5. Transportní vrstva

PB156cv: Počítačové sítě

Miloš Liška

Podle přednášek z PB156

Fakulta informatiky Masarykovy univerzity

jaro 2021

Struktura přednášky

Struktura přednášky

L4. Transportní vrstva – Přehled

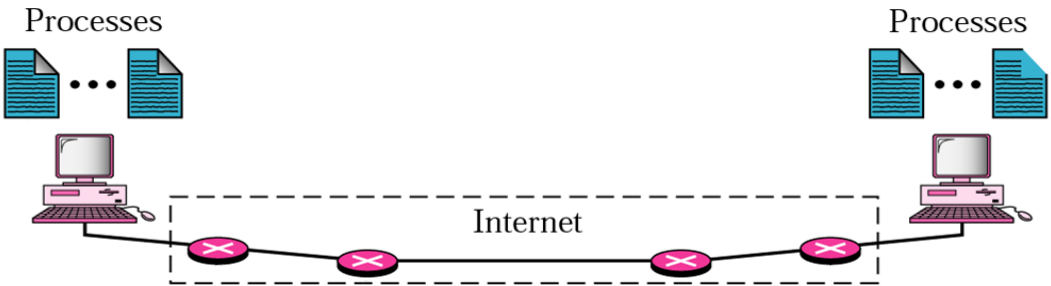
Proč nestačí L3?

- nemožnost identifikovat aplikaci, které jsou data určena
  - na každém uzlu by tak mohla běžet maximálně jedna aplikace
- neřeší defekty sítě (ztrátu/znásobení datagramu, zahlcení sítě, atp.)

Co nás nyní čeká...

- představení L4, poskytované služby
- mechanismy zajištění spolehlivého přenosu
- protokoly UDP, TCP

L4 z pohledu sítě – kde se pohybujeme?



- komunikace konkrétních aplikací (identifikovány transportní vrstvou) na konkrétních uzlech sítě (identifikovány síťovou vrstvou)
  - na uzlech tak může běžet více služeb
- možnosti zajištění spolehlivého přenosu nad nespolehlivou (best-effort) IP sítí

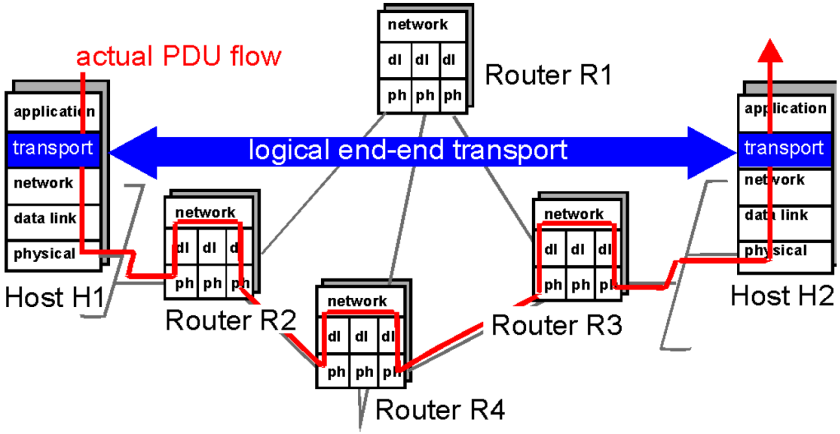
Struktura přednášky

Úvod I.

transportní vrstva:

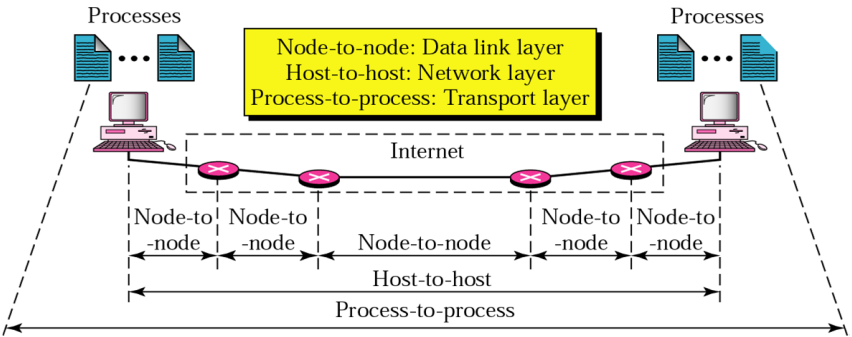
- poskytuje služby pro *aplikační vrstvu*:
  - přijímá data odesílací aplikace, které transformuje do *segmentů*
  - přijaté segmenty pak předává cílové aplikaci
- ve spolupráci se síťovou vrstvou zajišťuje doručení dat (segmentů) mezi komunikujícími *aplikacemi/procesy*
  - s případným zajištěním spolehlivosti přenosu
  - poskytuje jim logický komunikační kanál
    - iluze fyzického propojení (přímého komunikačního kanálu)
  - tzv. *process-to-process delivery*
- nejnižší vrstva poskytující tzv. *end-to-end* služby
  - hlavičky generované na straně odesílatele jsou interpretovány „jen“ na straně příjemce
  - směrovače vidí data transportní vrstvy jako payload přenášených paketů

Úvod II.



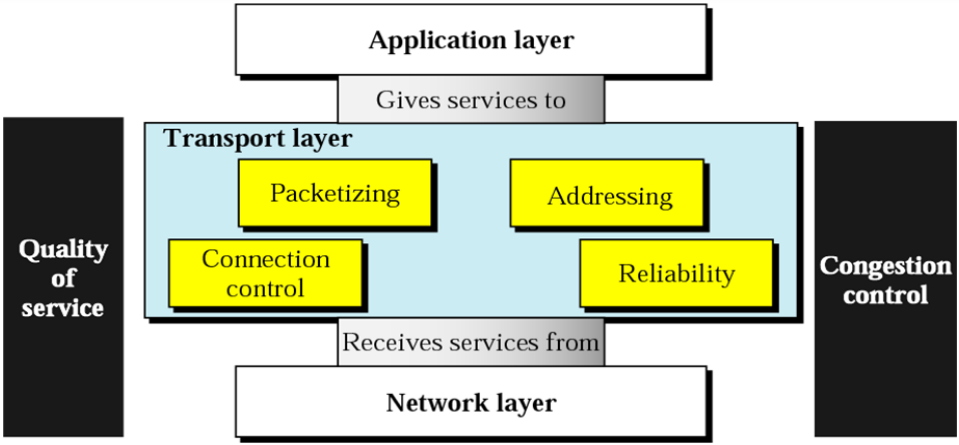
Obrázek: Ilustrace end-to-end služeb poskytovaných transportní vrstvou.

Úvod III.



Obrázek: Formy komunikace.

Struktura přednášky



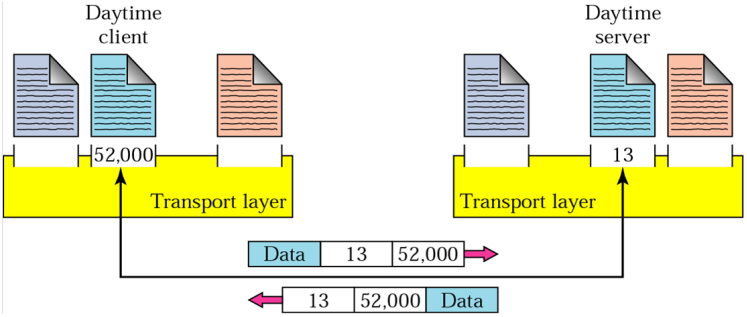
Obrázek: Ilustrace služeb transportní vrstvy.

