



VYSOKÉ UČENÍ FAKULTA
TECHNICKÉ INFORMAČNÍCH
V BRNĚ TECHNOLOGIÍ



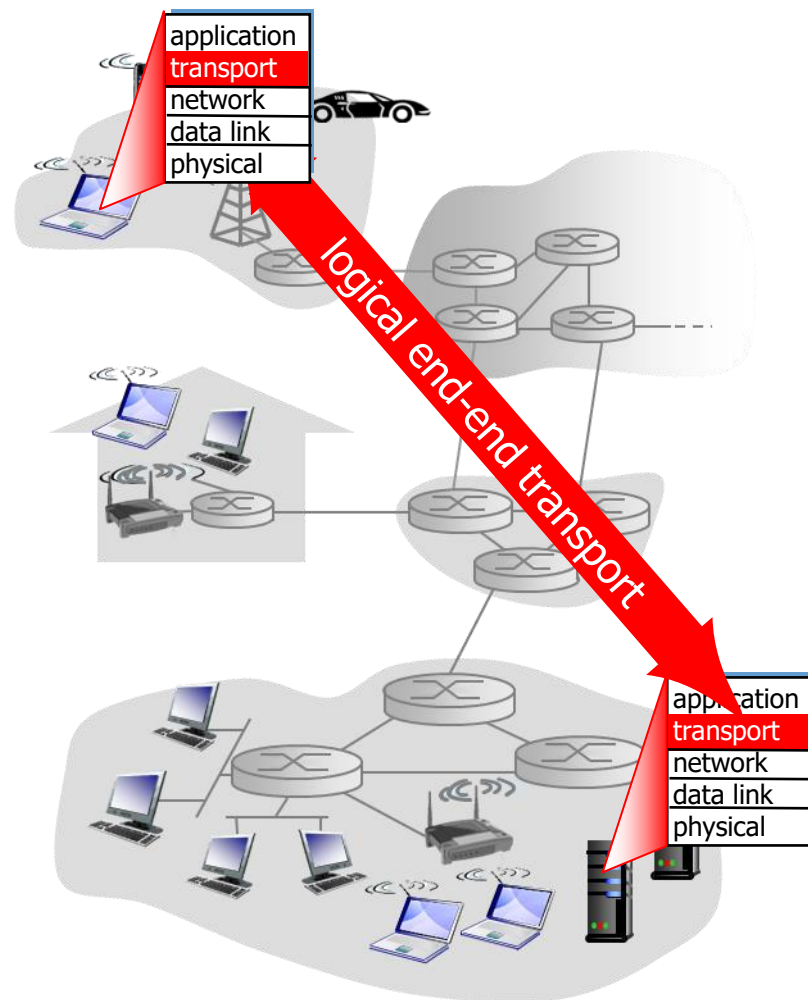
3

Transportní vrstva

IPK 2020/2021 L

Služby transportní vrstvy

- Logická komunikaci mezi aplikačními procesy
- L4 implementována především koncovými stanicemi
 - odesílatel „krájí“ (**segmentation**) aplikační data
 - příjemce „slepuje“ (**reassembling**) data dohromady
- Majoritně se využívá TCP a UDP
 - Dnes se rozmáhají i další protokoly
 - SCTP, MP-TCP, QUIC, RSVP



L4 vs. L3

- Síťová vrstva poskytuje
 - logickou komunikaci mezi zařízeními (hosts, nodes)
- Transportní vrstva poskytuje
 - logickou komunikaci mezi aplikacemi (application processes)

Analogie:

12 dětí z Kunhutina domu posílá dopisy 12 dětem z Božidařina domu:

- hosté = domy
- procesy = děti
- aplikační zprávy = dopisy v obálkách
- **transportní protokol** = Kunhuta a Božidara provedou multiplexing dopisu potomkům
- **síťový protokol** = pošta

L4 vrstva

- **Spolehlivé doručování v pořadí**
 - TCP
 - řízení toku (flow control)
 - = odesílatel/příjemce detekují ztráty a reagují na ně
 - řízení zahlcení (congestion control)
 - = odesílatel nezahltí příjemce či síť daty
 - ustavení spojení (connection-oriented)
- **Nespolehlivé doručování bez garance pořadí**
 - UDP
 - pokračování konceptu „best-effort“ od IP
 - bez ustavení spojení (connection-less)



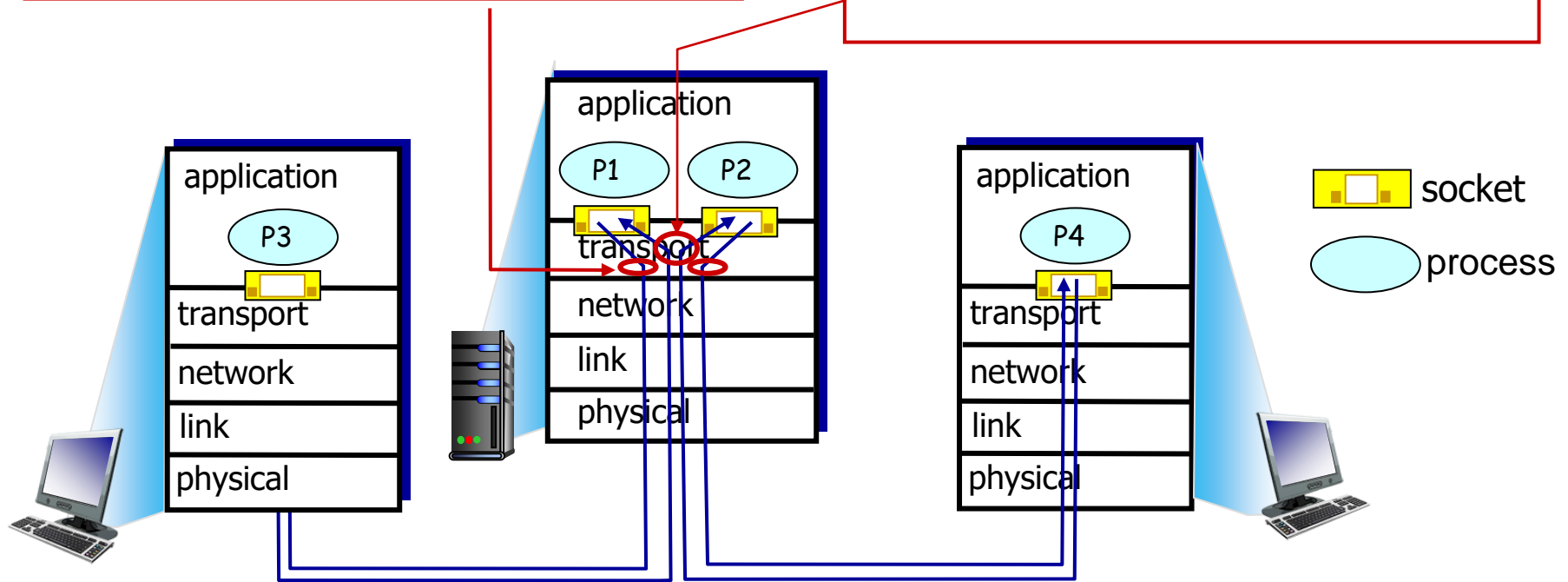
Multiplexing

multiplexing odesílatel:

Odesílá různá data z různých socketů, **přidá transportní hlavičku** (později demultiplexing)

demultiplexing příjemce:

použije transportní hlavičku **k doručení dat správnému socketu**



Obsah

1) PROTOKOL UDP

- Funkce, formát rámce
- Porty

2) SPOLEHLIVÝ PŘENOS

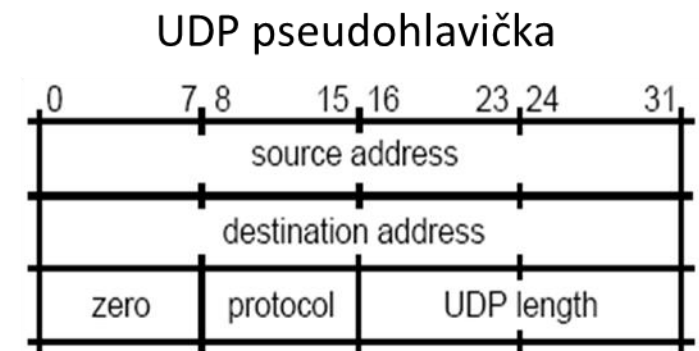
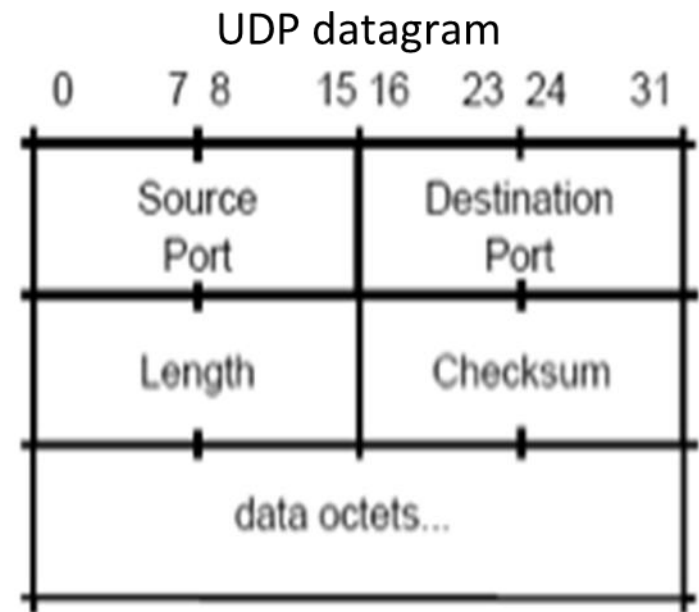
3) PROTOKOL TCP

Protokol UDP

- User Datagram Protocol [[RFC 768](#)]
- Jednoduchý protokol transportní vrstvy
- Datagramová služba bez záruky doručení, pořadí paketů
 - Řadě aplikací vyhovuje (DNS, multimedia, ...)
 - Pokud je vyžadována spolehlivost, řeší se na úrovni aplikace
- Nespojovaná služba ([connection-less](#))
 - Nevytváří spojení
 - = Žádná domluva na parametrech spojení ([no handshaking](#))

Datagram

- **Port** odesílatele (**source port**) a příjemce (**destination port**): identifikace odesílající a přijímací aplikace
- **Length** = délka hlavičky + data (v B)
- **Kontrolní součet** (checksum): pokrývá pseudohlavičku síťového protokolu (IP, pouze některé položky [protocol=17]) + UDP hlavičku + data (volitelně doplněno nulovým B na sudý počet)



Porty

- K odeslání dat je třeba znát číslo portu dané aplikace
- Servery používají standardní (well-known) porty
 - <http://www.iana.org/assignments/port-numbers>
 - UNIX /etc/services
 - Nestandardní port serveru musí klientovi (aplikaci) specifikovat uživatel
- Klienti používají náhodná čísla portů
 - Závisí na implementaci
 - Hodnoty > 1023
- Rozdělení čísel portů
 - 0 – 1023: Well-Known Ports (registruje IANA)
 - 1024 – 49151: Registered Ports (registruje IANA)
 - 49152 – 65535: Dynamic and/or Private Ports

