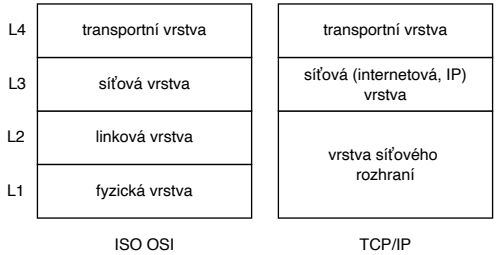
Počítačové sítě 1

transportní vrstva

Martin Trnečka

Katedra informatiky
Univerzita Palackého v Olomouci

TCP/IP architektura



Transportní vrstva

- zprostředkovává komunikaci mezi procesy/aplikacemi (pozor na rozdíl oproti předchozím vrstvám)
- adresace procesu $\rightarrow$ MAC adresa + IP adresa = „neúplná adresa“ $\rightarrow$ *port*
- balení dat do *segmentů*, (uživatelských) *datagramů*, transportních paketů
- řízení toku dat mezi koncovými uzly
- detekci chyb (integrita dat, chyby přenosu = ztracené data)

Port

- identifikátor (adresa) aplikace (aplikace může používat více portů)
- 2 B, 0–65535
  - porty 0–1023 jsou privilegované (bezpečné, well-known) → vyhrazené pro „standardní“ služby (určeno RFC)
  - např. port 80 pro službu WWW
  - porty 1024–65535 jsou klientské → určené pro klientské aplikace
- pokud je port používaný, nelze jej přidělit (záležitost OS)
- rozdělení je pouze doporučení → privilegované lze použít i pro jiné služby
- z pohledu aplikace je síťová architektura zpřístupněna skrze *socket* (socket API)
  - IP adresa + transportní protokol + port

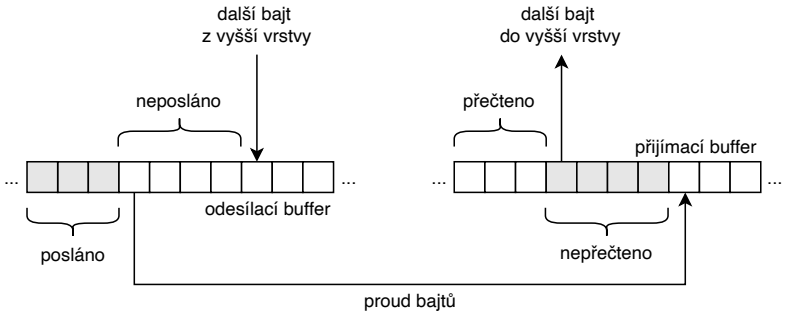
Transportní vrstva: Služby

- spojová spolehlivá služba
  - navazuje, udržuje a ukončuje (plně duplexní) spojení
  - spolehlivost = zajištění správného doručení dat
  - zpracovává souvislý proud dat z vyšší vrstvy a dělí jej na segmenty
  - protokol TCP
- nespojová nespolehlivá služba
  - nenavazuje spojení
  - nespolehlivost = není zajištěno správné doručení dat (data nemusí být vůbec doručena)
  - zpracovává nesouvislé části dat z vyšší vrstvy (datagramy) → data jsou dělena na aplikační vrstvě
  - protokol UDP
- nezávislý rozsah portů pro TCP a UDP, porty označeny `služba/číslo`

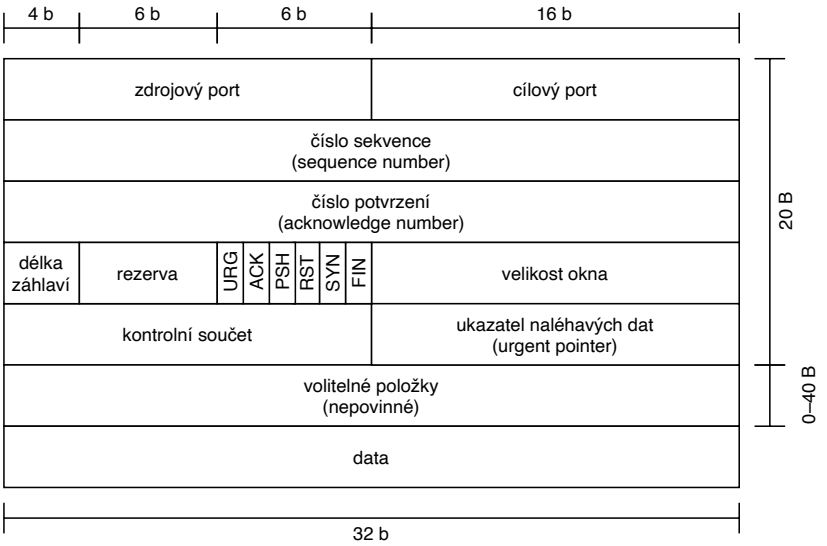
TCP

- protokol IP je nespolehlivý! $\rightarrow$ TCP
- zaručení správného pořadí dat
- potvrzování přijetí dat (pozitivní potvrzování)
- vyžádání opakování přenosu ztracených nebo poškozených dat
- ztráta dat
  - příjemce nestíhá zpracovávat $\rightarrow$ je třeba řídit tok dat mezi odesílatelem a příjemce
  - síť je zahlcena $\rightarrow$ je třeba řídit tok dat sítí
- řízení toku dat a předcházení zahlcení sítě pomocí:
  - bufferů (odesílací a přijímacích)
  - opakovaného odeslání a potvrzení přijetí dat
  - časových prodlev
  - posuvného okna a okna zahlcení
  - adaptivního přizpůsobení parametrů protokolu podle stavu spojení