Služby

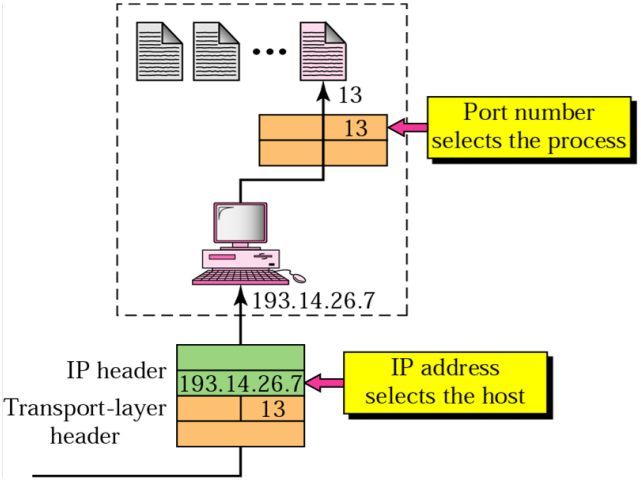
- *Tvorba paketů (Packetizing)*
  - aplikací zaslaná data transformována na pakety (s přidanou transportní hlavičkou)
- *Řízení spojení (Connection Control)*
  - *spojované (connection-oriented)* a *nespojované (connectionless)* služby
- *Adresace (Addressing)*
  - adresy entit transportní vrstvy (= síťových aplikací/služeb) – tzv. *porty*
  - pakety obsahují zdrojový a cílový port (identifikaci zdrojové a cílové aplikace)
    - aplikace tak jsou v síti jedinečně identifikovány dvojicí *IP_adresa:port*
- *Zajištění spolehlivosti přenosu (Reliability)*
  - *řízení toku (Flow Control)* a *řízení chyb (Error Control)*
    - na nižších vrstvách poskytováno node-to-node, zde *end-to-end*
  - zajištění spolehlivosti nad *best-effort* službou (IP)
- *Řízení zahlcení sítě (Congestion Control) a zajištění kvality služby (Quality of Service, QoS)*

Adresace na L4 I.

- adresy na L4 – *čísla portů (ports, port numbers)*
  - ≈ adresy služeb
  - identifikují odesílací aplikaci na zdrojovém uzlu (identifikován IP adresou)
  - identifikují přijímající aplikaci na cílovém uzlu (identifikován IP adresou)
- identifikace portu *16bitovým číslem*
  - rozsah 0 – 65535



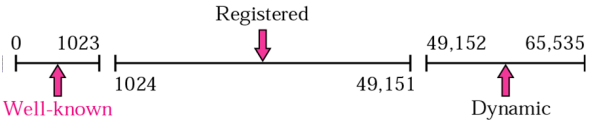
Adresace na L4 II.



Obrázek: Doručení dat cílové aplikaci – IP adresa a port.

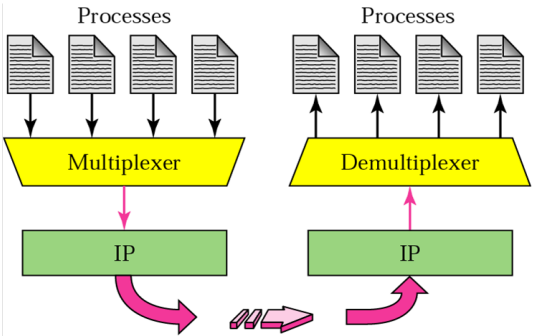
Adresace na L4 III.

- porty rozděleny do 3 tříd
  - rozděleno organizací *IANA (Internet Assigned Number Authority)*
- třídy:
  - *well-known („dobře známé“) porty*
    - rozsah 0 – 1023
    - identifikují známou konkrétní službu
    - přidělovány organizací IANA
  - *registrované porty*
    - rozsah 1024 – 49151
    - volně využitelné porty, nejsou přidělovány organizací IANA
    - lze je však u organizace IANA zaregistrovat (zamezení duplikací)
  - *dynamické porty*
    - rozsah 49152 – 65535
    - dynamicky přidělované porty, využity zejména jako zdrojové porty odesílacích aplikací



Adresace na L4 – Multiplexing vs. Demultiplexing

- mechanismus adresace na L4 představuje formu *multiplexingu* a *demultiplexingu*
  - na odesílací straně mnoho aplikací a jeden transportní protokol – *multiplexing*
    - odesílací aplikace identifikována zdrojovým portem
  - na přijímací straně jeden transportní protokol, výběr vhodné aplikace pro doručení – *demultiplexing*
    - přijímající aplikace identifikována cílovým portem



Řízení spojení – spojované vs. nespojované L4 služby

Spojované služby

- na začátku přenosu ustaveno spojení (udržováno po celou dobu přenosu dat)
- pakety jsou číslovány
  - jejich doručení/nedoručení je explicitně potvrzováno

Nespojované služby

- pakety zasílány cílové aplikaci bez ustaveného spojení
- pakety nejsou číslovány ($\Rightarrow$ nejsou ani potvrzovány)
  - mohou se ztratit, dorazit se zpožděním, dorazit mimo pořadí, atp.

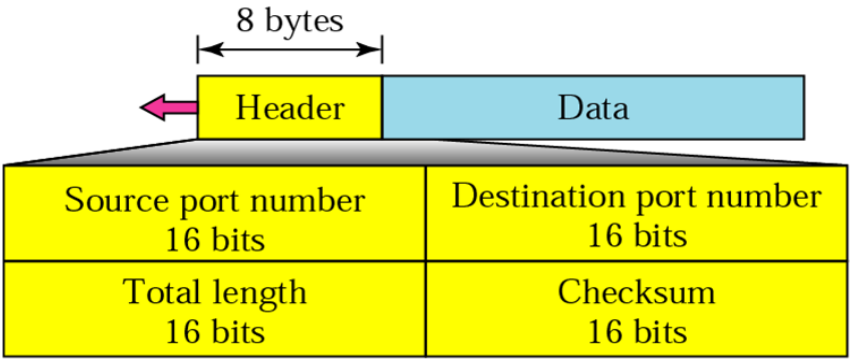
Struktura přednášky

UDP protokol

User Datagram Protocol (UDP)

- nejjednodušší transportní protokol poskytující **nespojovanou** a **nespolehlivou (= nezajištěnou)** službu
  - poskytuje *best-effort* službu
  - ke službám IP vrstvy přidává pouze process-to-process komunikaci a jednoduchou kontrolu chyb
  - případné zajištění spolehlivosti přenosu je na aplikaci
- *hlavní přednosti:* jednoduchost, minimální režie
  - žádná nutnost ustavení spojení (přináší zpoždění na začátku přenosu)
  - žádná nutnost uchovávání stavových informací na komunikujících stranách
  - malá hlavička

UDP protokol – hlavička paketů



- **zdrojový port (source port)** – identifikace odesílací služby/aplikace
- **cílový port (destination port)** – identifikace přijímající služby/aplikace
- **délka UDP paketu (length)** – celková délka UDP paketu
- **kontrolní součet (checksum)** – kontrolní součet UDP paketu (hlavička + data)

UDP protokol – odesílání dat z pohledu aplikace - vliv MTU

- Maximum Transfer Unit (MTU) – maximální velikost dat, které lze přenést využitým L2 protokolem
  - určuje maximální velikost přenositelného IP datagramu (Total size)
- Ethernet: 1500 B
- Jumbo frames: na většině HW až 9216 B
- Proces $\rightarrow$ snd buffer $\rightarrow$ packet payload + hlavičky $\rightarrow$ IP datagram
  - všech protokolů použitých na jednotlivých vrstvách ISO/OSI modelu