Protokoly a Porty

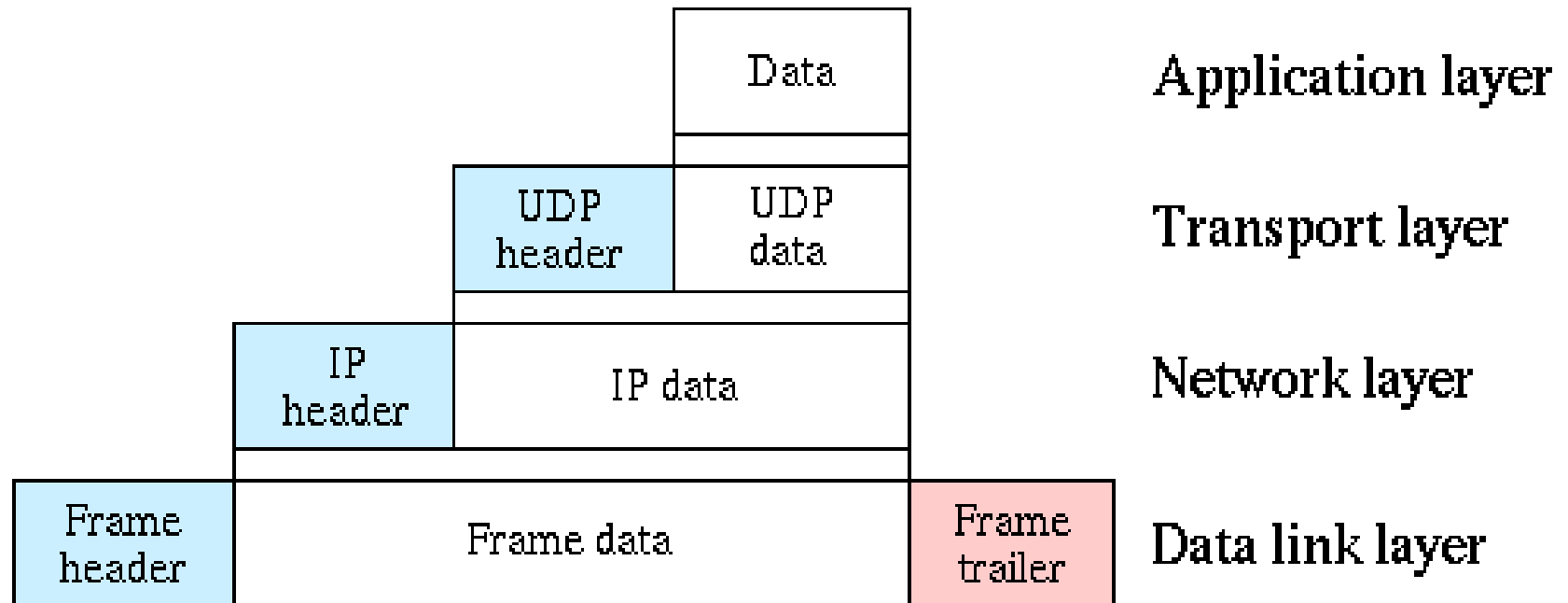
- Mapování vestavěné do funkce operačního systému
- Např.
Windows\System32\drivers\etc\services



```
Listner - [c:\WINDOWS\System32\drivers\etc\services]
Soubor Upravit Možnosti Kódování Nápoředa 13 %
# Copyright (c) 1993-2004 Microsoft Corp.
#
# This file contains port numbers for well-known services defined by IANA
#
# Format:
#
# <service name> <port number>/<protocol> [aliases...] [#<comment>]
#
echo 7/tcp
echo 7/udp
discard 9/tcp sink null
discard 9/udp sink null
systat 11/tcp users #Active users
systat 11/udp users #Active users
daytime 13/tcp
daytime 13/udp
qotd 17/tcp quote #Quote of the day
qotd 17/udp quote #Quote of the day
chargen 19/tcp ttytst source #Character generator
chargen 19/udp ttytst source #Character generator
ftp-data 20/tcp
ftp 21/tcp #FTP, data
ssh 22/tcp #SSH Remote Login Protocol
telnet 23/tcp
smtp 25/tcp mail #Simple Mail Transfer
Protocol
time 37/tcp timserver
time 37/udp timserver
rtp 39/udp resource #Resource Location Protocol
nameserver 42/tcp name #Host Name Server
nameserver 42/udp name #Host Name Server
nicname 43/tcp whois
domain 53/tcp #Domain Name Server
domain 53/udp #Domain Name Server
bootps 67/udp dhcps #Bootstrap Protocol Server
bootpc 68/udp dhcps #Bootstrap Protocol Client
tftp 69/udp #Trivial File Transfer
gopher 70/tcp
finger 79/tcp
http 80/tcp www www-http #World Wide Web
hosts2-ns 81/tcp #HOSTS2 Name Server
hosts2-ns 81/udp #HOSTS2 Name Server
kerberos 88/tcp krb5 kerberos-sec #Kerberos
kerberos 88/udp krb5 kerberos-sec #Kerberos
hostname 101/tcp hostnames #NIC Host Name Server
iso-tsap 102/tcp #ISO-TSAP Class 0
rtelnet 107/tcp #Remote Telnet Service
pop2 109/tcp postoffice #Post Office Protocol -
Version 2
pop3 110/tcp #Post Office Protocol -
Version 3
```

Zapouzdření UDP

- https://upload.wikimedia.org/wikipedia/en/d/d8/UDP_encapsulation.png



UDP ve Wiresharku

The image shows a Wireshark packet capture of a DNS query and response over UDP. The packet list at the top shows several packets, with packet 719 selected. The packet details pane shows the structure of the packet, including Ethernet II, Internet Protocol Version 6, and User Datagram Protocol (UDP). The packet bytes pane shows the raw data of the packet, with the DNS query data highlighted.

No.	Time	Source	Destination	Protocol	Length	Info
718	9.812536000	2001:67c:1220:8b5:7497:89fc:6b44:6619	2001:4860:4860::8888	DNS	89	Standard query 0xcf47 A seznam.cz
719	9.812901000	2001:67c:1220:8b5:7497:89fc:6b44:6619	2001:4860:4860::8888	DNS	89	Standard query 0x4d49 AAAA seznam.cz
723	9.833512000	2001:4860:4860::2001:67c:1220:8b5:7497	2001:67c:1220:8b5:7497:89fc:6b44:6619	DNS	105	Standard query response 0xcf47 A 77.7
724	9.834195000	2001:4860:4860::2001:67c:1220:8b5:7497	2001:67c:1220:8b5:7497:89fc:6b44:6619	DNS	117	Standard query response 0x4d49 AAAA 2
5371	47.118646000	fe80::2a37:37ff:ff02::fb		MDNS	201	Standard query response 0x0000 PTR _s
5372	47.120020000	147.229.181.21	224.0.0.251	MDNS	88	Standard query 0x0000 PTR _services._
5373	47.120021000	147.229.181.65	224.0.0.251	MDNS	113	Standard query response 0x0000 PTR _t

Frame 719: 89 bytes on wire (712 bits), 89 bytes captured (712 bits) on interface 0

Ethernet II, Src: AsustekC_8a:95:6e (78:24:af:8a:95:6e), Dst: ExtremeN_1d:4e:30 (00:04:96:1d:4e:30)

Internet Protocol Version 6, Src: 2001:67c:1220:8b5:7497:89fc:6b44:6619 (2001:67c:1220:8b5:7497:89fc:6b44:6619)

0110 = Version: 6

.... 0000 0000 = Traffic class: 0x00000000

.... 0000 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000

Payload length: 35

Next header: UDP (17)

Hop limit: 64

Source: 2001:67c:1220:8b5:7497:89fc:6b44:6619 (2001:67c:1220:8b5:7497:89fc:6b44:6619)

Destination: 2001:4860:4860::8888 (2001:4860:4860::8888)

[Source GeoIP: Unknown]

[Destination GeoIP: Unknown]

User Datagram Protocol, Src Port: 52746 (52746), Dst Port: 53 (53)

Source Port: 52746 (52746)

Destination Port: 53 (53)

Length: 35

Checksum: 0x4ac2 [validation disabled]

[Stream index: 7]

Domain Name System (query)

0000 00 04 96 1d 4e 30 78 24 af 8a 95 6e 86 dd 60 00N0x\$...n..`

0010 00 00 00 23 11 40 20 01 06 7c 12 20 08 b5 74 97 ...#.@ .|.t.

0020 89 fc 6b 44 66 19 20 01 48 60 48 60 00 00 00 00 ..kdf. .H`H`....

0030 00 00 00 00 88 88 ce 0a 00 35 00 23 4a c2 4d 495.#J.MI

0040 01 00 00 01 00 00 00 00 00 00 06 73 65 7a 6e 61sezna

0050 6d 02 63 7a 00 00 1c 00 01 m.cz....

User Datagram Protocol (udp), 8 bytes

Packets: 5816 · Displayed: 13 (0,2%) · Dropped: 0 (0,0%)

Profile: Default

Obsah

1) PROTOKOL UDP

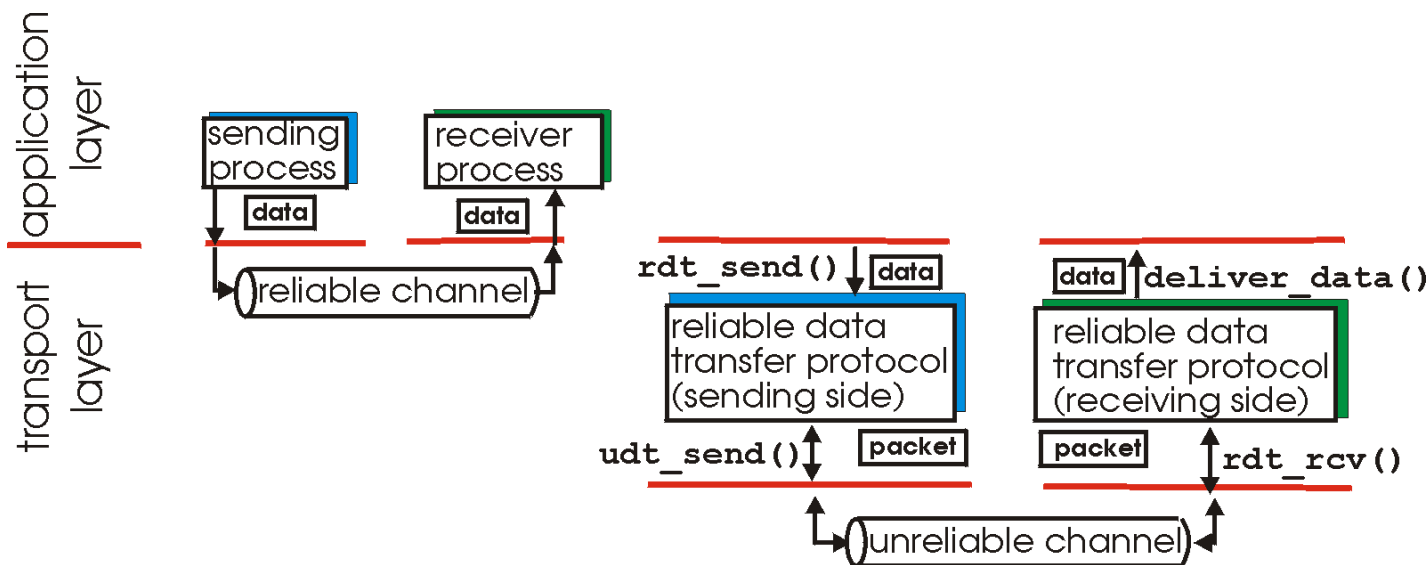
2) SPOLEHLIVÝ PŘENOS

- Principy spolehlivého přenosu
- Stop-and-wait
- Pipelined protokoly
- Go-Back-N
- Selective Repeat

