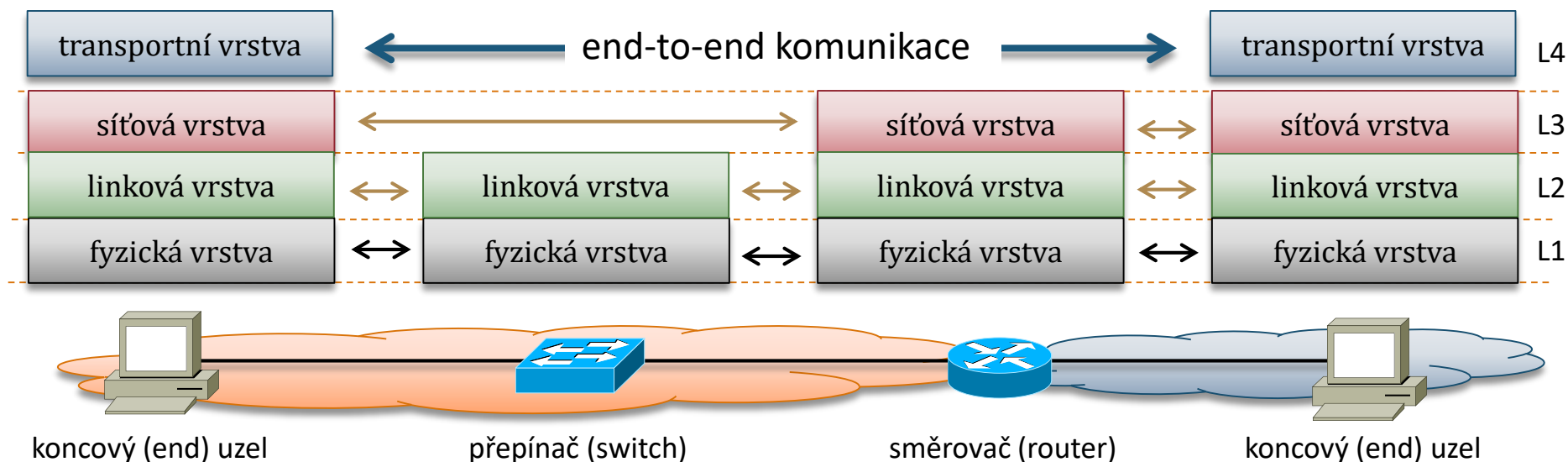
socket



další úkol transportní vrstvy

- **zajišťovat end-to-end komunikaci**
 - vzájemnou (a přímou) komunikaci mezi entitami koncových (end) uzlů
- **připomenutí:**
 - transportní vrstva je přítomna jen v koncových uzlech sítě
 - není přítomna ve vnitřních uzlech – ve směrovačích, přepínačích, opakovačích
- **výhody**
 - (pro end-to-end komunikaci) není nutná žádná podpora ve vnitřních uzlech sítě
 - požadované funkce/služby lze implementovat snadno a efektivně (typicky v SW)
 - na „běžných“ počítačích v roli koncových uzlů
 - zatímco směrovače jsou typicky specializovaná zařízení s vlastním specifickým HW i SW

jsou rozlišeny/adresovány pomocí portů



co je vhodné řešit jako end-to-end?

• otázka:

- které činnosti je vhodné řešit na end-to-end bázi (na transportní či vyšší vrstvě)?
 - a které raději na „hop-to-hop“ bázi (na fyzické, linkové či síťové vrstvě)?

• v úvahu připadá:

– spolehlivost přenosových služeb

- ISO/OSI: zajišťuje spolehlivost na všech vrstvách (na síťové i transportní)
- TCP/IP: až na transportní vrstvě (a jen volitelně – v rámci transportního protokolu TCP)
 - síťová vrstva (protokol IP) funguje nespolehlivě
 - výhoda end-to-end řešení: jednotlivé entity (protokoly, služby) si mohou vybrat, zda chtějí spolehlivé či nespolehlivé služby

end-to-end

– spojovaný charakter přenosových služeb

- ISO/OSI: spojovaně na všech vrstvách
- TCP/IP: jako se spolehlivostí
 - síťová vrstva (protokol IP) funguje nespojovaně
 - spojovaně až na transportní vrstvě (TCP)

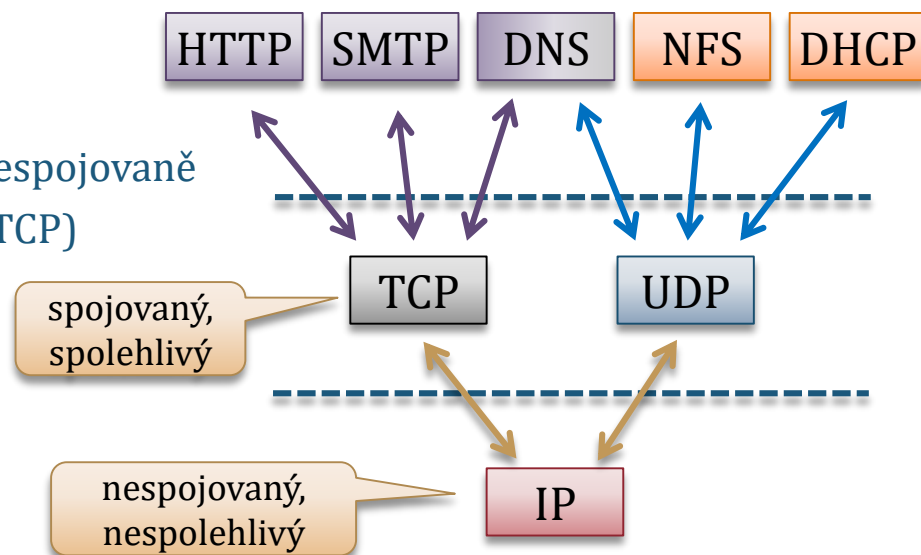
end-to-end

– fragmentaci a defragmentaci

– řízení toku, předcházení zahlcení

– podporu kvality služeb (QoS)