Protokol TCP: Proud dat

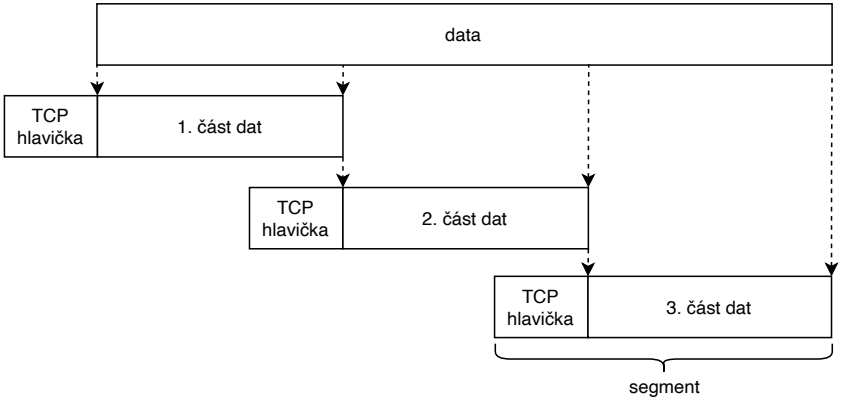


Struktura TCP segmentu

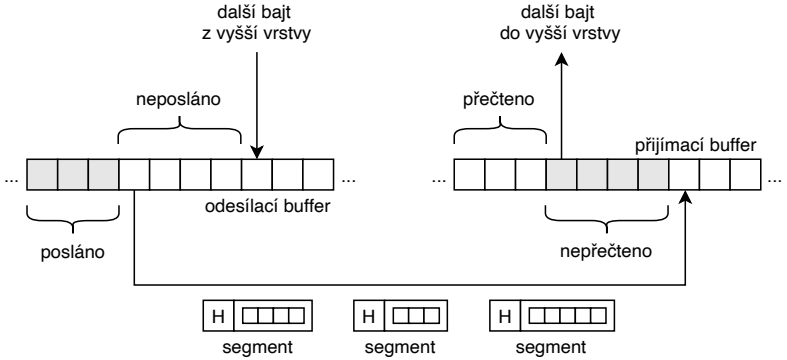


TCP: Segmentace dat

- velikost hlavičky + dat v TCP segmentu je omezena velikostí IP paketu
- segmentace = data z aplikační vrstvy jsou rozdělena na menší části (segmenty)



TCP: Proud dat – revize



TCP: Číslování

- číslo sekvence a číslo potvrzení = číslo bajtu
- pozor: nejedná se o číslo segmentu!
- bajty nemusí (a obvykle nejsou) číslování od nuly
  - odesílatel i příjemce si náhodně vybírají první číslo (tzv. Initial Sequence Number, ISN)
  - například: zvolíme číslo 42, posíláme 6000 B, bajty jsou číslovány 43–6043
- číslo sekvence = číslo prvního bajtu dat přenášeného v segmentu
- číslo potvrzení = číslo bajtu, který má být přijat (všechny předešlé bajty jsou přijaty)

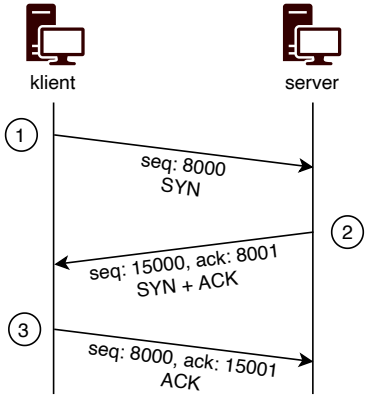
TCP: Příznakové bity

příznak	popis
URG	ukazatel naléhavých dat je nastaven
ACK	číslo potvrzení je nastaveno
PSH	požadavek na okamžité zpracování (push)
RST	reset spojení
SYN	synchronizace initial sequence number (ISN) – číslování bajtů
FIN	ukončení spojení

TCP: Navázání spojení

- klient-server architektura
- klient navazuje spojení, server jej přijímá nebo odmítá
- aplikace musí mít otevřený (přidělený) port (realizace pomocí soketu)
- TCP používá třífázový handshake pro navázání spojení
- navázání spojení = výměna TCP segmentů

TCP: Třífázový handshake

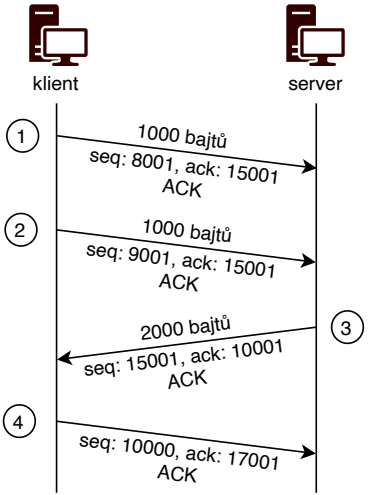


1. klient pošle segment (bez dat) s příznakem `SYN` a náhodně vygenerovaným číslem sekvence ($\text{ISN}_k = 8000$)
2. server pošle segment (bez dat) s příznaky `SYN` a `ACK`, náhodně vygenerovaným číslem sekvence ($\text{ISN}_s = 15000$) a číslem potvrzení $\text{ISN}_k + 1$
3. klient pošle segment (obvykle bez dat) s příznakem `ACK` a číslem potvrzení $\text{ISN}_s + 1$