3) PROTOKOL TCP

Úvod

- zajištění spolehlivého a zároveň efektivního přenosu je jedno z nejdůležitějších témat v oblasti sítí
- velká většina služeb závisí na spolehlivé komunikaci
- charakteristika nespolehlivého přenosového kanálu ovlivňuje složitost mechanismu spolehlivého přenosu



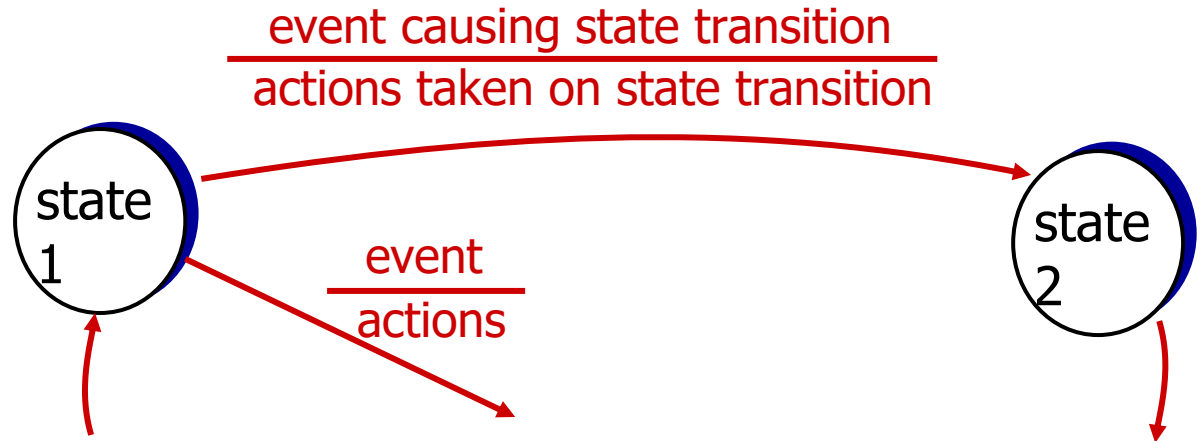
(a) provided service

(b) service implementation

Plán

- Postupně budeme definovat protokoly spolehlivého přenosu, které:
 - kompenzují různé vlastnosti nespolehlivého kanálu
 - injekce chyb, ztráta paketů, přeházení paketů
 - mají lepší výkonnost
- Předpoklady
 - uvažujeme pouze jednosměrný přenos dat
 - zpětná vazba může putovat ale oběma směry
 - chování odesílatele a příjemce bude popsáno konečným stavovým automatem (FSM)

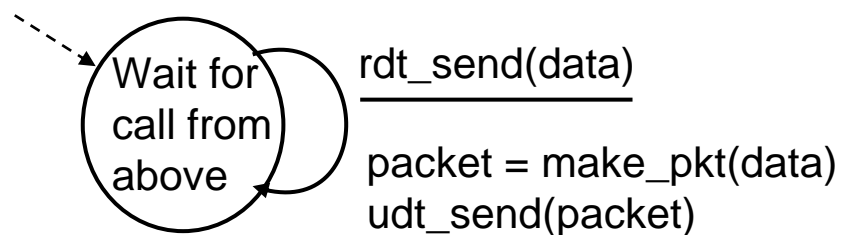
state: when in this “state” next state uniquely determined by next event



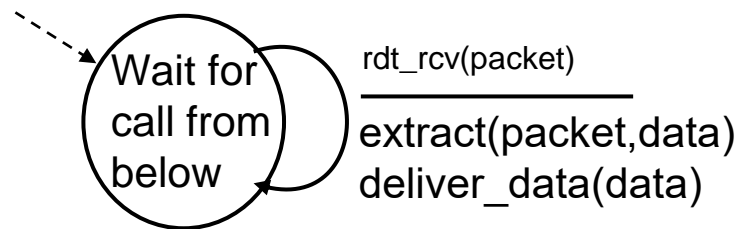
RDT 1.0

(spolehlivý kanál)

- Kanál je dokonale spolehlivý
 - nedochází k poškozování paketů
 - nedochází ke ztrátám paketů
 - nedochází k zpřeházení paketů



odesilatel



příjemce

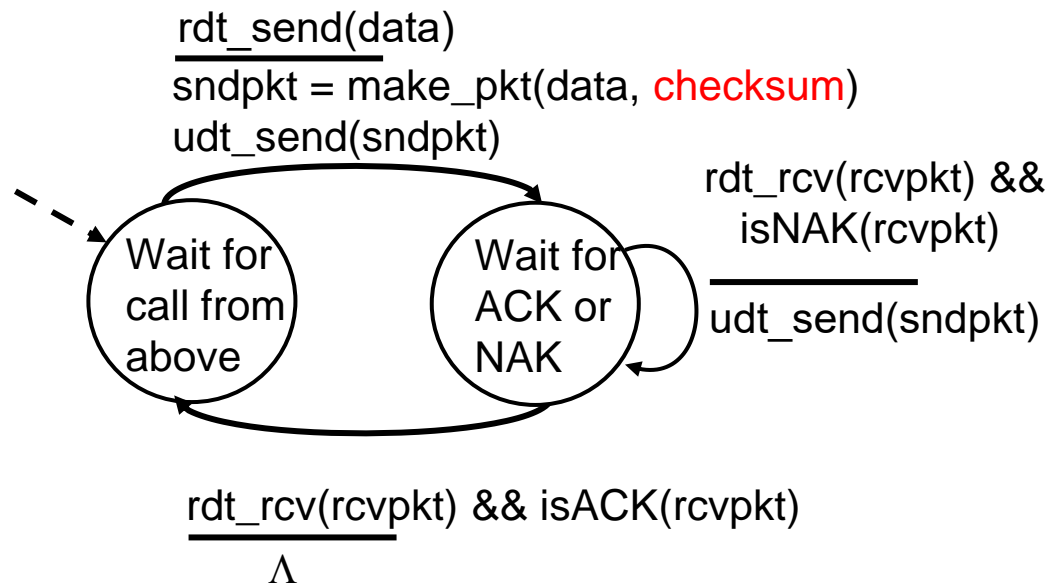
RDT 2.0

(kanál s bitovými chybami)

- *Předpoklad: na kanálu může dojít k chybě*
 - typicky bitová inverze 1 či více bitů
 - důvodem je např. rušení média, přeslechy, ad.
 - kontrolní součet pro detekci výskytu chyb
- **pozitivní potvrzování (ACK)**
 - příjemce potvrzuje přijatá korektní data
- **negativní potvrzování (NACK)**
 - příjemce potvrzuje přijatá NEkorektní data
- RDT2.0
 - je schopen kontroly chyb (**error detection**)
 - posílá potvrzení od příjemce odesilateli