UDP protokol – rychlost odesílání paketů

- Pipeline: userspace buffer (UDP paket + UDP hlavička) → (socket) kernel buffer → NIC (typicky jediná fronta) → síť
- Rychlost odesílání ovlivňuje:
  - Rychlost CPU/RAM
  - Zpracování přerušení
    - MTU
    - interrupt load balancing
    - interrupt coalescing
  - Ethernet flow control (pause frames)
- Síťová karta odesílá pakety z fronty jak nejrychleji dovede (1 Gbps, 10 Gbps, 40 Gbps, 100 Gbps apod.)
- Pokud odesílání UDP paketů nic nebrzdí, jsou odesílány rychlostí danou wire speed

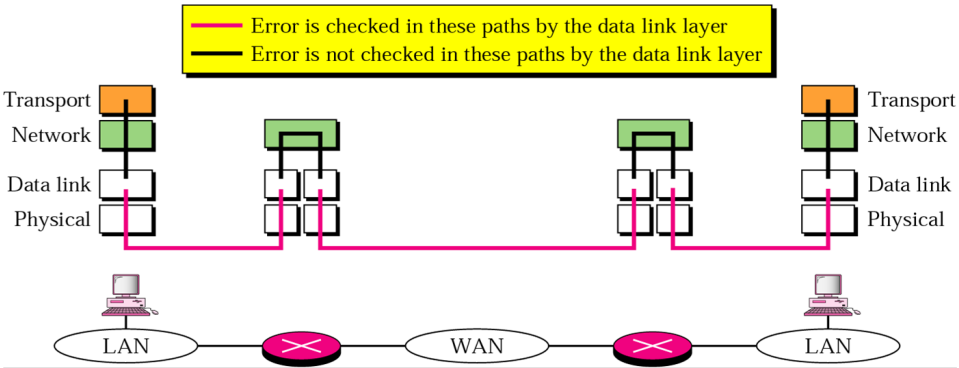
Struktura přednášky

Mechanismy zajištění spolehlivého přenosu I.

- *otázka:* k čemu je řízení chyb na L4, když je toto poskytováno L2 vrstvou?

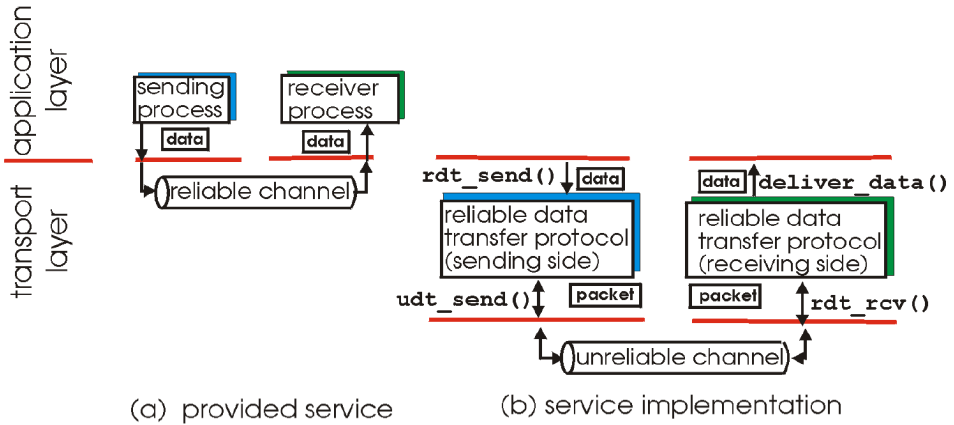
Mechanismy zajištění spolehlivého přenosu I.

- *otázka:* k čemu je řízení chyb na L4, když je toto poskytováno L2 vrstvou?
  - L2 poskytuje řízení chyb vždy pouze mezi dvěma uzly na cestě, ne mezi koncovými stanicemi



Mechanismy zajištění spolehlivého přenosu II.

- transportní protokoly tak *mohou* zajišťovat spolehlivý přenos nad nespolehlivou (best-effort) IP službou



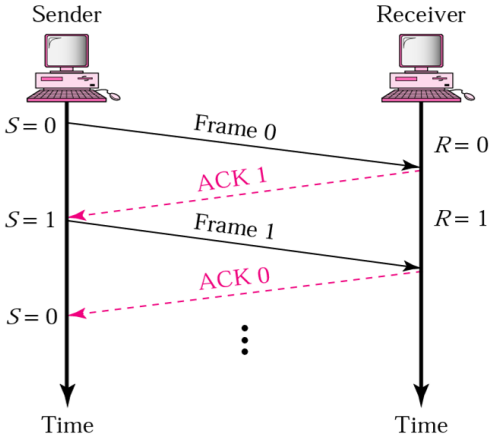
Mechanismy zajištění spolehlivého přenosu III.

- spolehlivost přenosu zajištěna mechanismem *potvrzování (acknowledgement)*
  - pakety číslovány tzv. *sekvenčními čísly (Sequence Numbers, SEQ)*
  - *pozitivní potvrzování (positive acknowledgement)*
    - potvrzení úspěšného přijetí paketu
    - ala „doručeno v pořádku“
  - *negativní potvrzování (negative acknowledgement)*
    - informace o neúspěšném přijetí/ztrátě datagramu
    - ala „prosím, zopakuj“
- v případě výskytu chyby jsou data opětovně přeposílána
  - mechanismy *ARQ (Automatic Repeat reQuest)*
    - *Stop-and-Wait ARQ*
    - *Go-Back-N ARQ*
    - *Selective-Repeat ARQ*
  - nutnost vypořádat se s **duplicitami!**

Stop-and-Wait ARQ I.

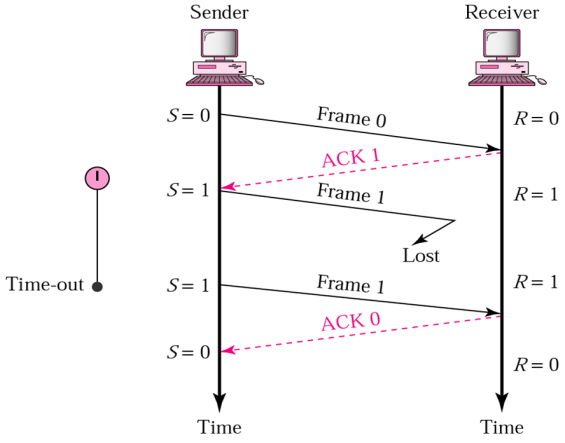
- nejjednodušší mechanismus řízení toku a řízení chyb
- odesílací strana po odeslání paketu vyčkává na jeho potvrzení
  - aniž by odesílala další
  - po uplynutí definované doby (*timeout*) je paket pokládán za ztracený
    - následuje znovuposlání
- pakety číslovány střídavě 0 a 1
  - potvrzení paketu = ACK s číslem následujícího (očekávaného) paketu
- v případě poškození paketu (vadný kontrolní součet) jej příjemce zahazuje a vyčkává na znovuposlání
  - nezasílá se žádné negativní potvrzení

Stop-and-Wait ARQ II.