TCP: Přenos dat

- po navázání spojení je možné posílat data oběma směry
- segmenty obsahují příznaky `ACK`, případně `ACK` a `PSH`
- data jsou odstraněna z bufferu až v okamžiku potvrzení jejich přijetí
- segmenty nenesoucí žádná data nezvyšující číslo sekvence
- výjimkou jsou segmenty s příznaky `SYN` a `FIN`

TCP: Přenos dat – příklad



Klient posílá 2000 B ve dvou segmentech, server 2000 B v jednom segmentu.

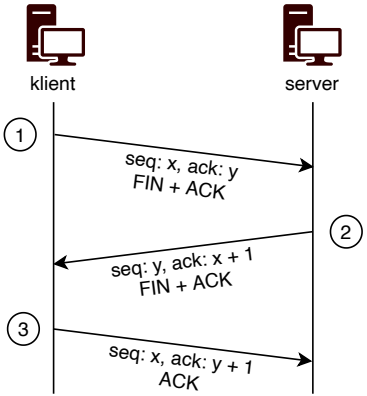
① klient pošle segment obsahující 1000 B dat s příznakem `ACK`, první bajt odeslaných dat má číslo 8001, `ack: 15001` indikuje očekávanou sekvenci

② klient pošle segment obsahující 1000 B dat s příznakem `ACK`, první bajt odeslaných dat má číslo 9001, `ack: 15001` se nemění (od serveru nepřišla žádná data)

③ server pošle segment obsahující 2000 B dat s příznakem `ACK`, `ack: 10001` potvrzuje přijetí segmentů z kroku ① a ②

④ klient pošle segment bez dat s příznakem `ACK`, `ack: 17001` potvrzuje přijetí segmentů z ③

TCP: Ukončení spojení (3 fázové)



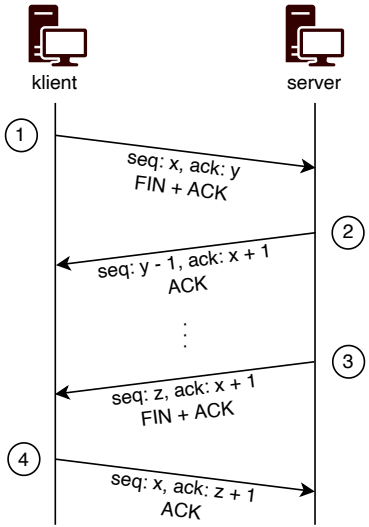
klient poslal bajty do č. `x-1`, server do č. `y-1`

① aktivní uzavření, klient pošle segment (může obsahovat data) s příznaky `FIN` a `ACK` a `seq: x` (`FIN` vždy zvyšuje `seq`), `ack: y`

② server pošle segment (může obsahovat data) s příznaky `FIN` a `ACK` a `seq: y`, pokud segment z kroku ① neobsahoval data, je `ack` nastaven na `x + 1` jinak na `x + n + 1`, kde `n` je počet bajtů dat v segmentu z kroku ①

③ klient pošle segment (bez dat) s příznakem `ACK`, pokud segment z kroku ② neobsahoval data je `ack` nastaven na `y + 1` jinak na `y + m + 1`, kde `m` je počet bajtů dat v segmentu z kroku ②

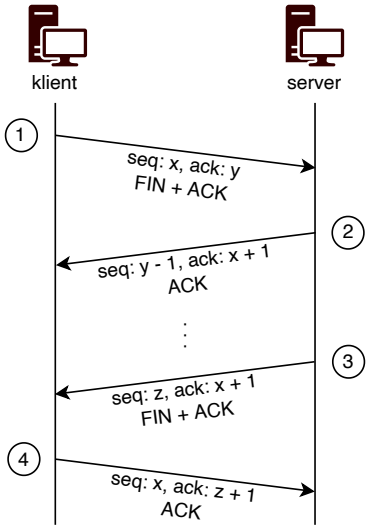
TCP: Polouzavřené spojení (4 fázové ukončení)



① klient pošle segment (může obsahovat data) s příznaky `FIN` a `ACK` a `seq: x`, `ack: y`, klient dále nemůže vysílat data, může ale přijímat a posílat segmenty s `ACK`

② server pošle segment (může obsahovat data) s příznakem `ACK` pokud segment neobsahuje data, je `seq` nastaven na `y - 1` jinak na `y`, pokud segment z kroku ① neobsahoval data, je `ack` nastaven na `x + 1` jinak na `x + n + 1`, kde `n` je počet bajtů dat v segmentu z ①, server může dále posílat data

TCP: Polouzavřené spojení (4 fázové ukončení)