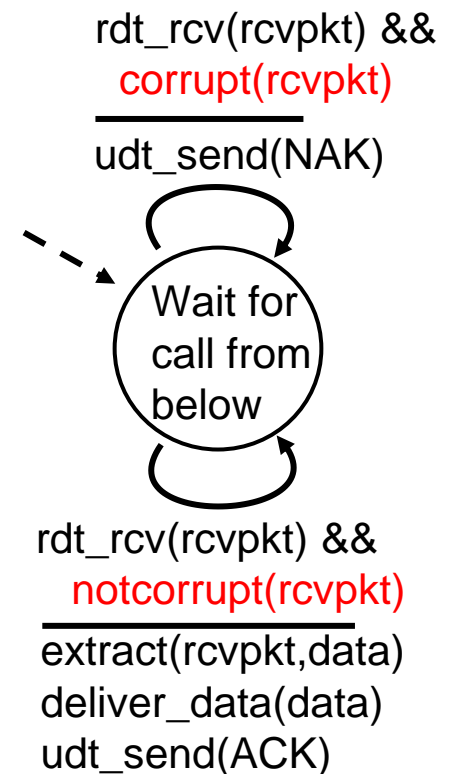
RDT 2.0

FSM



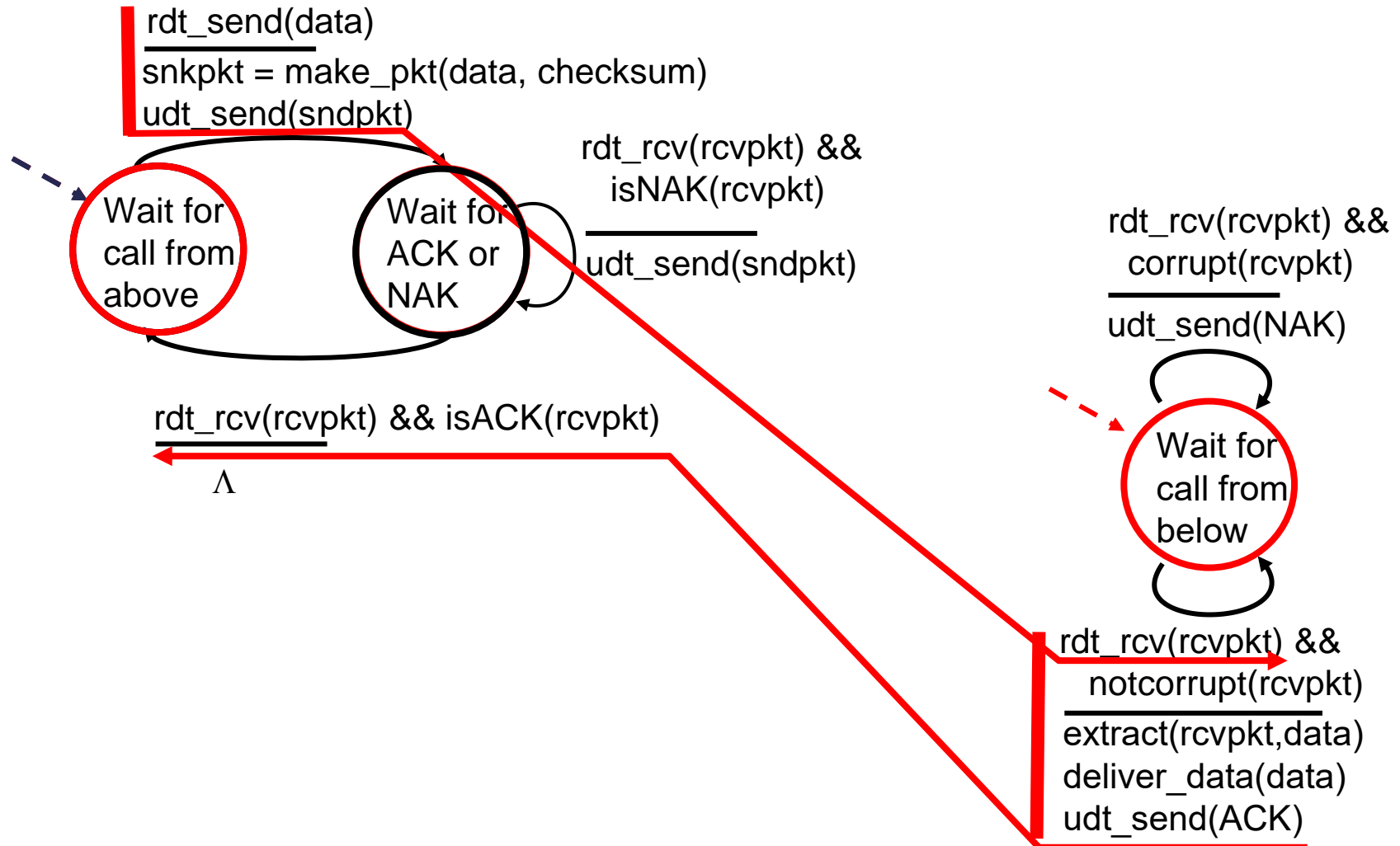
odesilatel

přijemce



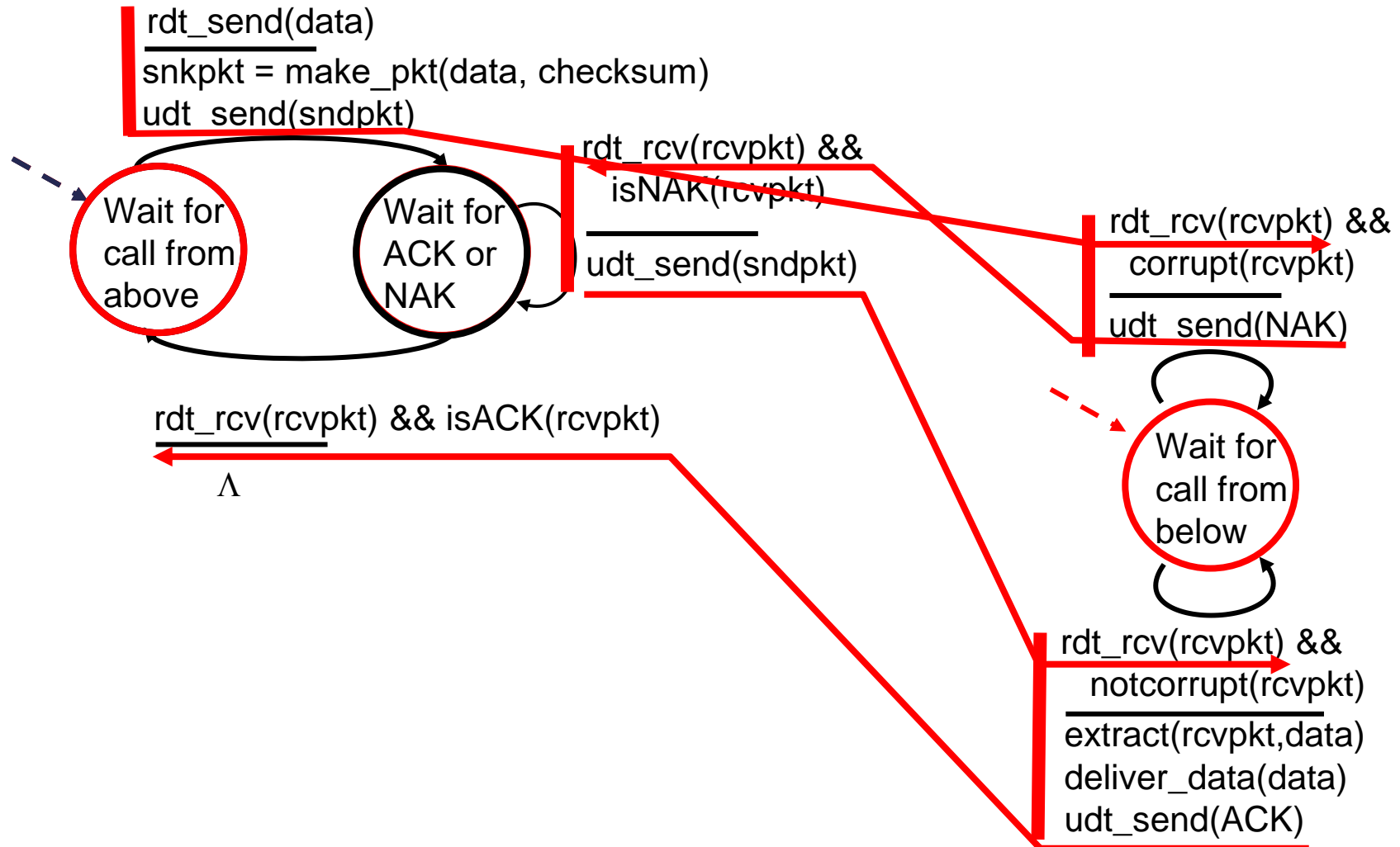
RDT 2.0

Normální chování



RDT 2.0

Příklad s chybou



- *Co když je potvrzení (ACK/NAK) poškozeno?*
 - odesílatel neví, co se na straně příjemce stalo
 - není možné jednoduše poslat znovu paket
 - vznikla by **duplikace paketů** (dat)
- Možné řešení:
 - přidání pořadového čísla do paketu
 - odesílatel pošle znovu paket, jestliže je potvrzení poškozeno
 - příjemce zahodí paket, jedná-li se o duplikát

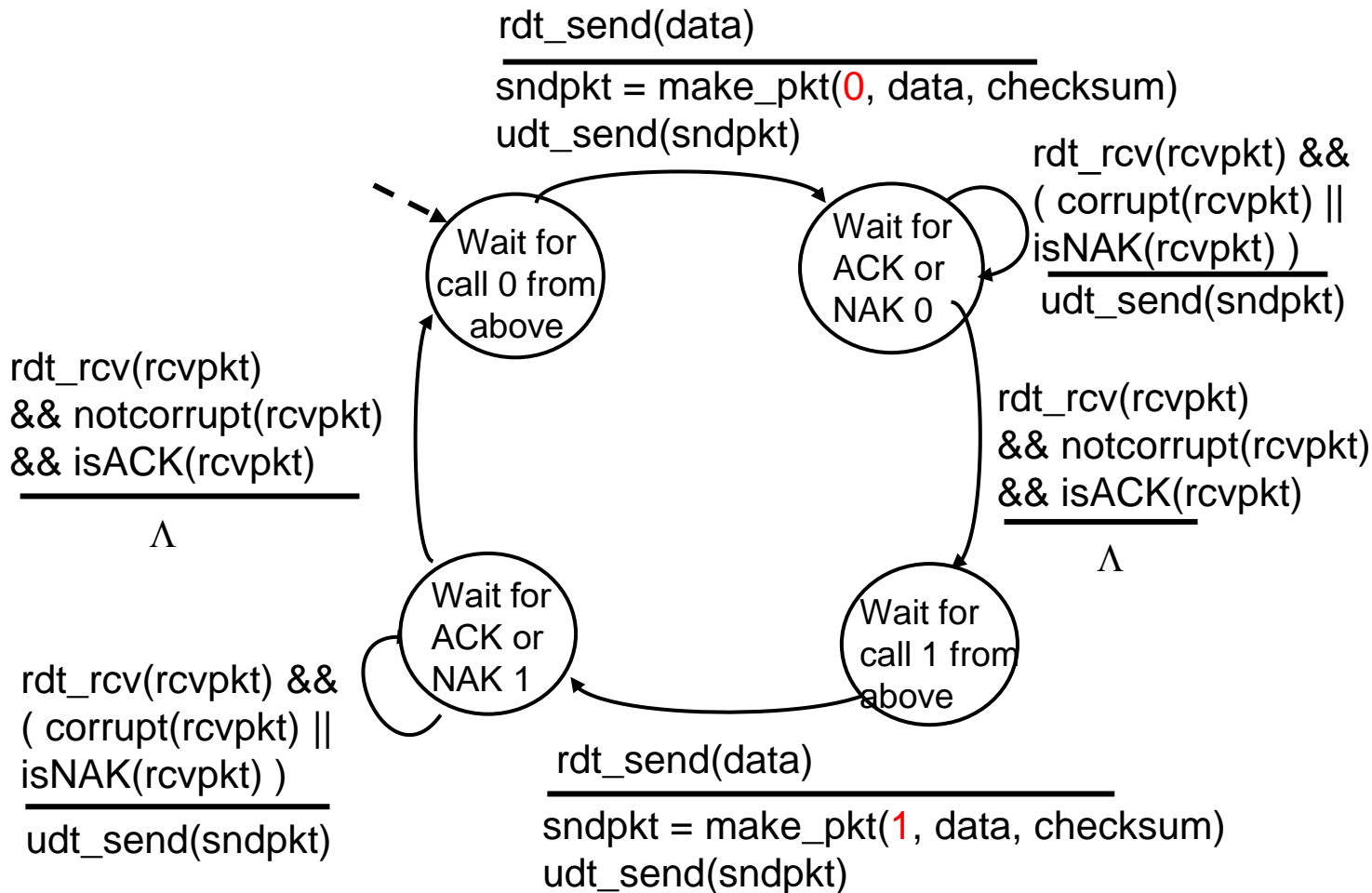
RDT 2.1

Vlastnosti

- Odesílatel
 - pořadové číslo přidáno do paketu
 - *Stačí 0 a 1 pro číslování paketu, proč?*
 - je potřeba kontrolovat zda potvrzení není poškozeno
 - dvakrát více stavů
- Příjemce
 - musí kontrolovat duplicitu příchozích paketů dle pořadového čísla
 - příjemce neví, zda potvrzení přišlo v pořádku odesílateli