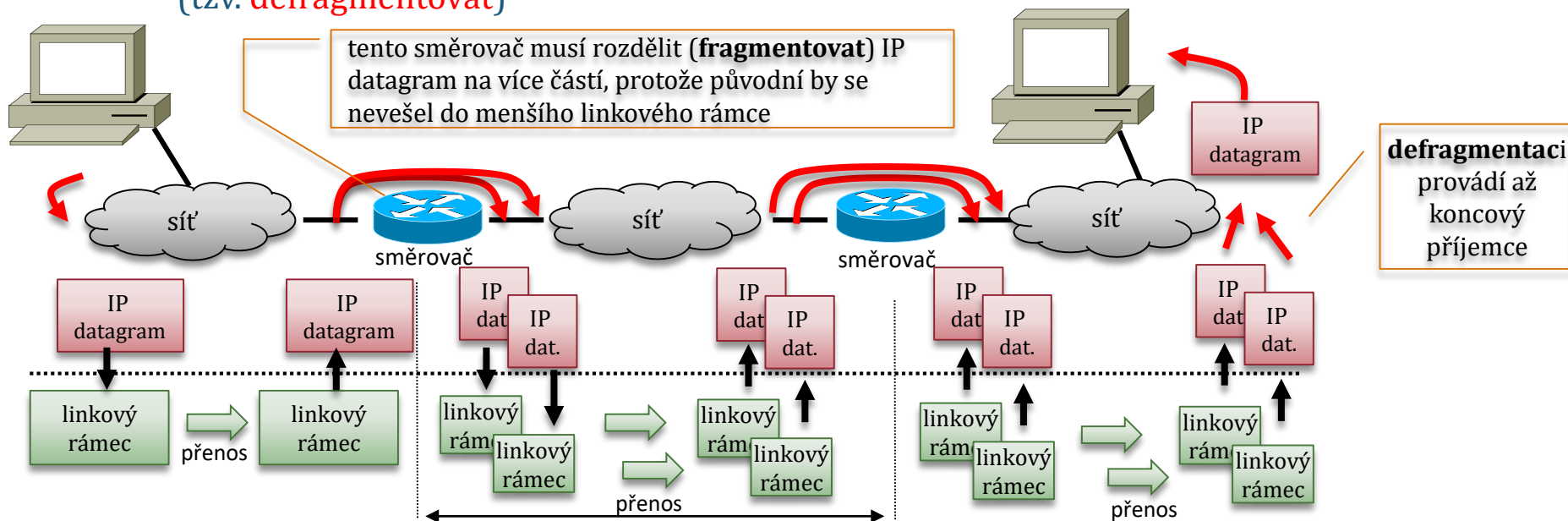
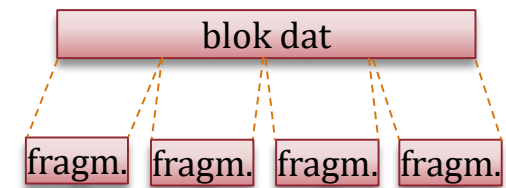
–



fragmentace a defragmentace

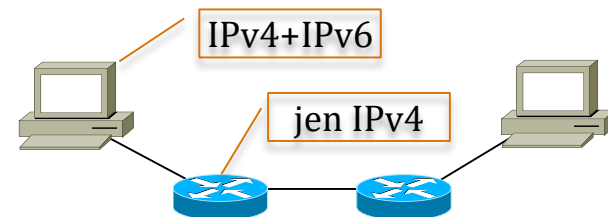
• nutnost fragmentace

- bloky dat (pakety, rámce), přenášené na určité vrstvě, mají vždy určitou max. velikost
- může se stát, že blok na vyšší vrstvě je příliš velký na to, aby se vešel do bloku na bezprostředně nižší vrstvě
 - příklad: Ethernet má max. velikost (nákladové části rámce) 1500 nebo 1492 bytů, zatímco celý IP paket může mít až 2^{16} (tj. 65 536) bytů
- pak je nutné blok vyšší vrstvy rozdělit (**fragmentovat**) na několik dílčích částí (**fragmentů**)
 - tak velkých, aby se již vešly do bloků nižší vrstvy
 - a „na konci“ (u příjemce) zase jednotlivé fragmenty poskládat zpět do původního bloku (tzv. **defragmentovat**)



problémy fragmentace

- **fragmentace vyžaduje podporu v přenosových protokolech**
 - aby bylo možné označit ty fragmenty, které patří k sobě, určit jejich pořadí i počet
 - příklad (síťová vrstva TCP/IP):
 - protokol IP podporuje fragmentaci (dělení IP paketů na fragmenty)
 - v IPv4 může fragmentovat koncový uzel (odesílatel) i kterýkoli směrovač „po cestě“
 - v IPv6 může fragmentovat jen koncový uzel (odesílatel)
- **nevýhody a problémy fragmentace**
 - je s tím spojena určitá (nenulová) reže
 - a to i když k fragmentaci nedochází
 - v hlavičkách IPv4 datagramů jsou pro potřeby fragmentace vždy vyhrazeny určité položky
 - zvyšuje to (dopady) chybovosti
 - pokud se ztratí či poškodí byť jen jediný fragment, je nepoužitelný celý původní blok
 - zavádí to stavový způsob fungování do jinak bezstavového
 - protokol IP standardně funguje bezstavově
 - nepřechází mezi různými stavy, neřeší přechody mezi stavy, nemá žádné time-outy
 - ale kvůli fragmentaci musí čekat na všechny fragmenty, musí mít nějaký time-out na doručení chybějících fragmentů
 - musí „ukládat“ do bufferů dosud přijaté fragmenty
 - musí čekat na vypršení time-outu
 - a pokud nedostal všechny fragmenty, musí zahodit všechny ty, které dosud přijal