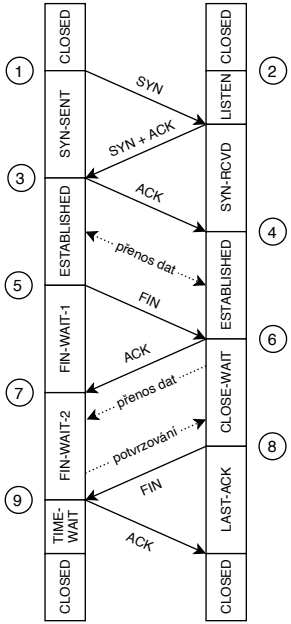
③ server pošle segment (může obsahovat data) s příznaky `ACK` a `FIN`, pokud segment z kroku ① neobsahoval data, je `ack` nastaven na `x + 1` jinak na `x + n + 1`, kde `n` je počet bajtů dat v segmentu z kroku ①

④ server pošle segment s příznakem `ACK`, pokud segment z kroku ③ neobsahoval data pak `ack: z + 1`, jinak `ack: z + m + 1`, kde `m` je počet bajtů dat v segmentu z kroku ③

TCP: Problém při uzavření spojení

- asymetrické uzavření → ztráta dat
  - pokud je známa struktura dat, je možné i asymetrické uzavření
- symetrické uzavření (strany se dohodnou na konci) → stále může dojít ke ztrátě dat
- problém dvou generálů → nemá řešení
- zjednodušení problému: každá strana si řeší ukončení samostatně → časovače

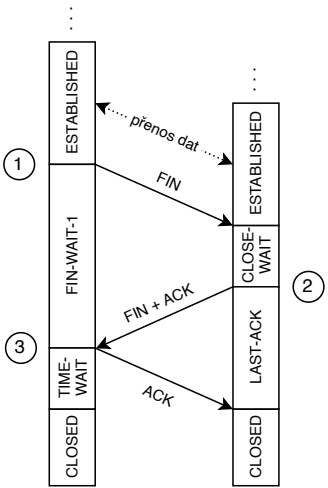
TCP: Stavy spojení (při polouzavřeném spojení)



1. aktivní otevření → zahájení spojení
2. pasivní otevření → čekání na segment s příznakem `SYN`
3. spojení je otevřeno
4. spojení je otevřeno
5. aktivní uzavření
6. spojení je polouzavřené
7. spojení je uzavřené
8. pasivní uzavření
9. čeká se 30 s – 2 m (dvojnásobek maximální životnosti segmentu, odvozeno z TTL IP paketu) → kvůli možné ztrátě posledního `ACK`

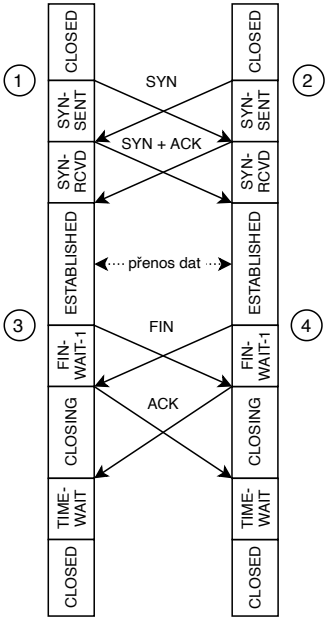
poznámka: stav `CLOSED` → spojení neexistuje

TCP: Stavy spojení (při 3 fázovém uzavření spojení)



1. aktivní uzavření
2. pasivní uzavření
3. spojení je uzavřeno

Odbočka: Simultánní navázání a ukončení spojení



- TCP podporuje současné otevření (není server, pouze dva klienti)
- netypický scénář

① aktivní otevření
② aktivní otevření
③ aktivní uzavření
④ aktivní uzavření

Protokol TCP: Zrušení a odmítnutí spojení

- příznak `RST`
- pokud klient adresuje port, který není ve stavu `LISTEN`, nebo pokud jsou segmenty zahazovány firewallem → klient opakuje (periodicky) zaslání segmentu s příznakem `SYN`
  - pauza mezi pokusy
  - opakuje se dokud nevyprší celkový čas
  - → časová prodleva
- pokud server nechce spojení přijmout může zaslat segment s příznakem `RST` (bez dat) → obě strany okamžitě ukončí spojení
- analogicky klient může zrušit spojení zasáláním segmentu s příznakem `RST` (v tomto případě je nastaven i příznak `ACK`, ale nemá význam)
  - používá se pro zrychlení ukončení spojení (obecně ale chápáno jako chybový stav)
  - použití např. při detekci dlouho neaktivního spojení nebo při neúspěšném vytvoření šifrovaného kanálů

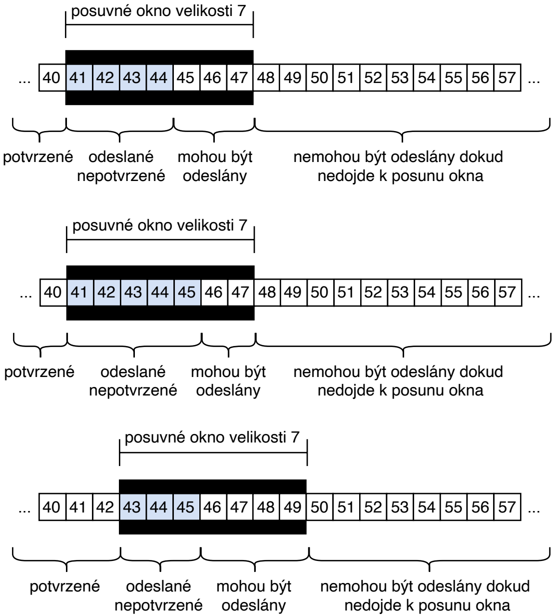
Odbočka: Urgentní data v TCP

- příznak `URG`
- indikace, že část dat (urgentní data) vyžaduje jiné zpracování
- urgentní data jsou vložena na začátek segmentu, zbytek dat je umístěn normálně
- ukazatel urgentních dat je nastaven na poslední bajt urgentních dat
- urgentní data $\neq$ prioritní data
- urgentní data jsou zpracovávána stejně jako normální data