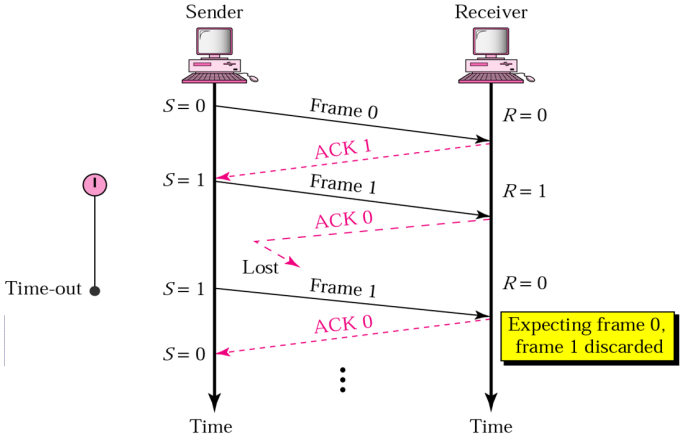
Obrázek: Stop-and-Wait ARQ: bezeztrátový přenos

Stop-and-Wait ARQ III.



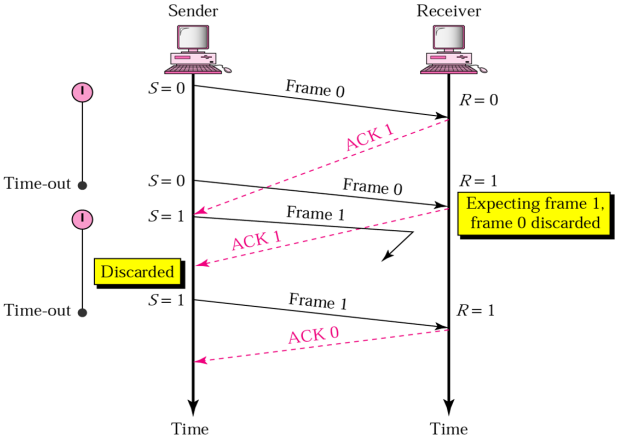
Obrázek: Stop-and-Wait ARQ: ztráta paketu

Stop-and-Wait ARQ IV.



Obrázek: Stop-and-Wait ARQ: ztráta potvrzení

Stop-and-Wait ARQ V.

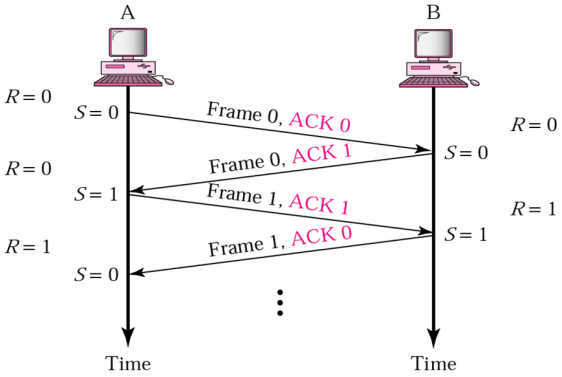


Obrázek: Stop-and-Wait ARQ: opožděné potvrzení

Stop-and-Wait ARQ VI.

V případě obousměrného přenosu lze využít mechanismus **Piggybacking**

- kombinace datového paketu s potvrzením
  - místo dvou samostatných paketů (potvrzení, data) se tak zasílá právě jeden



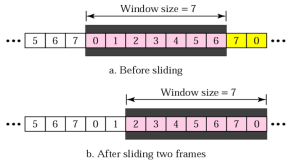
Go-Back-N ARQ I.

- problém Stop-and-Wait ARQ: do sítě lze v jakémkoliv okamžiku vyslat pouze jeden paket ⇒ degradace výkonu
- vylepšení mechanismu Stop-and-Wait
  - zasílání více paketů bez vyčkávání na jejich potvrzení
  - cílem je vyšší efektivita přenosu
- pakety číslovány postupně se zvyšujícími sekvenčními čísly
  - v případě dosažení horní hranice se začíná znovu od začátku
  - např. $0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, ...$
- potvrzení paketu = ACK se sekvenčním číslem následujícího (očekávaného) paketu
  - využití tzv. *kumulativních potvrzení*
  - v případě obousměrné komunikace možno využít *piggybacking*
- informace o odeslaných/přijatých paketech uchovávána za pomoci mechanismu tzv. *plovoucího okna (sliding window)*
  - udržováno jak na straně odesílatele, tak na straně příjemce
- varianta Go-Back-N ARQ využita v protokolu TCP (viz později)

Go-Back-N ARQ II. – *Sender Window* vs. *Receiver Window*

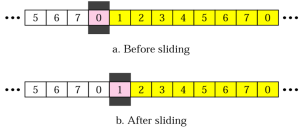
Okno odesílatele (Sender Window)

- maximální velikost $2^m - 1$ (m = počet bitů pro uchování SEQ)
  - důvody viz dále

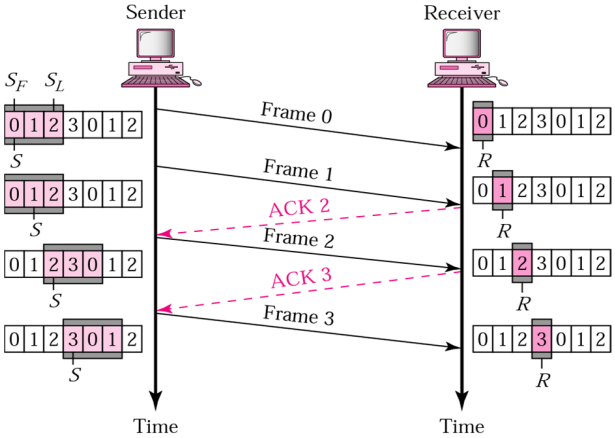


Okno příjemce (Receiver Window)

- velikost v případě Go-Back-N ARQ vždy 1 (vždy se očekává pouze určitý paket)

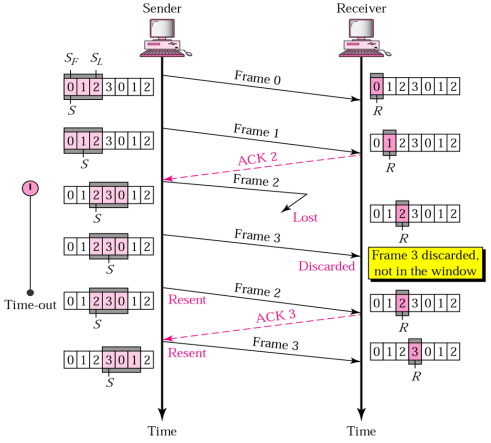


Go-Back-N ARQ III.



Obrázek: Go-Back-N ARQ: bezeztrátový přenos

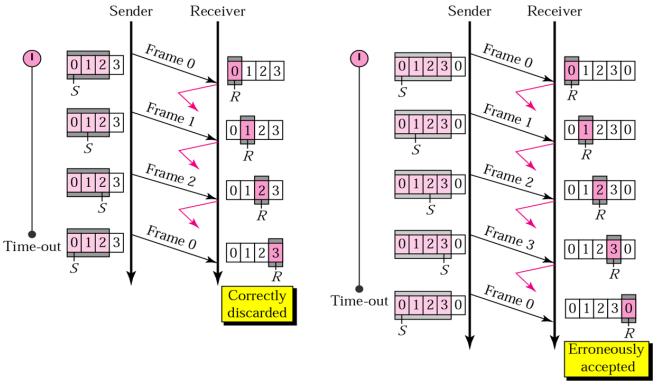
Go-Back-N ARQ IV.