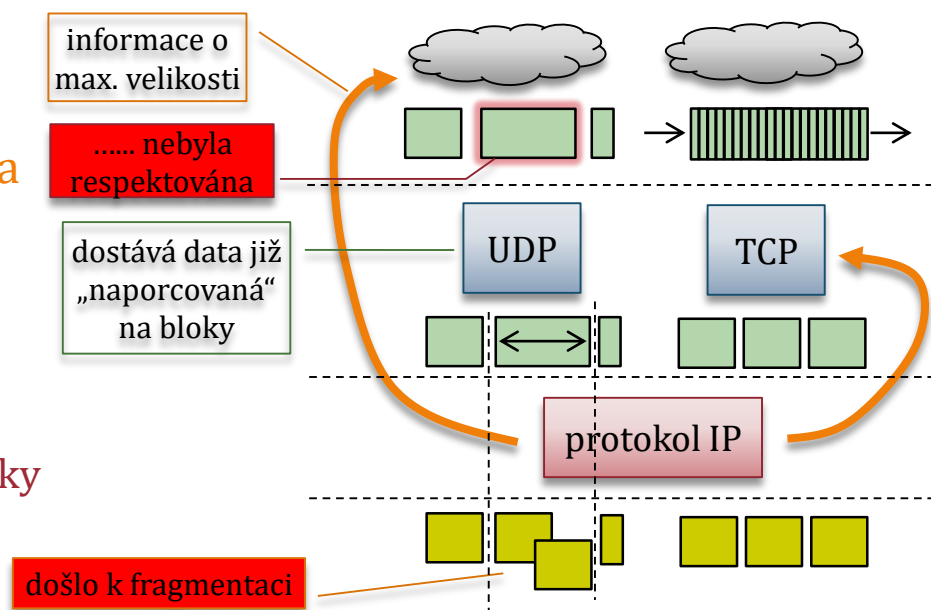


jak na fragmentaci?

- **obecně: snažit se, aby k potřebě fragmentace docházelo co nejméně**
 - generovat jen tak velké bloky dat, aby k fragmentaci nedocházelo
- **je možné (nezávisle na sobě):**
 1. podporovat fragmentaci v přenosových protokolech různých vrstev
 - v úvahu připadají hlavně: síťová a transportní
 2. těm entitám, které „porcují data na bloky“, poskytnout informaci o maximální velikosti bloku, který nebude nutné fragmentovat
 - typicky: velikost (nákladové části) linkového rámce
 3. těm entitám, které generují a odesílají data, vytvářet iluzi datového proudu
 - tedy představu toho, že nemusí/nemohou data porcovat na bloky, ale mohou je přenášet po jednotlivých bytech, jako souvislý proud

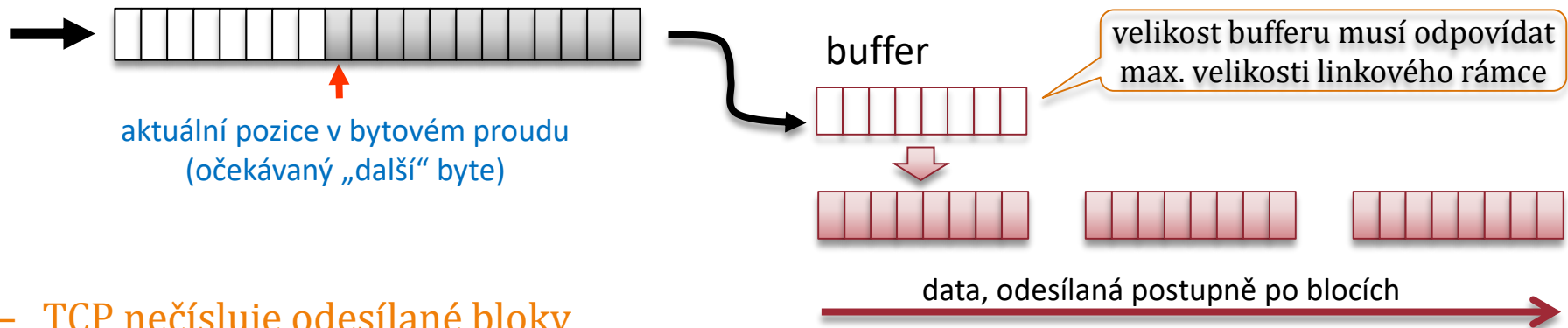
řešení v TCP/IP

- ad 1: podpora fragmentace je zabudována v protokolu IP
- ad 2: informaci o max. velikosti bloku (MTU) dostávají ty aplikační entity, které využívají protokol UDP
 - ty by měly vytvářet vhodně malé/velké bloky
- ad 3: protokol TCP vytváří aplikačním entitám iluzi bytového proudu



TCP: bytový proud

- **protokol TCP dostává data (od aplikačních entit) po jednotlivých bytech**
 - vytváří jim iluzi bytového proudu (že se data skutečně přenáší po jednotlivých bytech)
 - sám ale využívá služeb protokolu IP, který přenáší celé bloky dat (IP datagramy) a nikoli jednotlivé byty
- **proto:**
 - TCP ve skutečnosti ukládá jednotlivé byty do svého bufferu
 - a jeho obsah odesílá (jako blok, tzv. TCP segment) až tehdy, když se celý naplní
 - případně když si aplikace explicitně vyžádá předčasné odeslání (příkaz PUSH)