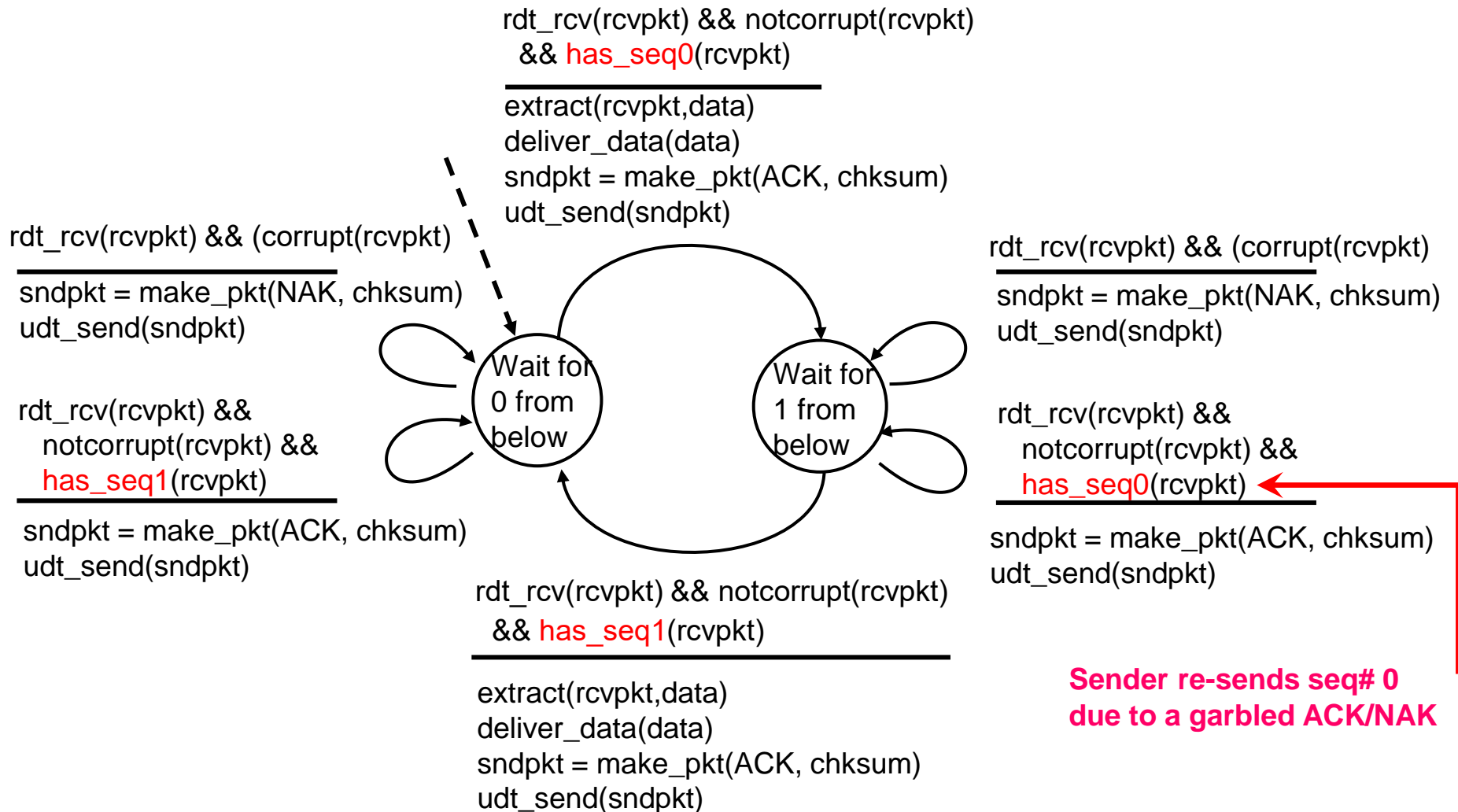
RDT 2.1

Odesílatel



RDT 2.1

Příjemce



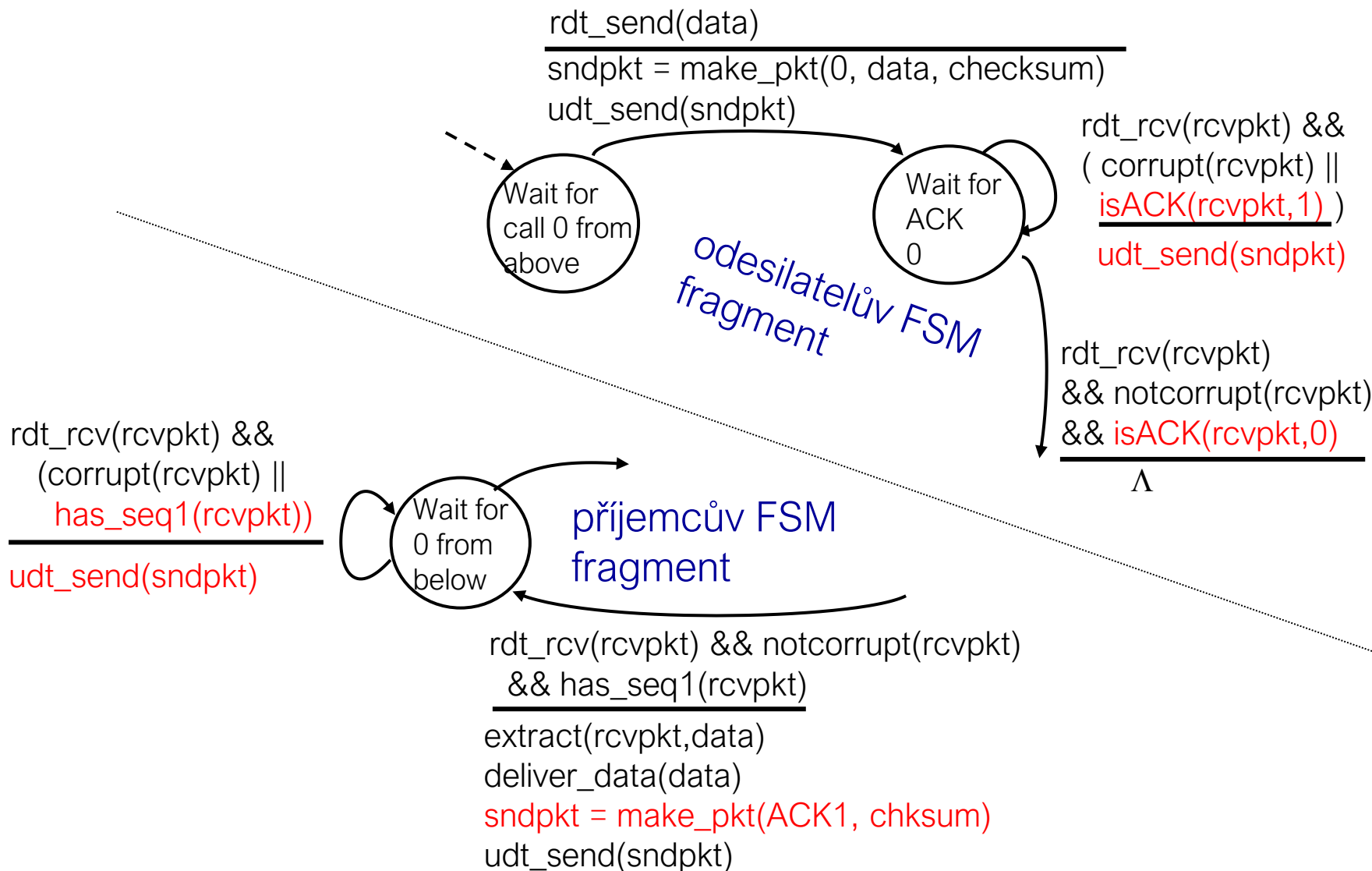
RDT 2.2

Pouze pozitivní potvrzování

- Stejná funkcionality jako RDT 2.1
- místo NAK je použito pouze ACK
 - to vyžaduje uvedení pořadí potvrzovaného paketu v ACK zprávě
- Duplikace ACK má stejnou sémantiku jako příchod NAK

RDT 2.2

FSM



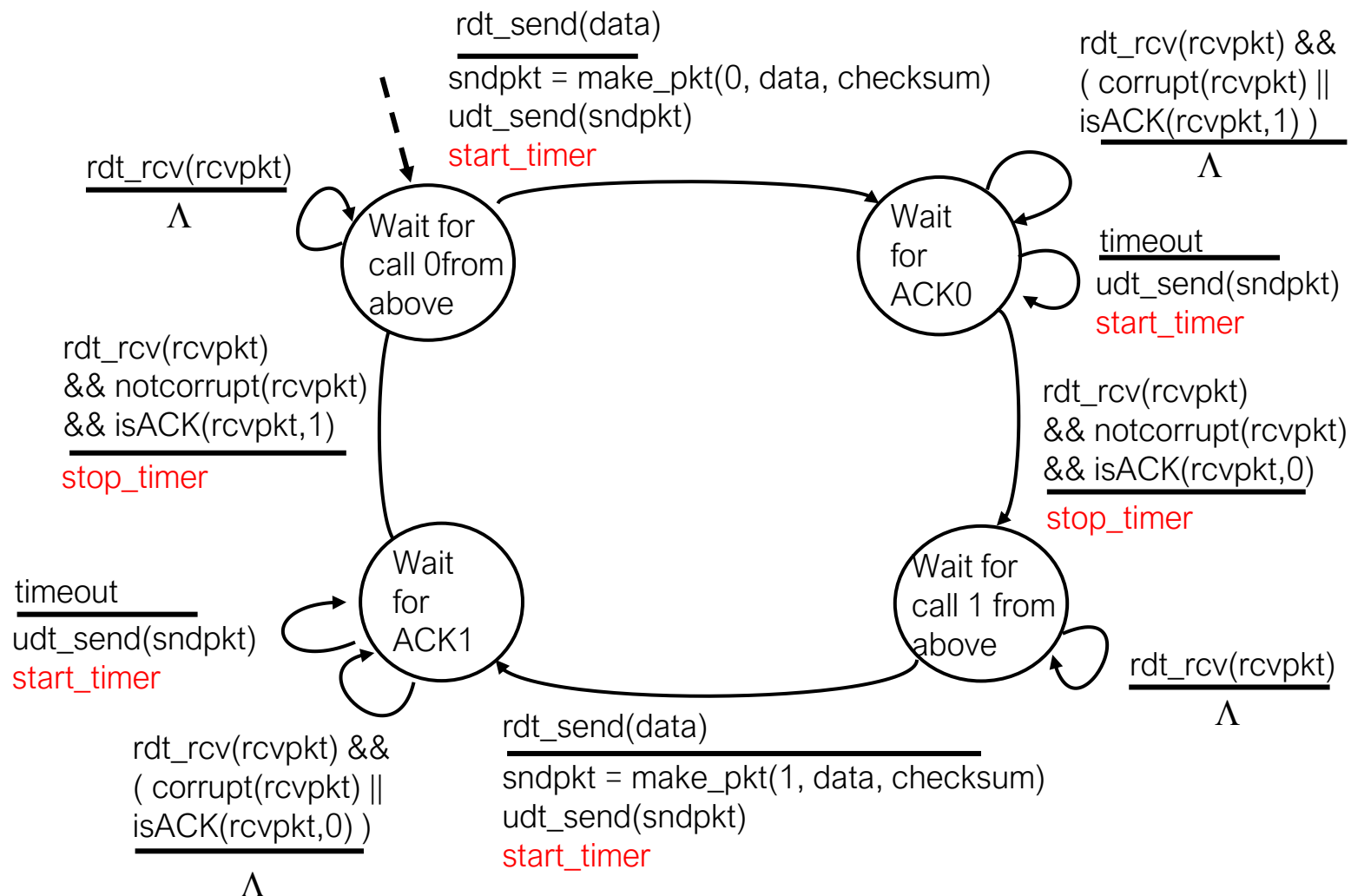
RDT 3.0

(kanál s chybami)

- *Předpoklad: kanál může pakety i ztrácet*
- Řešení
 - odesílatel čeká na potvrzení určitý „rozumný“ čas
 - **znovuzaslání (retransmission)** po vypršení časovače
 - jestliže je paket více pozdržen:
 - znovuzaslání vede na duplikování paketů
 - příjemce musí specifikovat pořadí paketů v potvrzování
 - vyžaduje použití časovače