Obrázek: Go-Back-N ARQ: ztráta paketu

Go-Back-N ARQ V. – omezení maximální velikosti okna

Okno odesílatele musí být menší než 2^m (m je počet bitů pro uchování SEQ) kvůli správné detekci duplicit!



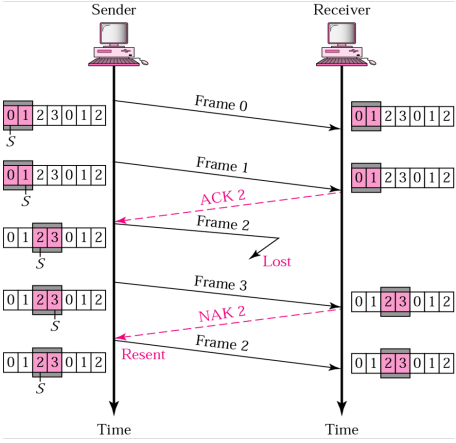
a. Window size < 2^m

b. Window size = 2^m

Selective-Repeat ARQ I.

- problém Go-Back-N ARQ: neefektivní pro vysoce ztrátové linky
  - vyšší ztrátovost ⇒ vyšší procento paketů došlých mimo pořadí (*out-of-order*)
  - Go-Back-N ARQ pakety mimo pořadí zahazuje
    - neefektivní, již došlé pakety musí být znovu zasílány
- pakety opět číslovány postupně se zvyšujícími sekvenčními čísly
- rozšíření Go-Back-N ARQ v oblasti okna příjemce
  - místo 1 paketu jich může pojmout více
  - ⇒ out-of-order pakety na straně příjemce bufferovány
- potvrzení paketu = ACK se sekvenčním číslem následujícího (očekávaného) paketu
  - opět využívá *kumulativních potvrzení*
  - v případě obousměrné komunikace možno využít *piggybacking*
- kromě pozitivních potvrzení využívá i negativních potvrzení
  - *Negative Acknowledgements* zasílány v případě detekce ztráty/porušení paketu

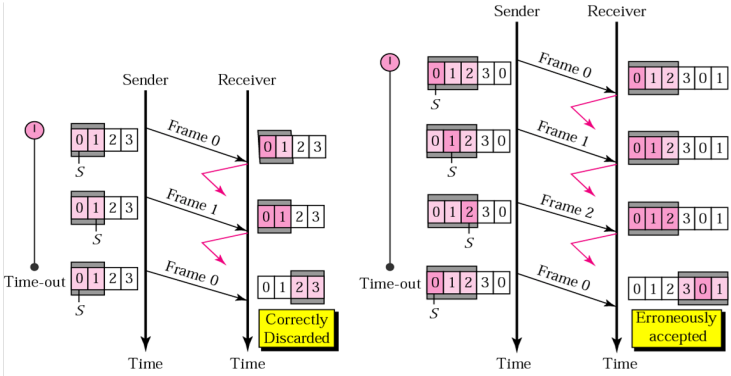
Selective-Repeat ARQ II.



Obrázek: Selective-Repeat ARQ: ztráta paketu

Selective-Repeat ARQ III. – omezení maximální velikosti okna

Okno odesílatele musí být menší nebo rovno 2^{m-1} (m je počet bitů pro uchování SEQ) kvůli správné detekci duplicit!



Struktura přednášky

TCP protokol

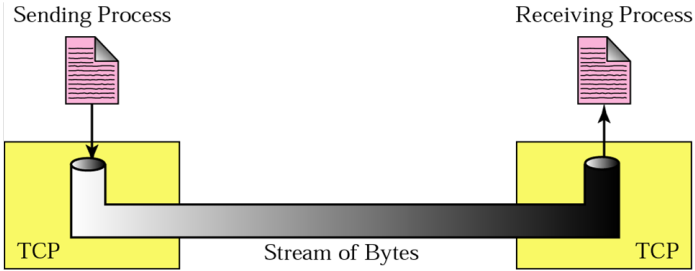
Transmission Control Protocol (TCP)

- transportní protokol poskytující **spojovanou** a plně **spolehlivou (= zajištěnou)** službu
  - pokud je to možné, odeslaná data budou přijímající aplikaci doručena kompletní a ve správném pořadí
  - oproti UDP orientován na přenos proudu bytů (UDP orientováno na přenos bloků dat)
- před začátkem přenosu nutnost ustavení *spojení* mezi odesílací a přijímající stranou
  - tzv. *handshake* před začátkem přenosu zahrnuje výměnu všech potřebných parametrů
  - spojení rozeznatelné jen na koncových uzlech (end-to-end služba)
    - směrovače tato spojení „nevidí“
  - ustavené spojení možno využít pro plně duplexní komunikaci
    - řídící data přibalována do dat jdoucích opačným směrem (piggybacking)
  - spojení může být pouze **dvoubodové (point-to-point)**
    - komunikace mezi více partnery (ala multicast) není podporována
- multiplexing/demultiplexing a detekce chyb stejné jako v UDP

TCP protokol – poskytované služby

Přenos proudu bytů

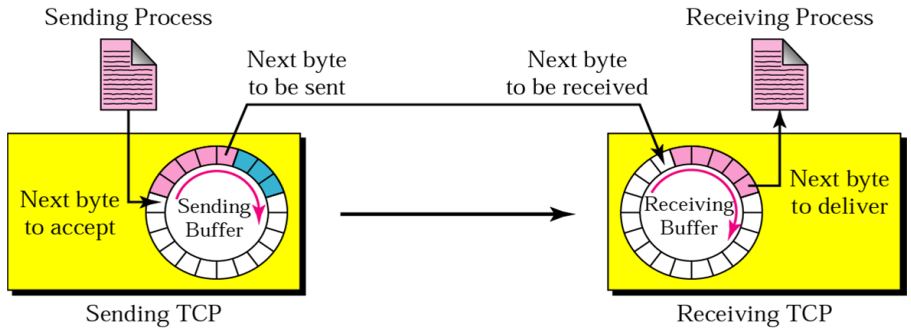
- přenos dat v rámci UDP:
  - aplikace předává bloky dat, které UDP opatřuje hlavičkou a předává síťovému protokolu (např. IP)
- přenos dat v rámci TCP:
  - aplikace předává TCP protokolu proud bytů, ktere TCP segmentuje, opatřuje hlavičkou a předává síťovému protokolu
  - aplikacím poskytuje iluzi roury, která přenáší jejich data



TCP protokol – poskytované služby

Odesílací a přijímající buffery

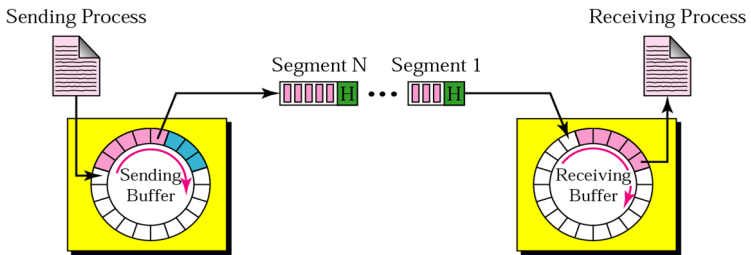
- aplikací předaná data nutno uchovávat v bufferech
  - nutnost vyrovnání rozdílných rychlostí komunikujících stran
    - rychlost odesílajícího a přijímajícího procesu nemusí být stejná
  - buffery navíc využity pro řízení toku a chyb (viz dále)