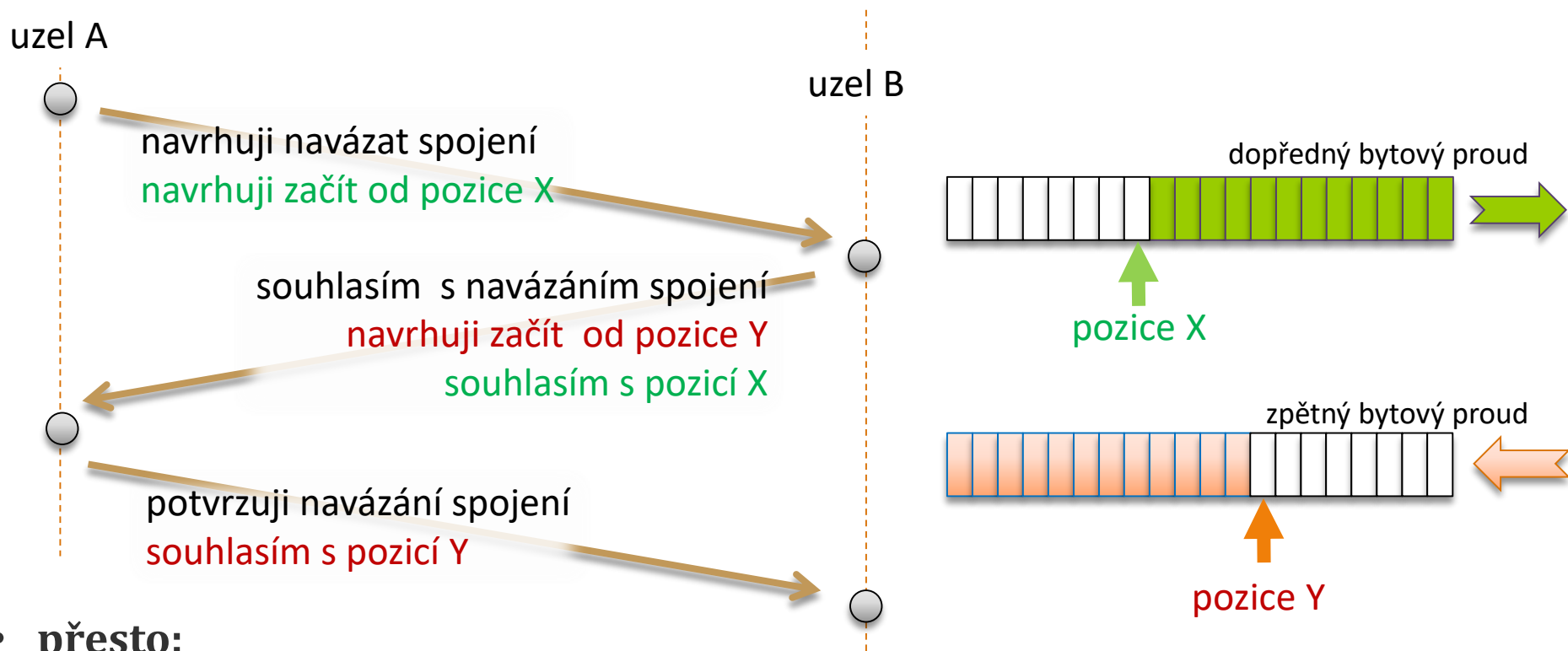
- TCP nečísluje odesílané bloky
 - ale udává pozici právě přenášených dat v bytovém proudu (jako 32bitové číslo)
 - využívá se při zpětném sestavování bytového proudu, při potvrzování atd.
 - kvůli bezpečnosti se pozice nepočítá od 0, ale od náhodně zvolené počáteční hodnoty
 - tyto počáteční hodnoty musí být zvoleny a předány druhé straně při navazování spojení
 - pro oba směry přenosu

spojovaný charakter komunikace

- **připomenutí:**
 - v sítích, fungujících na principu přepojování paketů, má spojovaný charakter komunikace vždy virtuální charakter
 - jde o virtuální okruhy, nikoli o okruhy skutečně vyhrazené (ve smyslu přepojování okruhů)
 - na nižších vrstvách (do vrstvy síťové včetně) jde nejčastěji o permanentní virtuální okruhy
 - tj. okruhy zřizované ručně/správcem a existující trvale (permanentně)
 - na vyšších vrstvách (transportní a výše) jde typicky o okruhy zřizované „na žádost“
 - tzv. komutované, vytvářené až v okamžiku potřeby, a následně také rušené
- **navazování (i rušení) spojení „na žádost“ je složité a má řadu nástrah**
 - musí být ošetřeny všechny potenciální problémy, které mohou vzniknout
 - že se ztratí žádost o navázání spojení, nebo odpověď na takovouto žádost,
 - že nedojde k situacím charakteru zahlcení (druhá strana nereaguje, tak pošlu znovu ...) nebo „vyhladovění“ (druhá strana nereaguje, tak čekám dál, až zareaguje ...)
 - že někdo zneužije/naruší/“unese“ navazované spojení
 - že někdo nebude útočit přemírou žádostí o navázání spojení (DOS/DDOS útok)
 - že nebude docházet k útokům formou podvrženého spojení
 -
 - vše musí fungovat rychle a s minimální spotřebou zdrojů
 - příklad (WWW): každé kliknutí způsobí navázání nejméně jednoho spojení, někdy i mnoha ...

navazování spojení v protokolu TCP

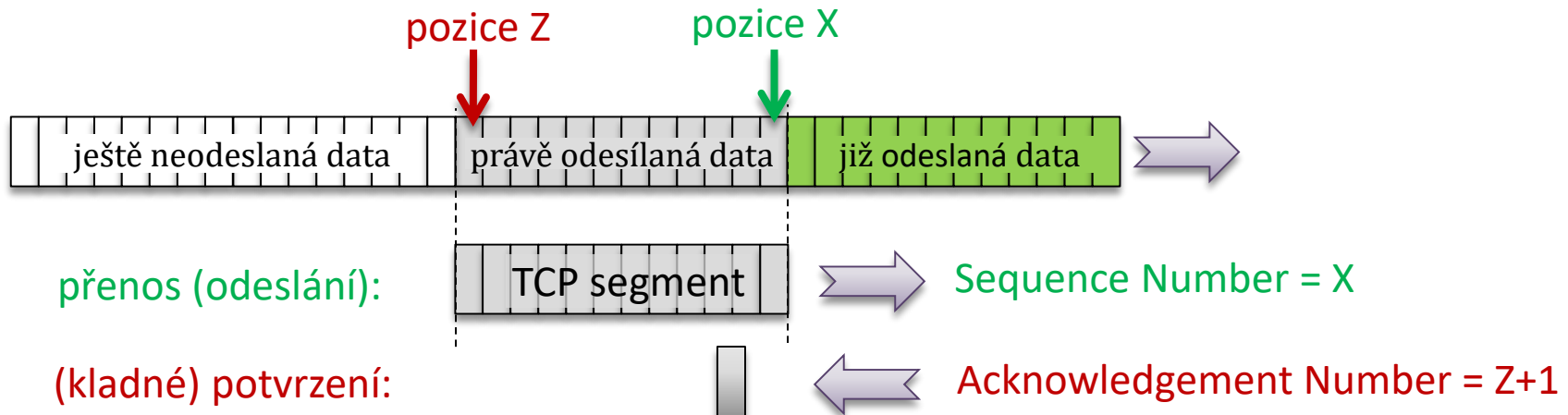
- **protokol TCP: navazování spojení má 3 fáze (tzv. 3-way handshake)**
 - aby se obě strany mohly korektně dohodnout na navázání spojení
 - a aby se (lépe) předešlo nežádoucím situacím – zahlcení, vyhladovění
 - aby si obě strany stihly předat (a potvrdit) počáteční pozice v bytových proudech