TCP: Posuvné okno

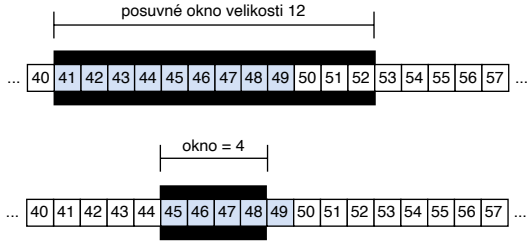
- omezení vysílacího a přijímacího bufferu $\rightarrow$ řízení toku
  - při navazování spojení si strany vymění velikosti okna (položka v hlavičce TCP)
  - příjemce informuje odesílatele o počtu bajtů, které dokáže přijmout (kolik dat lze poslat aniž by došlo k zahlcení příjemce)
  - každá strana je zároveň odesílatelem i příjemcem $\rightarrow$ dvě okna
- omezení počtu potvrzení $\rightarrow$ snížení provozu
  - stop and wait $\rightarrow$ neefektivní
  - data v okně lze odeslat bez nutnosti čekání na potvrzení jejich přijetí
  - jakmile je část dat potvrzena okno se posouvá

TCP: Posuvné okno – ilustrace



TCP: Posuvné okno

- velikost okna se může v průběhu spojení měnit → řízení toku
- pokud příjemce nestíhá zpracovávat požádá o zmenšení okna
- při zmenšení okna na 0 dochází k jeho úplnému uzavření → odesílatel přestává posílat data
- změnit lze pouze odesílací okno nikoliv přijímací
- při zmenšení okna nesmí dojít k situaci, kdy odeslané bajty jsou mimo okno



- nové `ack` + nová velikost okna $\leq$ poslední `ack` + poslední velikost okna

TCP: Zpoždění odpovědi

- velká režie při odesílání menšího množství dat
- typicky konzolové aplikace (např. napsání znaku)
- zpoždění potvrzení
  - potvrzování (`ACK`) ne ihned při příjmu, ale se zpožděním, během kterého se mohou nahromadit data k odeslání
  - zpoždění $\leq$ 500 ms $\rightarrow$ zabránění opětovného odeslání
  - v době čekání se mohou objevit data k odeslání $\rightarrow$ snížení provozu
- Nagleův algoritmus
  1. první část dat je odeslána (i malá data)
  2. poté se data kumulují až do potvrzení odeslaných dat nebo až velikost kumulovaných dat odpovídá maximální velikosti segmentu
  3. krok 2. se opakuje
  - vyvažuje rychlost generování dat a rychlost sítě (generování rychlejší než síť $\rightarrow$ větší segmenty, generování pomalejší než síť $\rightarrow$ menší segmenty)
- kombinace vede na konstantní zpoždění (používá se vždy jedna varianta)

TCP: Pozitivní potvrzování

- klíčová součást TCP
- pravidla pro generování `ACK`
  - P1 při poslání dat je nastaven `ACK` a `ack` na číslo následující očekávané sekvence (snížení počet segmentů)
  - P2 pokud příjemce nemá data k odeslání a obdrží očekávaný segment (očekávané číslo sekvence) a předchozí segment byl potvrzen, pozdrží odeslání potvrzení dokud nepřijde další segment, maximálně ale 500 ms (snížení počtu potvrzení)
  - P3 pokud příjemce obdrží očekávaný segment (očekávané číslo sekvence) a předešlý segment nebyl potvrzen, je okamžitě odesláno potvrzení (zabránění zbytečnému opakovanému poslání)
  - P4 pokud příjemce obdrží neočekávaný segment (větší číslo sekvence) je okamžitě posláno potvrzení očekávaného segmentu
  - P5 pokud příjemce obdrží chybějící segment, je posláno potvrzení očekávaného segmentu (informování protistrany, že chybějící segment přišel)
  - P6 pokud příjemce obdrží duplicitní segment zahodí jej a je posláno potvrzení očekávaného segmentu (kompenzace ztracených potvrzení)