TCP protokol – poskytované služby

Segmentace dat

- aplikace TCP protokolu předává proud bytů
- síťová vrstva (IP protokol) očekává bloky dat
- ⇒ nutnost tvorby bloků dat (*segmentů*)
  - velikost segmentů omezena hodnotou *Maximum Segment Size (MSS)*
    - definováno implementací TCP / operačním systémem
    - identifikuje maximální velikost uživatelských dat v segmentu (**ne** velikost celého segmentu)
  - segmenty následně opatřeny TCP hlavičkou a předány síťovému protokolu



TCP protokol – poskytované služby

Segmentace dat – číslování segmentů

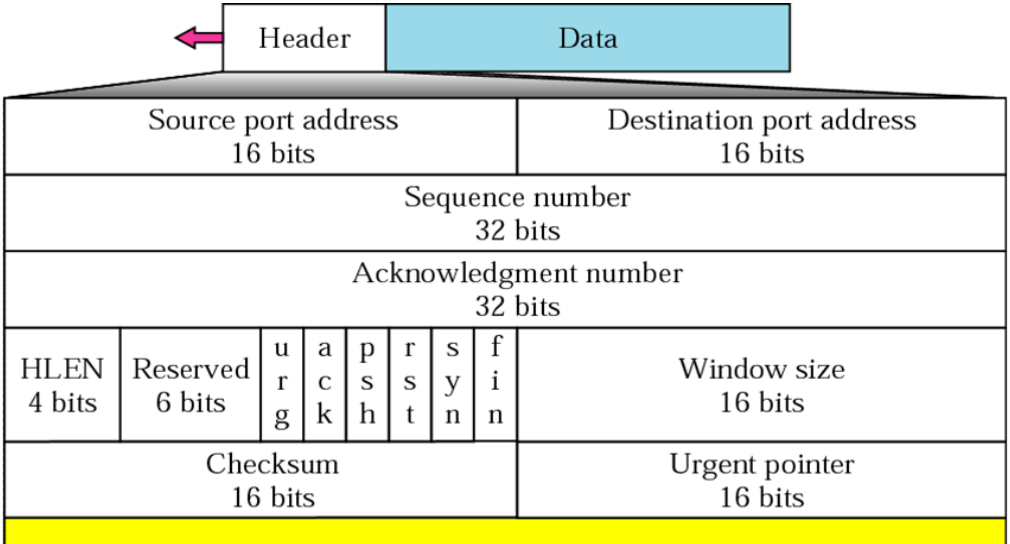
- číslovány nejsou bloky dat (segmenty), ale jednotlivé přenášené bajty
  - každý aplikací předaný bajt je opatřen číslem
    - začátek náhodně zvolený; inkrementováno po 1
- sekvenční číslo přenášeného TCP segmentu je pak číslo prvního bajtu přenášeného daným segmentem

Příklad: Přenos souboru o velikosti 6000 bajtů. První bajt očíslován jako 10010. Poslední segment přenáší 2000 bajtů, ostatní 1000 bajtů.

The following shows the sequence number for each segment:

Segment 1 ==> sequence number: 10,010 (range: 10,010 to 11,009)
Segment 2 ==> sequence number: 11,010 (range: 11,010 to 12,009)
Segment 3 ==> sequence number: 12,010 (range: 12,010 to 13,009)
Segment 4 ==> sequence number: 13,010 (range: 13,010 to 14,009)
Segment 5 ==> sequence number: 14,010 (range: 14,010 to 16,009)

TCP protokol – hlavička segmentů I.



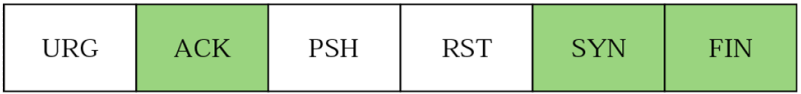
TCP protokol – hlavička segmentů II.

- **zdrojový port (source port)** – identifikace odesílací služby/aplikace
- **cílový port (destination port)** – identifikace přijímající služby/aplikace
- **sekvenční číslo (sequence number)** – sekvenční číslo segmentu
- **číslo potvrzovaného segmentu (acknowledgement number)**
  - číslo bajtu, který přijímající strana očekává jako následující
  - *piggybacking*
- **délka hlavičky (header length)** – délka TCP hlavičky ve 4B slovech
- **rezervovaná pole (reserved)**

TCP protokol – hlavička segmentů III.

- **řídící data (control)** – 6 bitů identifikujících nejrůznější řídící informace

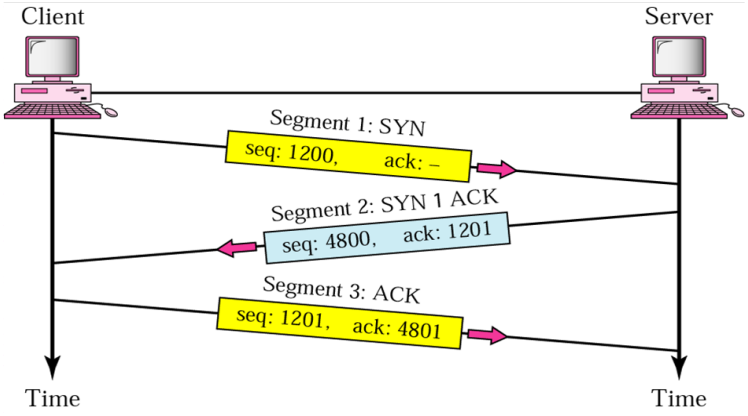
URG: Urgent pointer is valid	RST: Reset the connection
ACK: Acknowledgment is valid	SYN: Synchronize sequence numbers
PSH: Request for push	FIN: Terminate the connection



- **velikost okna (window size)** – velikost okna, které musí komunikující strana spravovat
  - určeno pro účely řízení toku (viz dále)
- **kontrolní součet (checksum)** – kontrolní součet TCP segmentu (hlavička + data)
- **urgentní data (urgent pointer)** – zasílání dat mimo pořadí
- **volby (options)**

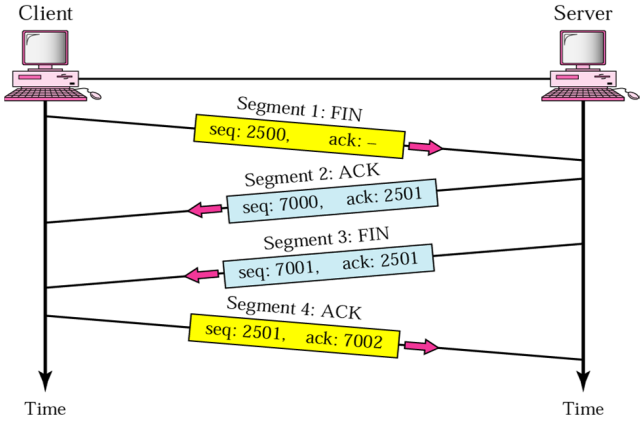
Správa spojení – ustavení spojení

- full-duplexní přenos ⇒ obě strany musí iniciovat spojení
- mechanismus známý jako **třícestný handshake (three-way handshake)**