- **přesto:**
 - hrozí nebezpečí zneužití (tzv. SYN flooding)
 - jedna strana neustále zahajuje 1. fázi navázání spojení, ale už nepokračuje v dalších fázích
 - což protistraně váže prostředky, alokované na navazování spojení

zajištění spolehlivosti (v TCP)

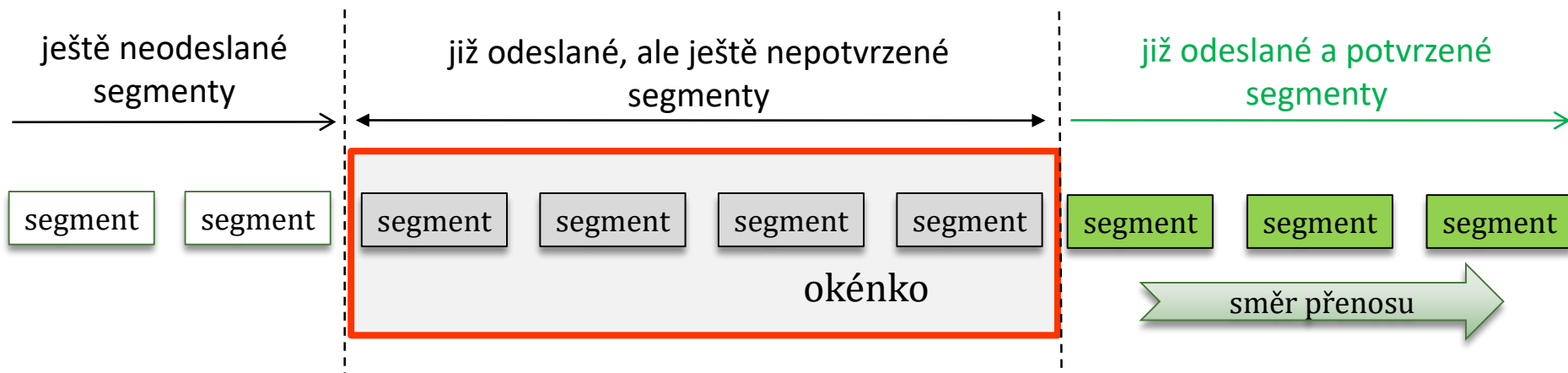
- **připomenutí: TCP funguje spolehlivě (zajišťuje spolehlivý přenos)**
 - používá kontinuálního potvrzování
 - díky tomu dokáže dobře fungovat i v sítích s velkou dobou obrátky
 - ale: nečísluje jednotlivé TCP segmenty
 - místo toho identifikuje data podle jejich pozice v bytovém proudu
- **při odesílání:**
 - říká: „*posílám data z proudu počínaje pozicí X*“ (Sequence Number)
- **při potvrzování:**
 - říká: „*přijal jsem v pořádku data až do pozice Z*“ (Acknowledgement Number)
 - přesněji: „*jako další očekávám data od pozice Z+1*“



- obdobně pro opačný směr přenosu (používá vlastní pozice X a Z)

spolehlivost a řízení toku

- **připomenutí (viz lekce 6): protokol TCP používá metodu okénka**
 - jak pro (kontinuální) potvrzování při zajišťování spolehlivosti, tak i pro řízení toku



- **představa:**
 - **okénko** udává, kolik dat ještě může odesílatel odeslat
 - v rámci kontinuálního potvrzování: ještě než dostane potvrzení o jejich doručení
 - v rámci řízení toku: aby nezahltl příjemce
 - **velikost okénka (spolu)určuje:**
 - odesílatel, podle toho, jak rychle (za jakou dobu) dostává zpět potvrzení
 - z toho odvozuje, jaká je doba obrátky (RTT), a podle ní stanovuje velikost okénka
 - příjemce, který odesílateli říká (inzeruje), kolik (dalších) dat je ještě schopen přijmout
 - tento údaj příjemce poskytuje v rámci potvrzení o přijetí dat, ve smyslu:
 - „přijal jsem data do pozice *X* včetně, a jsem schopen přijmout dalších *Y* dat“

řízení toku a předcházení zahlcení

• řízení toku:

- aby odesílatel nezahltl příjemce
 - předpoklad: přenosová síť mezi odesílatelem a příjemce je dostatečně průchodná (není „úzkým hrdlem“)



– princip řízení toku:

- příjemce diktuje odesílateli tempo odesílání dat

• dá se řešit:

– na nižších vrstvách

- na fyzické a linkové vrstvě
 - pomocí řídicích signálů na rozhraní

– na transportní vrstvě

- například metodou okénka

• v TCP/IP:

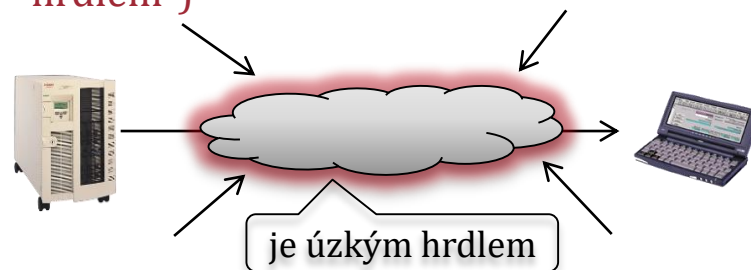
- protokoly IP ani UDP neřeší řízení toku
- TCP řídí tok pomocí metody okénka

konkrétní postupy
viz lekce č. 6

• předcházení zahlcení

- aby přenosová síť „stíhala“ přenášet data a nemusela je zahazovat

- předpoklad: příjemce dokáže přijímat data dostatečně rychle (není „úzkým hrdlem“)