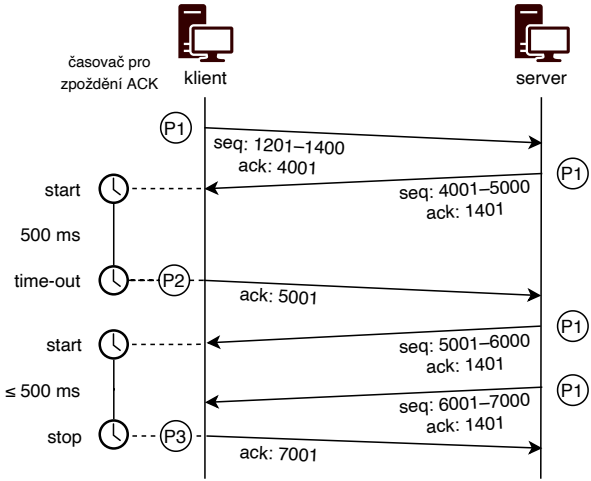
TCP: Opětovné poslání

- odeslaný segment je uložen do fronty
- je spuštěn časovač (retransmission time-out, RTO) $\rightarrow$ čas se dynamicky mění v závislosti na stavu sítě (počítán z round-trip time, RTT)
- odbočka: RTT = čas potřebný pro potvrzení segmentu $\rightarrow$ fluktace $\rightarrow$ vážený průměr
- pokud je segment potvrzen je odstraněn z fronty
- pokud vyprší časovač je poslán segment na začátku fronty a je opět restartován časovač
- pokud strana obdrží 3 duplicitní potvrzení (originál + 3 kopie) je okamžitě poslán segment na začátku fronty $\rightarrow$ rychlé opětovné poslání (fast retransmission)

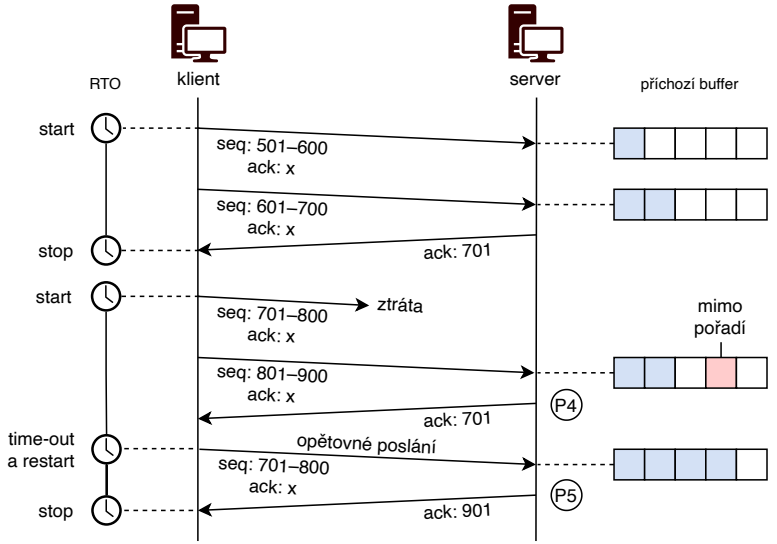
TCP: Segmenty mimo pořadí

- neduplicitní segment, který není očekáván $\rightarrow$ uložen do bufferu
- pokud přijde chybějící segment(y) $\rightarrow$ je možné potvrdit souvislou sekvenci segmentů
- segmenty mimo pořadí nejsou předány do vyšší vrstvy

TCP: Příklad – normální provoz

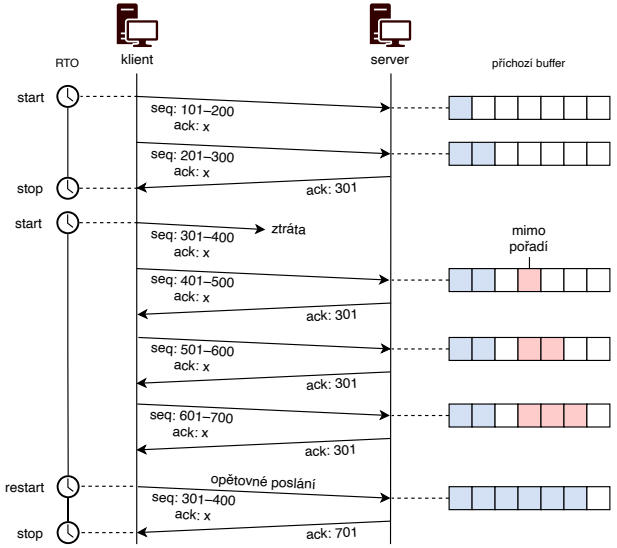


TCP: Příklad – ztracený segment



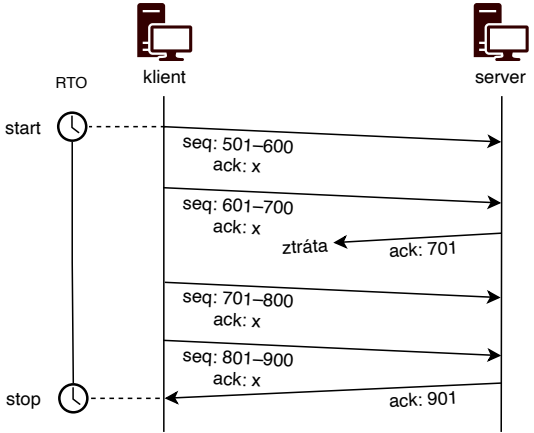
- pro jednoduchost P1, P2 a P3 není značeno

TCP: Příklad – rychlé opětovné poslání



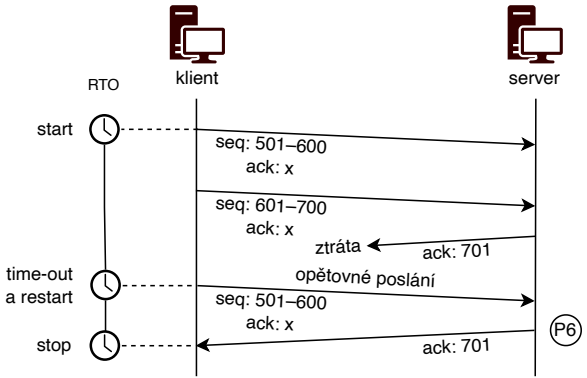
- P1–P5 není značeno, k opětovnému poslání dochází před vypršením časovače