Správa spojení – ukončení spojení

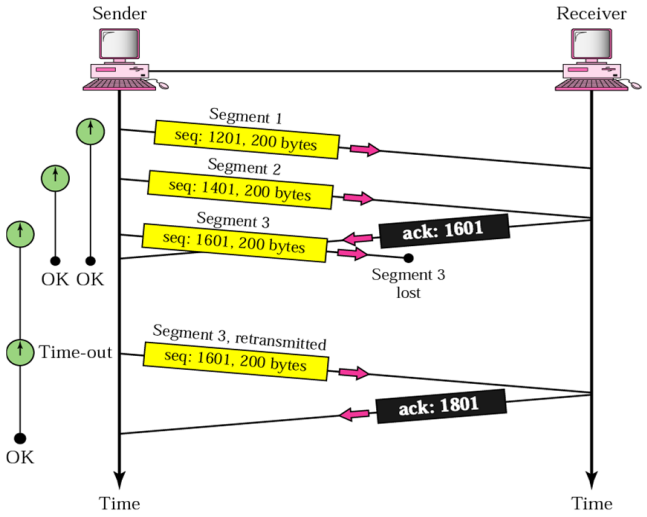
- iniciováno jednou z komunikujících stran
- spojení musí být uzavřeno oběma stranami



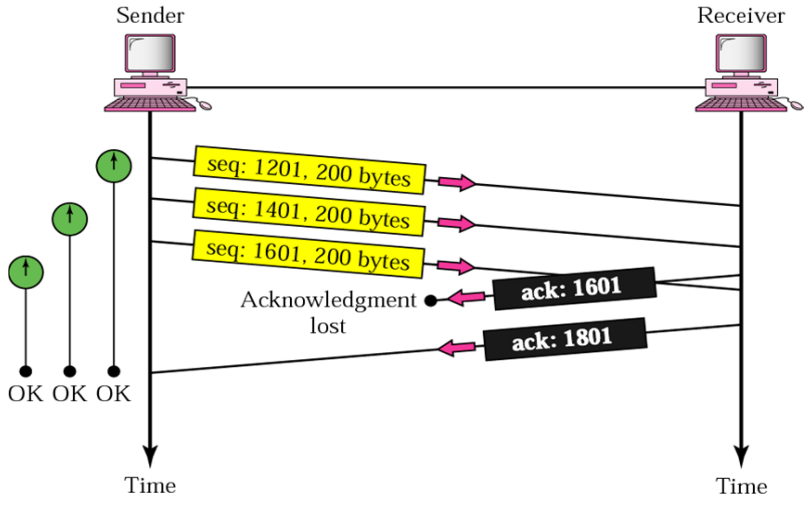
Řízení chyb (Error Control)

- během přenosu je nutno detekovat poškozené, ztracené, duplikované a out-of-order segmenty
- TCP mechanismy pro zajištění spolehlivého přenosu:
  - *kontrolní součty* – detekce poškozených segmentů
  - *potvrzování přijatých segmentů (acknowledgements)* – detekce ztracených (na straně příjemce), duplikovaných a out-of-order segmentů
    - zajištěno mechanismem pozitivního potvrzování (*positive acknowledgements*)
    - využito *kumulativní potvrzování*
  - *timeoutů* – detekce ztracených segmentů (na straně odesílatele)
- mechanismus přeposílání založen na Go-Back-N ARQ
  - *rozdíl:* buffer pro out-of-order segmenty na přijímající straně

Řízení chyb (Error Control) – Ztráta segmentu



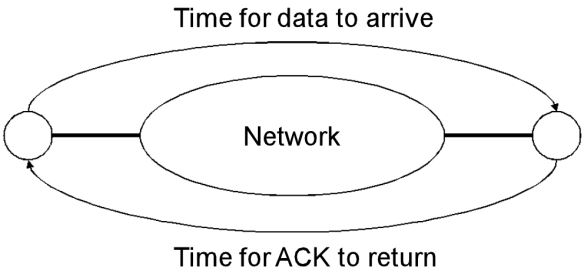
Řízení chyb (Error Control) – Ztráta potvrzení



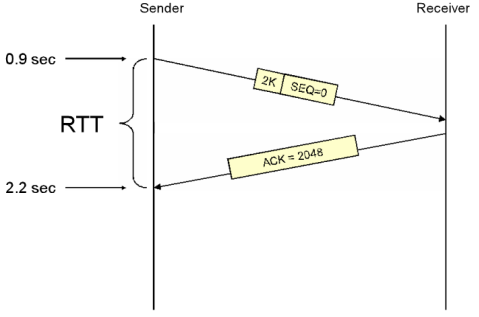
Řízení chyb (Error Control) – Timeouty I.

timeout – doba, po kterou se čeká na potvrzení odeslaného segmentu

- založeno na tzv. *Round-Trip Time (RTT)*
  - čas potřebný pro cestu segmentu od odesílatele k příjemci a zpět
  - typicky: $timeout = 2 \cdot RTT$



Řízení chyb (Error Control) – Timeouty II.



RTT = 2.2 sec - 0.9 sec. = 1.3 sec

RTT upravováno s využitím následující formule (vyhlazování abnormalit):

$$RTT_{new} = \alpha \cdot RTT_{old} + (1 - \alpha) \cdot RTT_{measured}$$

kde $\alpha = 0.875$ (typicky)

TCP mechanismy pro řízení množství zasílaných dat

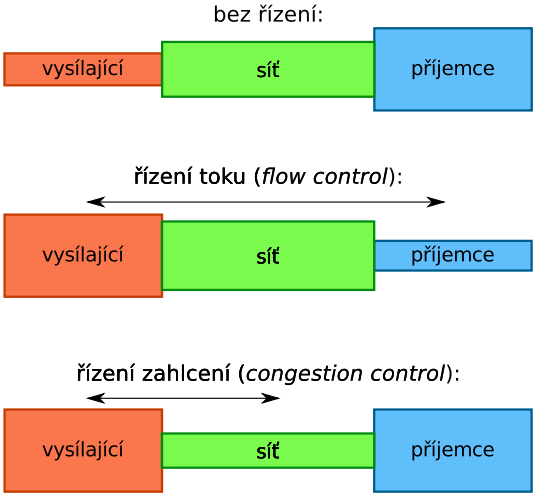
TCP řídí množství zasílaných dat tak, aby:

- *zabránilo zahlcení příjemce* = **řízení toku (Flow Control)**
- *zabránilo zahlcení sítě* = **řízení zahlcení (Congestion Control)**

Množství dat, které je možno zaslat do sítě je definováno:

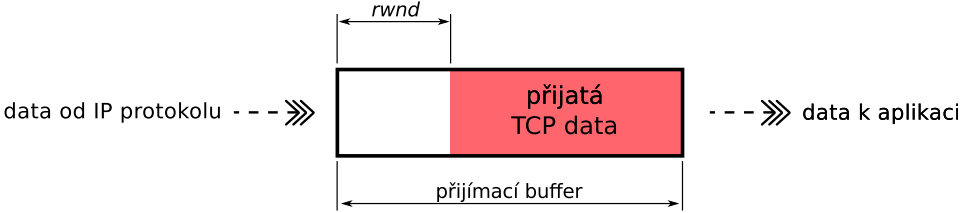
- velikostí okna příjemce (řízení toku)
- velikostí tzv. *okna zahlcení (congestion window)* (řízení zahlcení)
  - na straně odesílatele
- množství skutečně vysílaných dat ohraničeno **menší hodnotou z obou jmenovaných**

Flow Control vs. Congestion Control



Řízení toku (Flow Control)

- mechanismus pro zábranu zahlcení přijímající strany
- **explicitní zpětná vazba od příjemce**
  - příjemce informuje odesílatele o stavu svého přijímacího bufferu (*receiver window*, *rwnd*)
    - o zbývajícím volném místě
    - snižující *rwnd* indikuje nutnost snížení rychlosti vysílání
    - $rwnd = 0 \Rightarrow$ buffer přijímající stanice je plný, vysílač musí pozastavit vysílání
    - informace zasílána v rámci *ACK* packetů nebo s využitím *piggybackingu* (obousměrná komunikace)



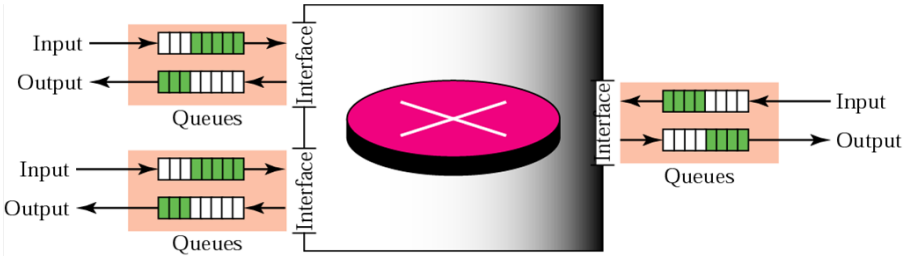
Řízení zahlcení (Congestion Control) I.