– možnosti předcházení zahlcení:

- dopředné: generovaný datový tok se upravuje tak, aby „lépe prošel“
- zpětnovazební: odesílatel dostává zpětnou vazbu o stavu sítě a podle ní se řídí

• dá se řešit:

- na linkové až transportní vrstvě

• v TCP/IP:

- IP ani UDP neřeší, TCP řeší

best effort vs. QoS

- **připomenutí**

- „best effort“ znamená, že se všemi přenášenými daty se při přenosu nakládá stejně
 - nedělá se mezi nimi žádný rozdíl – ani v situaci, kdy „nelze všem plně vyhovět“
- negativní projevy (v situaci, kdy nestačí dostupná přenosová či výpočetní kapacita pro okamžité zpracování všech přenášených dat):
 - zpoždování přenosu některých dat, nepravidelnosti v jejich doručování (vyšší jitter)
 - až, v extrému: nutnost zahazování některých dat
- vadí to hlavně multimediálním datům
 - protože jejich příjemce je zpracovává průběžně

označováno též jako
„přístup hrubou silou“

přechází se na
rychlejší
přenosové cesty,
výkonnější
směrovače atd.

- **možnosti řešení:**

- a) zachování principu „best effort“, a posílení (předimenzování) kapacit

- záměr: posílit kapacity tak, aby nedocházelo (tak často) k situacím, kdy se přenosových či výpočetních kapacit nedostává a je nutné některá data zpoždovat či dokonce zahazovat
- v praxi: vychází to levněji a je to jednodušší než jiná řešení, která nahrazují princip best-effort
 - proto: v praxi je to nejčastější řešení !!!!

- b) zachování principu „best effort“, a nasazení „doplňkových opatření“

- jako je například technika client buffering

- c) nahrazení principu „best effort“ jiným řešením

- obecně: když nejde o „best effort“, jde o „podporu kvality služeb“ (**QoS, Quality of Service**)

požadavky služeb/aplikací na QoS

- různé druhy aplikací mají různé požadavky na podporu QoS
 - „počítačové služby“ (např. elektronická pošta, přenos souborů, web,)
 - vystačí zcela bez podpory QoS (vyhovuje jim best effort)
 - protože svá data nezpracovávají průběžně, ale „až když jsou všechna“
 - potřebují spolehlivost (data bez chyb a ztrát)
 - neinteraktivní multimediální služby (audio/video on demand, IPTV, internet. rádia):
 - vyžadují hlavně pravidelnost doručování svých dat (nízký jitter)
 - nevadí jim (tolik) občasná ztráta či poškození dat – dokáží je extrapolovat
 - lidské smysly nemusí být schopné zaznamenat výpadek/poškození
 - nevadí jim (tolik) delší doba přenosu (vyšší latence)
 - interaktivní multimediální služby (VOIP, videokonference,)
 - vyžadují pravidelnost doručování (nízký jitter) i krátkou dobu přenosu (nízkou latenci)
 - nevadí jim (tolik) občasná ztráta či poškození dat

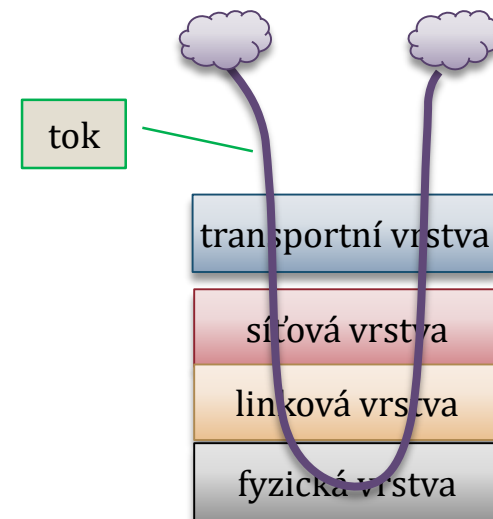
požadavek na	spolehlivost	malé zpoždění (nízká latence)	pravidelnost (nízký jitter)	přenosovou kapacitu
email	max.	min.	min.	min.
přenos souborů	max.	min.	min.	medium
www	max.	medium	min.	medium
remote login	max.	medium	medium	min.
audio on demand	min.	min.	max.	medium
video on demand	min.	min.	max.	max.
IP telefonie	min.	max.	max.	min.
videokonference	min.	max.	max.	max.

podpora kvality služeb (QoS)

- **obecně jde o jakékoli jiné řešení než „best effort“**
 - kdy je nějak rozlišováno podle toho, o jaká data při přenosu jde
- **podpora kvality služeb (QoS) může být založena na různých principech**
 - ale hlavní jsou dva:
- **upřednostnění/prioritizace:**
 - zavede se několik úrovní priority
 - každý přenos (každý blok dat) se přihlásí k určité úrovni priority
 - a podle toho je s ním také nakládáno
 - je to jen „relativní“ řešení
 - neskýtá záruku požadovaného zacházení
 - pokud by se všechna data přihlásila ke stejné úrovni priority, byl by z toho znovu princip best effort
- **vyhrazení/garance**
 - každý přenos (blok dat) si řekne, jaké podmínky pro svůj přenos požaduje
 - jaké parametry, v jakém rozsahu,
 - např. jakou požaduje přenosovou rychlost, latenci, pravidelnost doručování (jiter), ztrátovost, ...
 - zda požaduje nějaký rozsah (MIN-MAX)
 -
 - přenosová síť zjistí, zda požadavky může splnit, a podle toho je buď akceptuje (a následně poskytne), nebo požadavek odmítne
 - je to „absolutní“ řešení
 - které garantuje splnění požadavků na podporu QoS

na které vrstvě řešit podporu QoS?