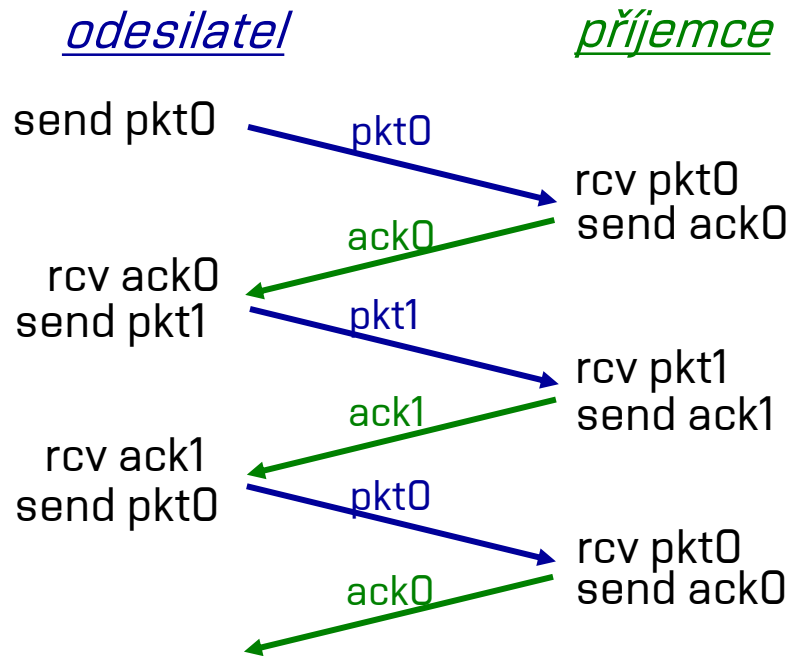
RDT 3.0

Odesilatel

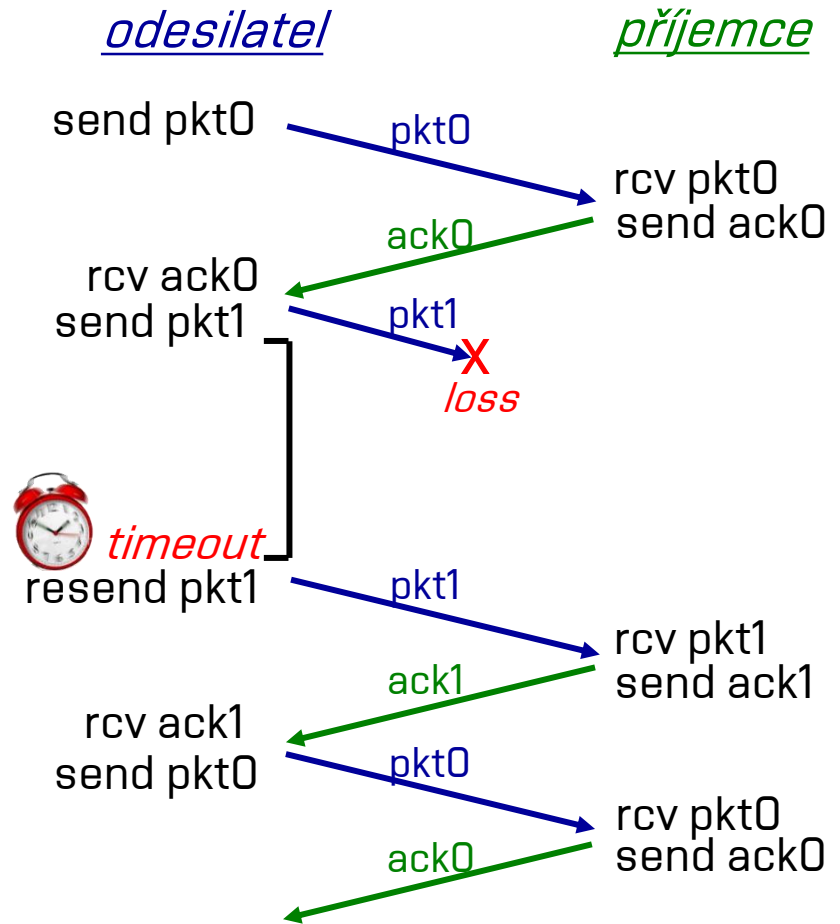


RDT 3.0

Scénáře (bez ztráty a se ztrátou)



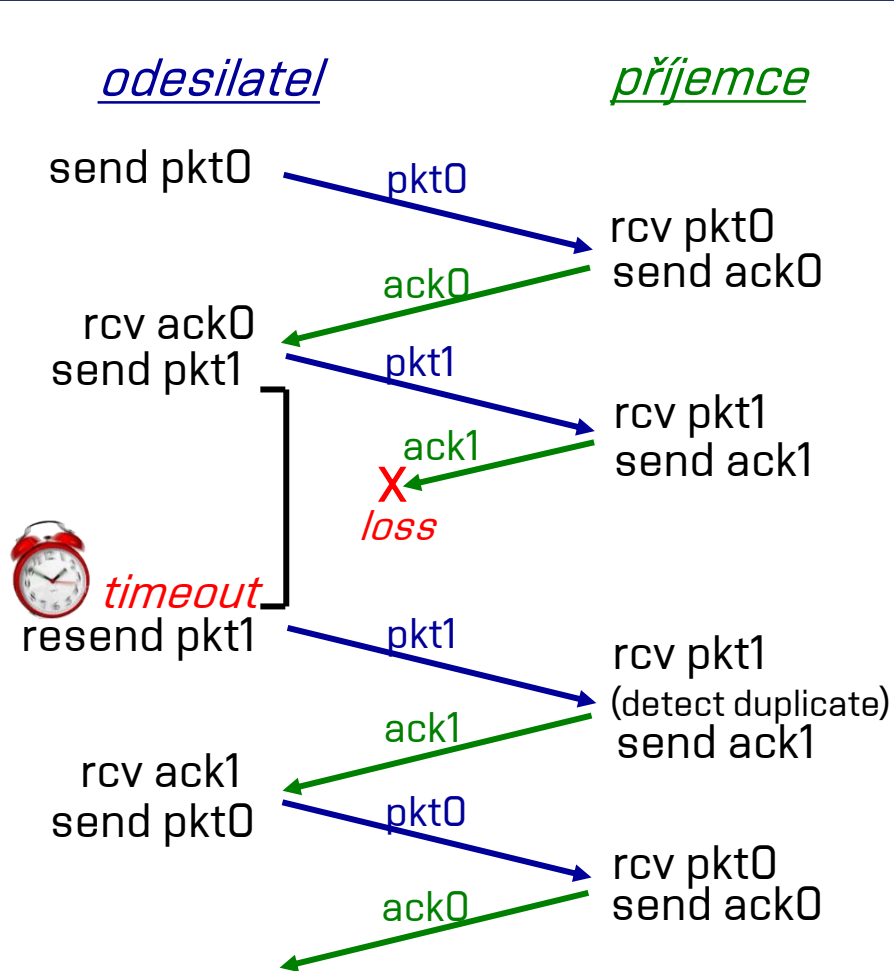
(a) bez ztráty



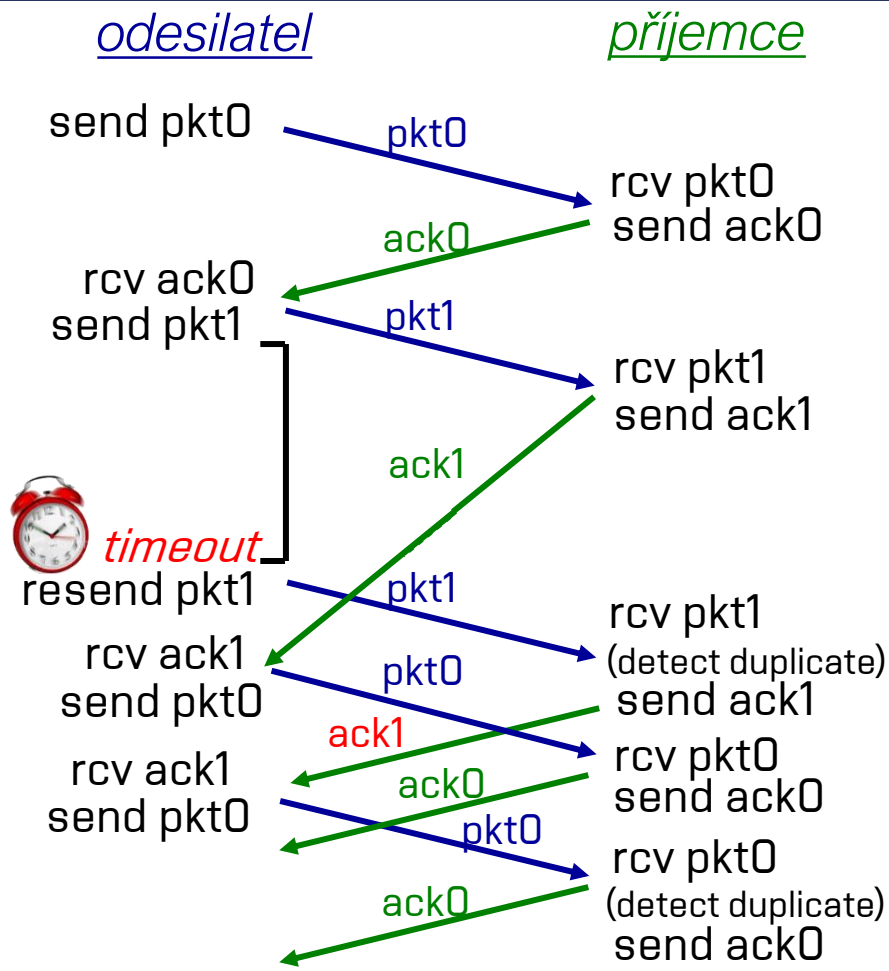
(b) ztráta paketu

RDT 3.0

Scénáře (ztráta potvrzení, zpoždění)



(c) ztráta ACK



(d) vypršení timeoutu / zpoždění ACK

RDT 3.0

Utilizace

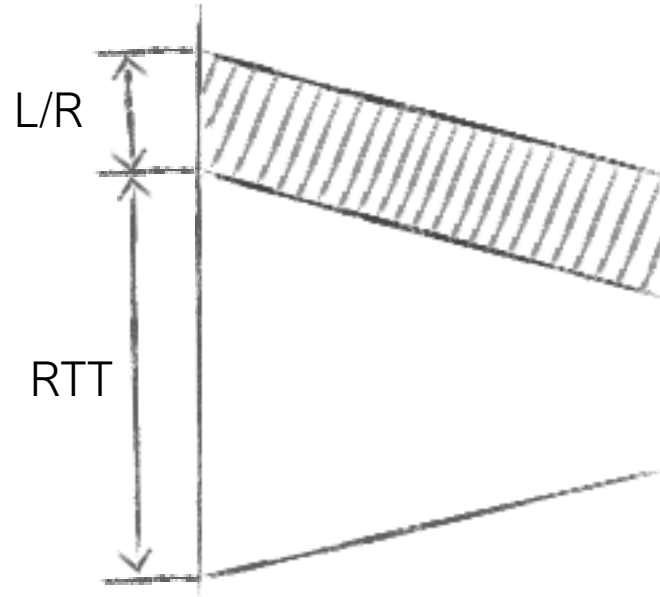
- RDT 3.0 garantuje spolehlivý přenos nad nespolehlivým kanálem
- Utilizace ovšem není vysoká:

$$U = \frac{L/R}{RTT + L/R}$$

L - velikost paketu [bits]
R – přenosová rychlost [bps]
RTT – round trip time
U – využití

Například:

- linka 1Gbps
- pakety 1kb ~ 8kb
- RTT = 30ms
- $U = 0,027\% \rightarrow 1\text{kb}$
každých 30ms dává
264kb/s na 1Gbps lince!