- snaha o přizpůsobení rychlosti vysílání dostupné kapacitě sítě
  - nejmenší volné kapacitě na trase
  - zahlcení (*congestion*) sítě $\Leftrightarrow$ počet paketů zaslaných do sítě $>$ kapacita sítě
- mechanismus závislý na dostupnosti informací ze sítě
  - explicitní zpětná vazba – síť dokáže informovat o (blížícím se) zahlcení
    - např. ATM sítě
  - bez zpětné vazby – nutnost **odhadovat** dostupnou kapacitu
    - běžné IP sítě
- dva možné přístupy k řešení:
  - *proaktivní přístup* – snaha zahlcení předcházet (tak, aby k němu nikdy nedošlo)
  - *reaktivní přístup* – jakmile dojde k zahlcení, je toto detekováno a řešeno (snížením rychlosti vysílání)

Řízení zahlcení (Congestion Control) II.

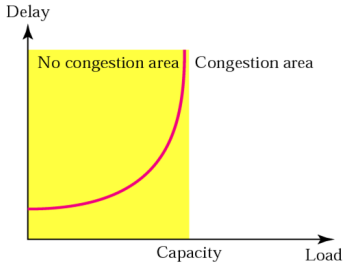
kde (a proč) může nastat zahlcení?

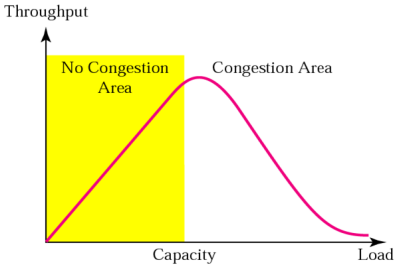
- směrovače/switche mají fronty (vstupní, výstupní)
- příchozí pakety je zapotřebí zpracovat (přeposlat blíže k cílové destinaci)
- zahlcení nastává $\Leftrightarrow$ když:
  - rychlost příchodu paketů je větší než rychlost jejich zpracování
    nebo
  - rychlost výstupu paketů je menší než rychlost jejich zpracování



Zahlcení sítě – důsledky

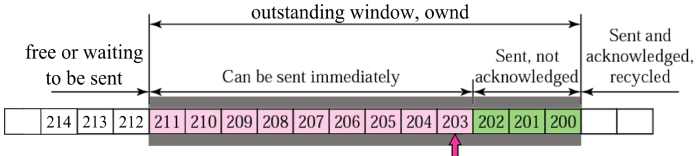
- zvýšení zpoždění přenosu a degradace rychlosti přenosu





Řízení zahlcení v TCP

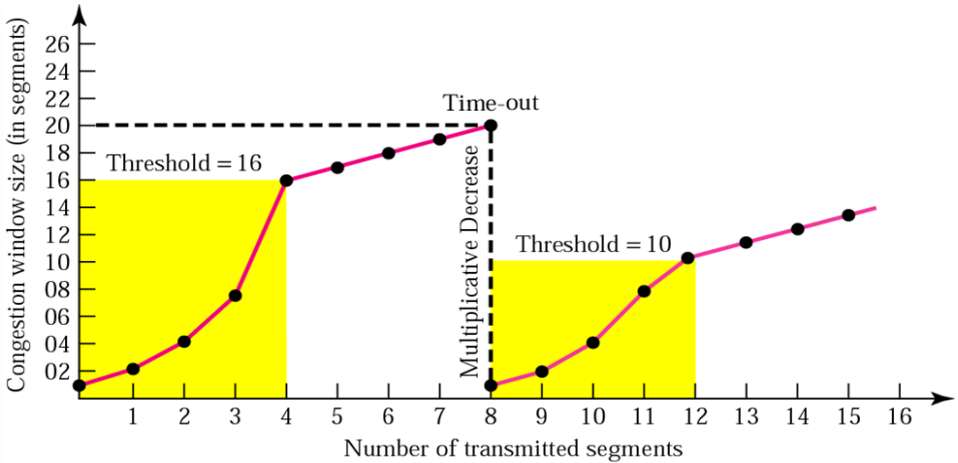
- zahlcení většinou detekováno na základě ztráty paketu
  - **předpoklad:** ztráta paketu způsobena přeplněním vstupní/výstupní fronty některého ze síťových zařízení po cestě
  - = reaktivní přístup
  - *většinou* = existují varianty TCP s proaktivním přístupem
- v průběhu přenosu odhadována velikost okna zahlcení (*congestion window, cwnd*)
  - *cwnd* určuje množství dat, které lze do sítě zaslat, aniž by došlo k zahlcení
- množství skutečně zasílaných dat definováno velikostí tzv. *outstanding window, ownd*
  - $ownd = min(rwnd, cwnd)$



Řízení zahlcení v TCP – odhad *cwnd* I.

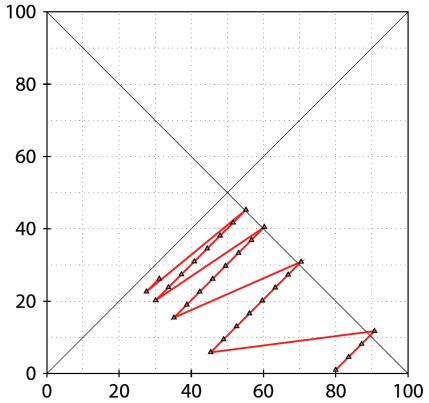
- běžná IP síť nepodává explicitní informace o dostupné přenosové kapacitě či blížícím se zahlcení
  - $\Rightarrow$ dostupnou přenosovou kapacitu (tj. velikost *cwnd*) musí TCP **odhadovat**
- využity tři základní algoritmy pro odhad *cwnd* (přístup *AIMD – Additive Increase, Multiplicative Decrease*):
  - fáze *Slow Start*
    - „pomalý“ start – snaha o rychlé navýšení rychlosti odesílání až do dosažení určité hranice
  - fáze *Additive Increase*
    - zpomalení rychlosti růstu
    - snaha o udržení vysoké rychlosti přenosu po co možná nejdelší dobu
  - fáze *Multiplicative Decrease*
    - zahlcení sítě (= ztráta paketu) – snížení rychlosti přenosu
    - mimo jiné umožňuje zajištění férovosti mezi TCP proudy

Řízení zahlcení v TCP – odhad *cwnd* II.



Obrázek: Ilustrace mechanismu AIMD.

TCP – zajištění férovosti



Obrázek: Zajištění férovosti dvou rovnocenných TCP proudů.