TCP: Příklad – automatická korekce ztraceného potvrzení



- P1–P5 není značeno

TCP: Příklad – korekce ztraceného potvrzení opětovným posláním



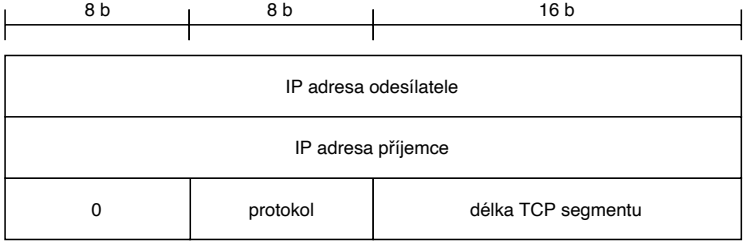
- P1–P5 není značeno

Odbočka: Ztráta potvrzení

- při ztrátě potvrzení může dojít k deadlocku
  - server pošle požadavek na uzavření okna
  - klient přestane posílat data
  - server chce otevřít okno
  - nemá data $\rightarrow$ pošle `ACK` segment bez dat
  - pokud se ztratí nastane deadlock
- persistence timer $\rightarrow$ spuštěn po uzavření okna
- po jeho vypršení je poslán speciální segment (probe) pomocí kterého je vynucen `ACK` od serveru
- pokud nepřijde odpověď na probe je znovu opakovaně probe vysílán (časovač je přenastaven na dvojnásobek při každém pokusu, dokud není dosaženo maxima – 60 s)
- pokračuje se dokud není okno znovu otevřeno nebo dokud není ukončeno spojení keepalive časovačem (cca 2 hodiny + žádná reakce na 10 probe)

TCP: Kontrolní součet

- počítán z pseudohlavičky (data z IP), hlavičky TCP a dat z aplikační vrstvy (přenášená data)
- data a volitelné položky se zarovnávají na násobek 16 bitů
- pseudohlavicka



- výpočet a verifikace jako u IP (součet 16 bitových částí)

TCP: Příznak PSH

- požadavek na okamžité zpracování
- data jsou odesláno okamžitě
  - nečeká se na naplnění okna
  - je vytvořen segment a aktuální data jsou odeslána
- pouze požadavek $\rightarrow$ může být ignorován

TCP: Volitelné položky

- až 40 B
- jedno-bajtové (pro zarovnání)
  - no-operation
  - end-of-option – konec položek (všech), lze použít pouze $1\times$
- více-bajtové
  - maximum-segment-size (MSS) – maximální velikost dat přenášených v segmentu (nikoliv velikost segmentu), 0–65535 B, MSS je určeno během navazování spojení, poté se již nemění, výchozí hodnota 536 B
  - windows-scale-factor – velikost okna je 0–65535 B $\rightarrow$ nemusí být dostateční pro rychlý přenos, lze navýšit

$$velikost\_okna = velikost\_okna\_v\_hlavice \cdot 2^{window\ \ scalefactor}$$

    pozn. velikost okna nemůže být větší než max. sequence number $\rightarrow 2^{32} \approx$ 1GB
  - timestamp – časové razítko pro segment, používá se pro výpočet RTT a pro zabránění duplicit v sequence number (segment je identifikován: `time:seq`)

TCP: Volitelné položky – selektivní potvrzování

- běžně známé jako SACK (selective ACK)
- možnost selektivního potvrzení na místo kumulativního → příjemce se dozví, které segmenty jsou mimo pořadí
- novější funkcionalita
- obě strany se musí dohodnou na začátku spojení (SACK-permited volitelná položka)
- lze zaslat informace o ztracených 4 blocích dat (začátek a konec)

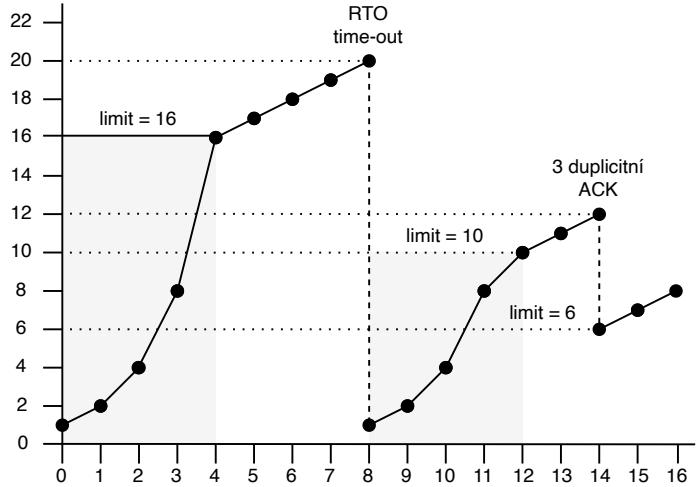
Odbočka: Zahlcení sítě

- problematika nad rámec kurzu → KMI/POS2
- manipulace s oknem nezohledňuje požadavky sítě (pouze požadavky příjemce)
- může dojít k zahlcení sítě → síť může určit velikost okna
- okno zahlcení (congestion window) → kolik dat je možné poslat, aby nedošlo k zahlcení sítě
- je volena menší hodnota z okna zahlcení a posuvného okna
- postup pro zabránění zahlcení
  - pomalý start – postupně $1\times$, $2\times$, $4\times$, $8\times$, ..., limit$\times$ MSS, nárůst vždy po přijetí ACK
  - vyhýbání se zahlcení – po skončení předchozí fáze, pokud je potvrzeno celé okno, je zvětšeno o 1
  - detekce zahlcení – detekována potřeba opětovného zaslání segmentu (RTO vypršel, 3 duplicitní ACK)