- **z hlediska využití a specifikace požadavků – na transportní vrstvě (ev. výše)**
 - podpora QoS je obvykle požadována pro potřeby komunikace jednotlivých entit
 - nikoli pro veškerou komunikaci z/do nějakého uzlu
 - různé entity (zajišťující různé služby) mohou mít jiné požadavky na podporu QoS
 - aplikační entity, zajišťující poštovní služby či přenos souborů apod., nepotřebují QoS
 - aplikační entity, zajišťující neinteraktivní multimediální služby (např. distribuce videa a audia, IPTV, ...), požadují garantovanou rychlost/kapacitu a pravidelnost doručování (nízký jitter)
 - entity, zajišťující interaktivní služby (např. VOIP), požadují navíc ještě malé zpoždění (latenci)
 - např. pro kvalitní VOIP je vhodný $RTT < 250$ ms
 - rozlišení entit je možné nejdříve na transportní vrstvě
 - stejně jako specifikace jejich požadavků na podporu QoS
- **z hlediska implementace – na síťové vrstvě**
 - pokud síťová vrstva funguje na principu best effort, musí být její fungování vhodně pozměněno
 - a to ve všech uzlech (směrovačích) na celé trase přenosu
 - jinak není možná podpora QoS !!!
- **„nativní“ podpora QoS – (může být) na linkové vrstvě**
 - některé přenosové technologie podporují QoS
 - například: ATM, Frame Relay, ...



řešení QoS v TCP/IP

- **původně:**

- v TCP/IP původně nebyla žádná podpora QoS – vše jen na principu best effort
 - podpora QoS se zavádí dodatečně, v praxi nepříliš úspěšně
 - aby to mělo smysl, musí být nasazeno na všech uzlech „po cestě“ (lze jen v privátních sítích)

- **nově:**

- existují dvě ucelená řešení pro dodatečné přidání podpory QoS
 - jedno na principu prioritizace (DIFFSERV), druhé na principu rezervace (INTSERV)
- obě určitým způsobem modifikují fungování síťové vrstvy (protokolu IP a směrovačů)

- **DIFFSERV (DIFFerentiated SERVices)**

- na principu prioritizace
- zavede se několik tříd priority

- každý paket (IP datagram) si ve své hlavičce nese údaj o tom, ke které třídě se hlásí

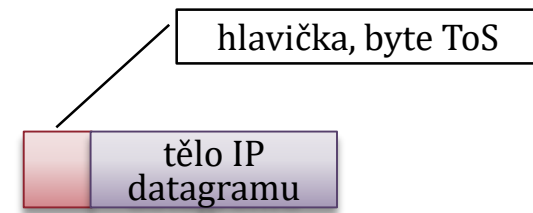
- ke specifikaci úrovně priority se využívá „zapomenutý“ byte QoS

- každý směrovač po cestě s paketem nakládá podle příslušné priority

- vyžaduje to změnu fungování protokolu IP

- podporu DIFFSERV v rámci všech směrovačů „po cestě“

- jakmile by jediný nepodporoval, eliminovalo by to celkový efekt



nepodporuje DIFFSERV:
datové pakety různých
priorit jsou sloučeny do
jediného toku (jediné
priority)



řešení QoS v TCP/IP

• INTSERV (INTEgrated SERVices)

– na principu garance

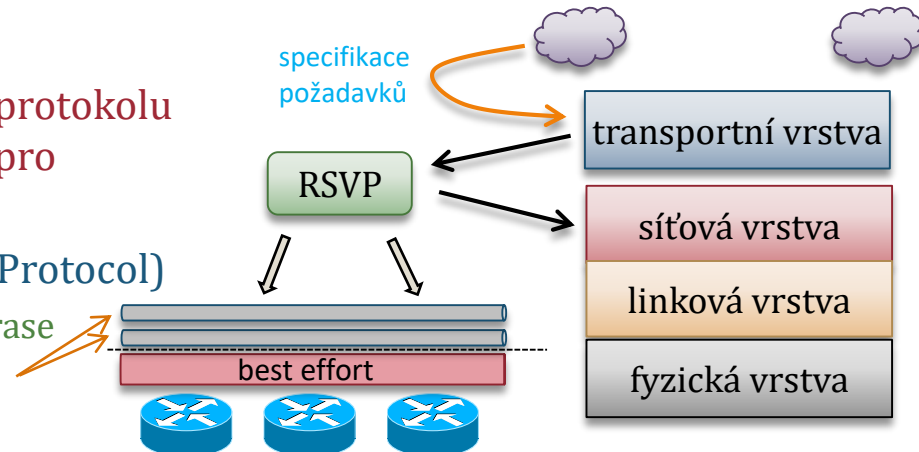
- díky rezervaci zdrojů na síťové vrstvě
 - v zásadě se protokolu IP „odejmou“ určité zdroje (přenosová i výpočetní kapacita)
 - které by protokol IP jinak „spotřeboval“ (využil) na principu best effort
 - a tyto zdroje se vyčlení pro přenosy s podporou QoS
 - v zásadě jde o přechod z principu přepojování paketů na princip přepojování okruhů
 - nejsnáze se realizuje pro spojovaný způsob přenosu
 - zdroje jsou vyhrazeny pro konkrétní spojení (které pak má vlastnosti vyhrazeného okruhu)

• na transportní vrstvě

- se specifikují požadavky na konkrétní formu/míru podpory QoS
 - a z toho vyjdou určité požadavky na rezervaci (vyčlenění) zdrojů na úrovni síťové vrstvy
 - řeší se v okamžiku navazování spojení na transportní vrstvě
 - pokud potřebné zdroje nejsou (na síťové vrstvě) k dispozici, spojení není navázáno

– nutný předpoklad:

- musí existovat možnost/mechanismus, jak protokolu IP „odejmout“ potřebné zdroje a ty přidělit pro přenosy s podporou QoS
 - to zajišťuje protokol RSVP (ReSerVation Protocol)
 - představa: projde všechny směrovače na trase přenosu a „sjedná“ s nimi vyčlenění zdrojů



doplňkové opatření: client buffering

- u jednosměrných (neinteraktivních) multimédií – například u přenosu videa – lze „obnovovat pravidelnost“ vhodným bufferováním u příjemce