Obsah

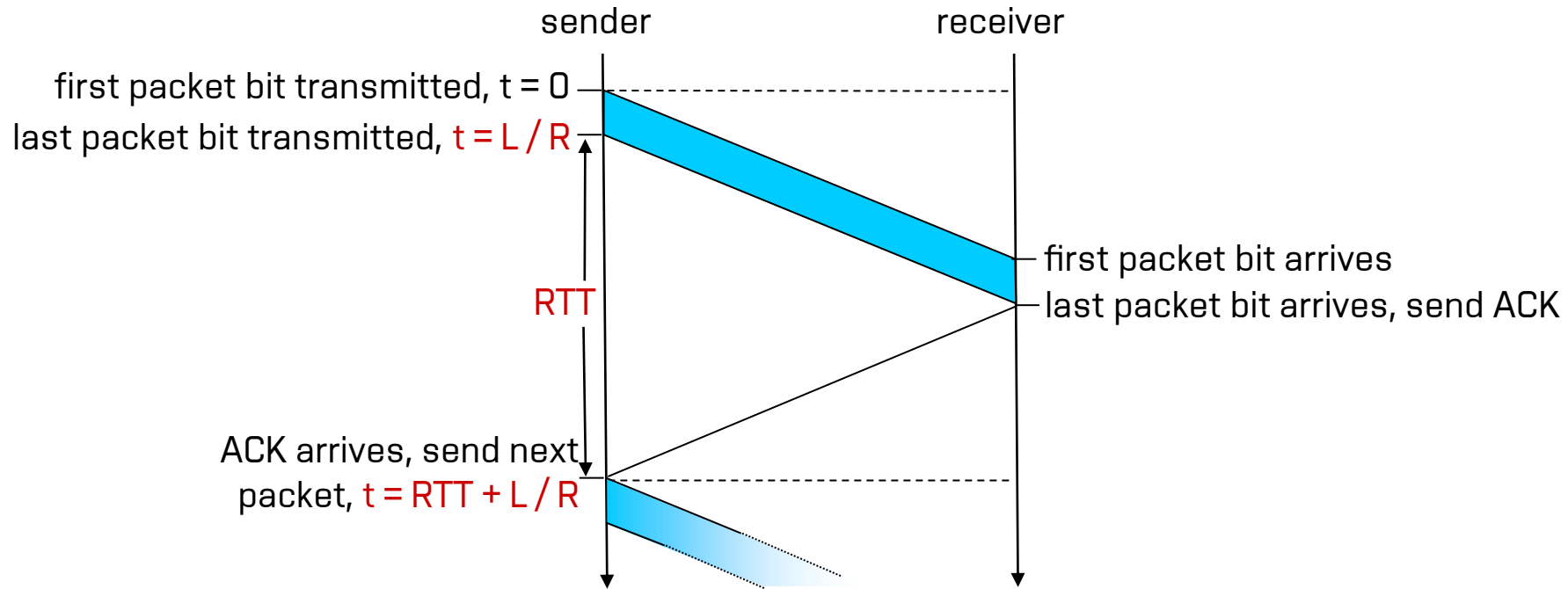
1) PROTOKOL UDP

2) SPOLEHLIVÝ PŘENOS

- Principy spolehlivého přenosu
- Stop-and-wait
- Pipelined protokoly
- Go-Back-N
- Selective Repeat

3) PROTOKOL TCP

Stop-and-wait (=RDT 3.0)

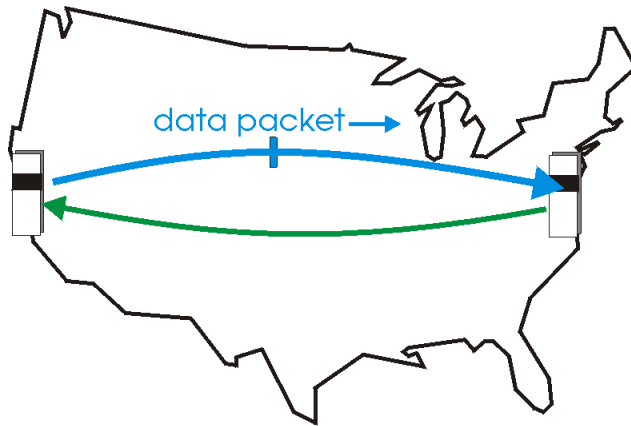


$$U = \frac{L/R}{RTT + L/R}$$

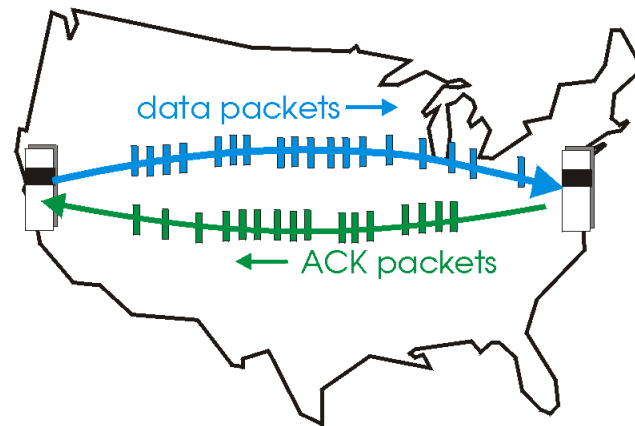
U – využití, tedy poměr času stráveného odesíláním dat ku celkovému času komunikace.

Zřetězené protokoly

- Zřetězení (**pipelining**) umožňuje poslat více paketů, aniž přišlo potvrzení pro předchozí
 - číslování paketů nemůže být pouze 0 a 1
 - vyrovnávací paměti jsou potřeba u odesílatele i příjemce
- Dva druhy **zřetězených protokolů**:
 - go-back-N
 - selective-repeat

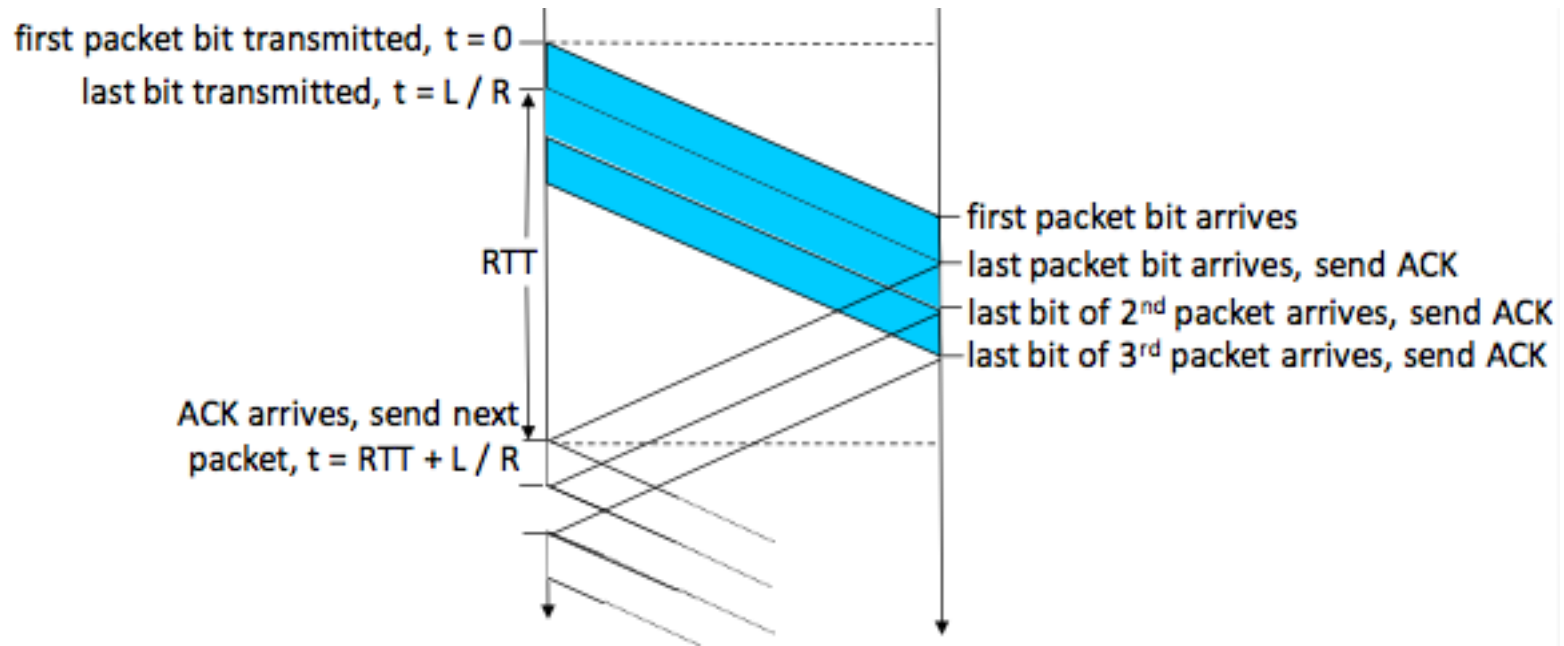


(a) a stop-and-wait protocol in operation



(b) a pipelined protocol in operation

Utilizace Pipeliningu



$$U = \frac{3 \times L / R}{RTT + L / R}$$

Přehled zřetězených protokolů

Go-back-N

- odesílatel
 - si udržuje v bufferu N nepotvrzených paketů
- příjemce
 - odesílá **kumulativní potvrzení**
 - neodesílá další ACK, pokud detektuje ztrátu
- odesílatel
 - nejstarší nepotvrzený paket má časovač
 - když vyprší, odešlou se všechny nepotvrzené pakety

Selective Repeat

- odesílatel
 - si udržuje v bufferu N nepotvrzených paketů
- příjemce
 - odesílá **individuální potvrzení** za každý paket
- odesílatel
 - každý nepotvrzený paket má vlastní časovač
 - když vyprší, odešle se daný nepotvrzené pakety

Obsah

1) PROTOKOL UDP

2) SPOLEHLIVÝ PŘENOS

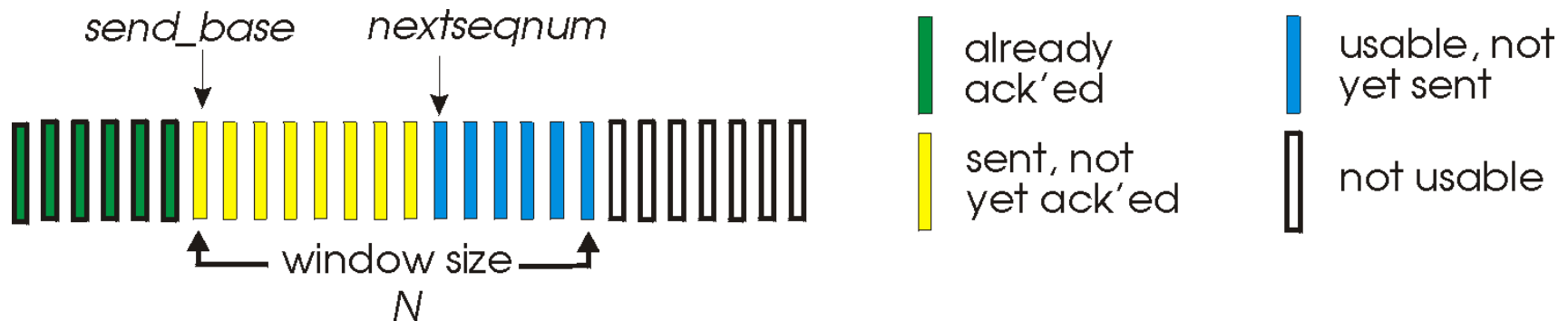
- Principy spolehlivého přenosu
- Stop-and-wait
- Pipelined protokoly
- Go-Back-N
- Selective Repeat