RTO limit $= \frac{1}{2}$ okna, okno zahlcení $= 1$, zahájí pomalý start

3 ACK limit $= \frac{1}{2}$ okna, okno zahlcení $=$ limit, zahájí vyhýbání se zahlcení

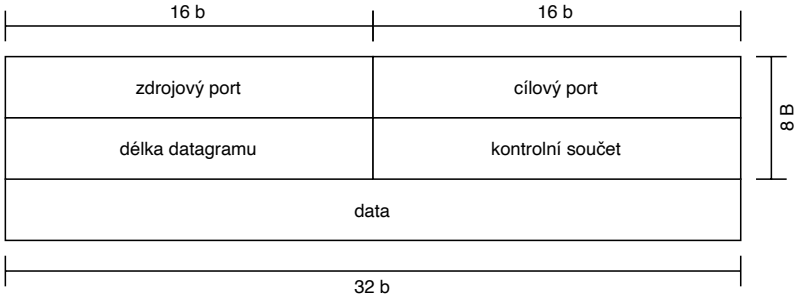
Odbočka: Zahlcení sítě



UDP

- nenavazuje spojení
- nezajišťuje spolehlivost
- nedisponuje prostředky pro řízení toku
- dělení dat do datagramu ponecháno na aplikační vrstvě
- vyšší výkon
  - absence potvrzování
  - menší hlavička
- použití
  - jednoduchá dotaz-odpověď komunikace
  - streamování mutlimedií
  - DNS
  - SNMP (Simple Network Management Protocol)
- oproti TCP může být příjemcem skupina uzlů (multicast i broadcast)

UDP: Struktura datagramu



- kontrolní součet jako u TCP
- na rozdíl od TCP je u UDP kontrolní součet volitelný (může tedy dojít k chybě, že data jsou doručena nesprávnému příjemci

SCTP a QUIC

- SCTP
  - transportní vrstva TCP/IP nabízí SCTP (Stream Control Transmition Protokol)
  - novější protokol
  - kombinace TCP a UDP (posílání datagramů + detekce ztracených, duplicitních a mimo pořadí)
  - poskytuje i možnosti nad rámec TCP a UDP (multiple-stream, multihoming – více odesílatelů a příjemců)
  - nad rámec kurzu → KMI/POS2
- QUIC
  - vyvíjený Googlem
  - vychází z UDP + šifrování
  - typicky integrace do L4 vrstvy (např. HTTP/3)

Odbočka: Užitečné nástroje

- `netstat`
  - výpis spojení a jejich stavů
  - pouze na lokálním počítači
- `nmap`
  - skenování portů na vzdáleném uzlu
  - zjištění portů ve stavu `LISTEN`
  - skenování TCP = obvykle zaslání `SYN` a čekání na `SYN` + `ACK`
  - při skenování nedostupného UDP portu je generována ICMP zpráva → složitější

Ukázka výpisu z nmap

```
nmap -A 192.168.1.11

Starting Nmap 7.80 ( https://nmap.org ) at 2020-01-29 06:57 CET
Nmap scan report for 192.168.1.11
Host is up (0.0015s latency).
Not shown: 977 closed ports

PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
53/tcp   open  domain      ISC BIND 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp  open  rpcbind     2 (RPC #100000)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp  open  exec        netkit-rsh rexecd
513/tcp  open  login       OpenBSD or Solaris rlogind
514/tcp  open  tcpwrapped
1099/tcp open  java-rmi    GNU Classpath grmiregistry
...
```

Bezpečnost na transportní vrstvě

- filtrace
  - na základě portu
  - na základě protokolu
  - na základě stavu spojení (např. zahazování segmentů, které nejsou součástí žádného existujícího spojení)
- na směrovačích bývají povoleny porty pro DNS (`53/udp`, `53/tcp`)
- plnohodnotný NAT $\rightarrow$ adresace na základ portu (PAT)
- strojové skenování (více) portů $\rightarrow$ detekováno (obvykle předchází útoku)
- dva protokoly: Secure Socket Layer (SSL) a Transport Layer Security (TLS, nahrazuje SSL) $\rightarrow$ šifrování dat z aplikační vrstvy

Bezpečnost na transportní vrstvě: Příklad DoS útok

- navázání TCP spojení → bezpečnostní problém
- `SYN` flooding útok (jeden z typů DoS)
- poslání mnoha segmentů s příznakem `SYN` z falešných IP adres (prosté poslání IP paketu s pozměněnou IP adresou odesílatele)
- příjemce odpoví `SYN + ACK` (navíc alokuje zdroje)
- při čekání na další krok navazování spojení (které nenastane) jsou zdroje blokovány
- příliš mnoho spojení → „příjemce se zhroutí“ (denial of services)
- zneužít (jiným způsobem) lze i příznaky `FIN`, `RST` nebo protokol UDP