- princip:

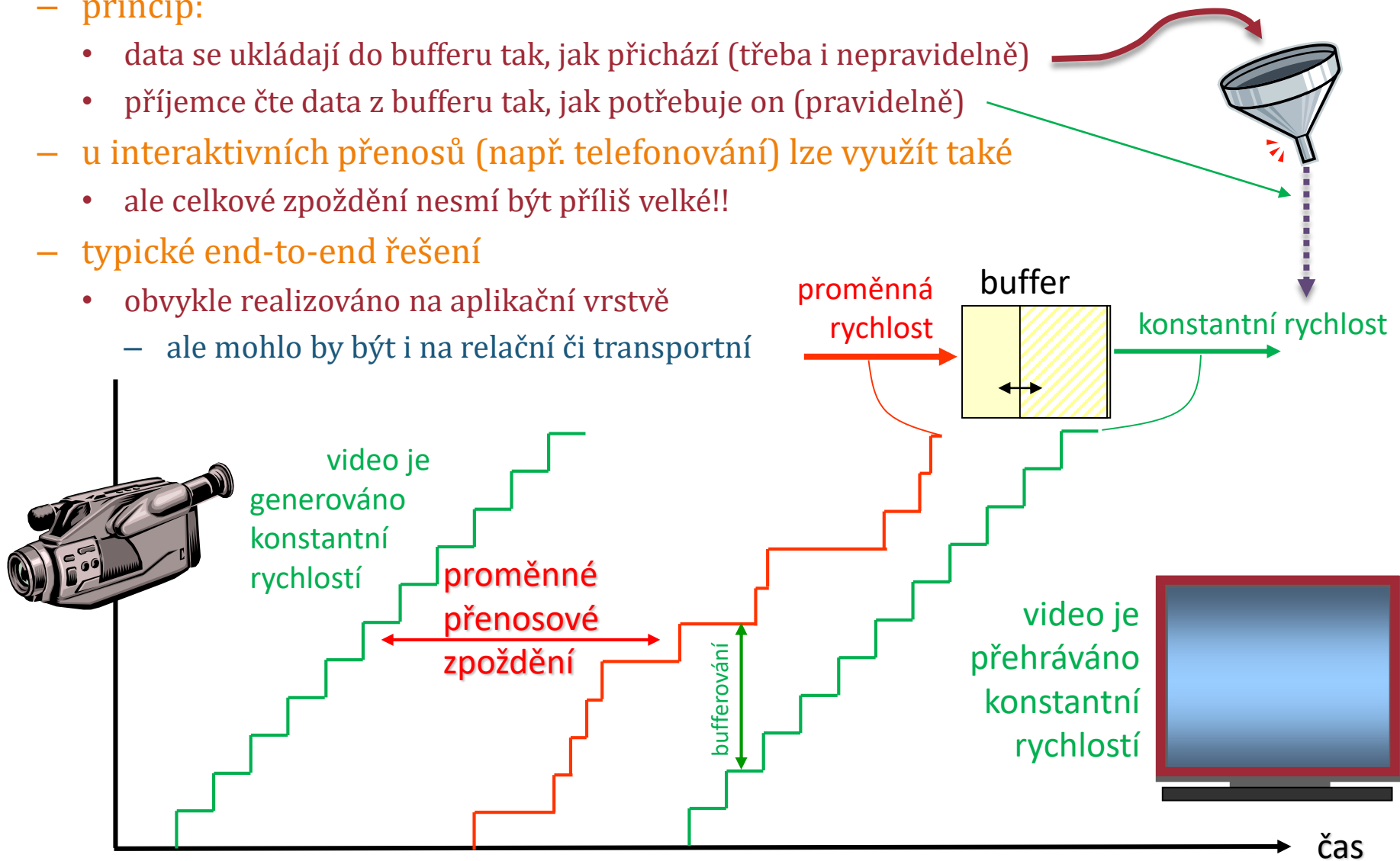
- data se ukládají do bufferu tak, jak přichází (třeba i nepravidelně)
- příjemce čte data z bufferu tak, jak potřebuje on (pravidelně)

- u interaktivních přenosů (např. telefonování) lze využít také

- ale celkové zpoždění nesmí být příliš velké!!

- typické end-to-end řešení

- obvykle realizováno na aplikační vrstvě
 - ale mohlo by být i na relační či transportní



protokol RTP (v TCP/IP)

• otázka:

– jaký transportní protokol mají používat multimediální služby/aplikace?

- typicky pracují s proudy dat (streamy) a mohou chtít využívat client buffering

• možnosti

– používat protokol **TCP** (méně časté)

- pracuje s proudy dat (vytváří iluzi datového proudu)
- je spolehlivý – a zajišťováním spolehlivosti způsobuje nepravidelnosti v doručování dat

– což multimediálním aplikacím obvykle vadí více, než případné poškození (či ztráta) dat

– používat protokol **UDP** (častější)

- nezajišťuje spolehlivost – nezvyšuje nepravidelnost v doručování dat
- nepracuje s proudy dat (ale s bloky – přenáší data po blocích)
- příjemce a odesílatel si musí zajistit vše potřebné sám
 - emulaci proudů, jejich identifikaci a popis, vyznačení času odeslání, ...

– používat protokol **RTP** (častější)

- Real Time Protocol
- je „nadstavbou“ nad protokolem UDP
 - aby si příjemce a odesílatel nemusel zajišťovat „vše potřebné“ pro přenos multimediálních dat sám, je vše realizováno samostatným protokolem
- "balí" jednotlivé části multimediálních dat do vlastních bloků (paketů)
 - a ty vkládá do UDP paketů
- připojuje informace
 - o typu multimediálního obsahu
 - o pořadí paketu
 - o čase vzniku dat (timestamp)
 - kdy přesně data vznikla, tím usnadňuje jejich bufferování na straně klienta
 - o konkrétním streamu (proudu)