3) PROTOKOL TCP

Go-back-N

Odesílatel:

- k -bitové sekvenční číslo pro každý paket
- „okno“ o velikosti až N pro zatím nepotvrzené pakety
- $ACK(n)$ – potvrzuje všechny nepotvrzené pakety $\leq n$
- vyžaduje časovač pro každý paket



Go-back-N chování

sender window (N=4)

012345678
012345678
012345678
012345678

012345678
012345678

012345678
012345678
012345678
012345678

odesílatel

send pkt0
send pkt1
send pkt2
send pkt3
(wait)

rcv ack0, send pkt4
rcv ack1, send pkt5

ignore duplicate ACK



pkt 2 timeout

send pkt2
send pkt3
send pkt4
send pkt5

příjemce

receive pkt0, send ack0
receive pkt1, send ack1

receive pkt3, discard,
(re)send ack1

receive pkt4, discard,
(re)send ack1

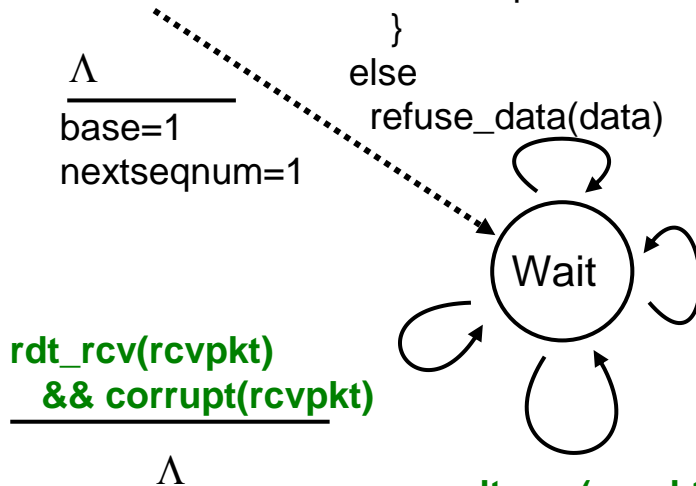
receive pkt5, discard,
(re)send ack1

rcv pkt2, deliver, send ack2
rcv pkt3, deliver, send ack3
rcv pkt4, deliver, send ack4
rcv pkt5, deliver, send ack5

GBN
Odeslatel

rdt_send(data)

```
if (nextseqnum < base+N) {
    sndpkt[nextseqnum] = make_pkt(nextseqnum,data,chksum)
    udt_send(sndpkt[nextseqnum])
    if (base == nextseqnum)
        start_timer
    nextseqnum++
}
else
    refuse_data(data)
```



timeout

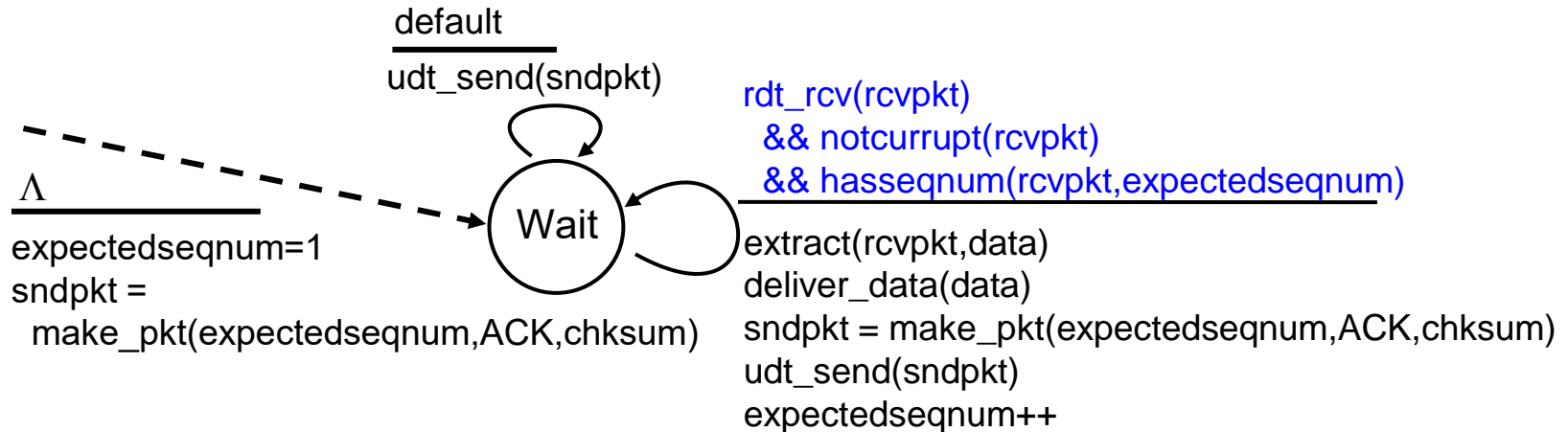
```
start_timer
udt_send(sndpkt[base])
udt_send(sndpkt[base+1])
...
udt_send(sndpkt[nextseqnum-1])
```

```
rdt_rcv(rcvpkt) &&  
notcorrupt(rcvpkt)
```

```
base = getacknum(rcvpkt)+1
If (base == nextseqnum)
    stop_timer
else
    start_timer
```


GBN

Příjemce



- Může duplikovat ACK
- Pakety mimo pořadí (**out-of-order**):
 - odpověz ACK s seqno posledního korektně přijatého
 - zahazuje (není potřeba buffer)

Obsah

1) PROTOKOL UDP

2) SPOLEHLIVÝ PŘENOS

- Principy spolehlivého přenosu
- Stop-and-wait
- Pipelined protokoly
- Go-Back-N
- **Selective Repeat**

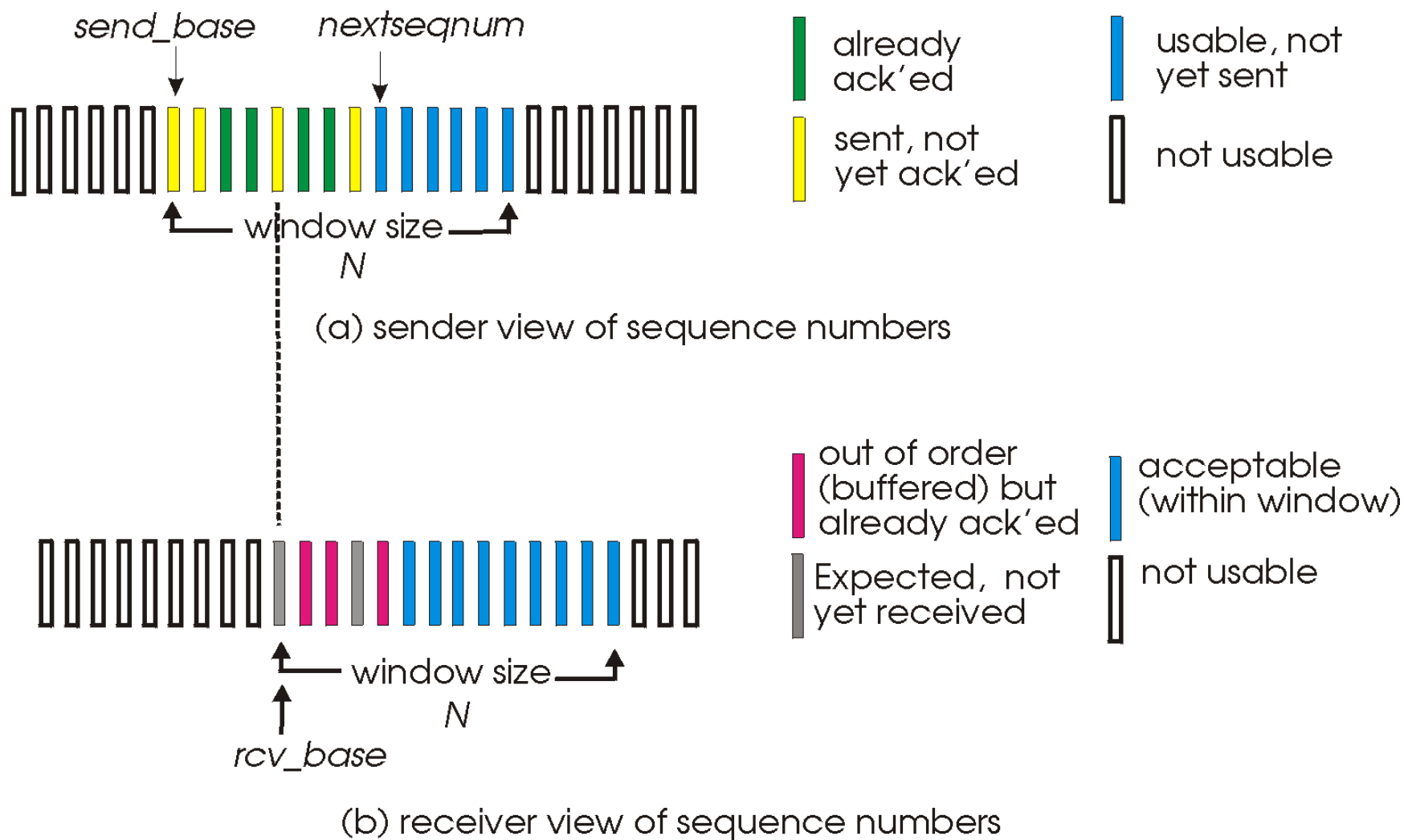
3) PROTOKOL TCP

Selective Repeat

Obecně

- Příjemce selektivně potvrzuje všechny správně přijaté pakety
 - Pakety jsou uchovány ve vyrovnávací paměti, kde po doplnění jsou předány vyšší vrstvě
- Odesílatel znovuzasílá pouze pakety pro které neobdržel potvrzení
 - časovač pro každý nepotvrzený paket
- Okénko odesílatele
 - N po sobě jdoucích pořadových čísel
 - podobně jako Go-back-N, omezuje maximální množství nepotvrzených dat
- Výhoda
 - oproti Go-back-N je, že není nutné znovu posílat celý rozsah, tedy posílají se pouze nepotvrzené pakety

Selective Repeat Sliding Windows



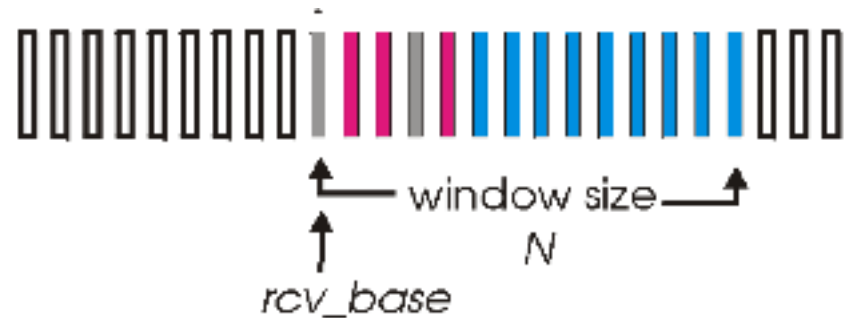
Selective Repeat FSM

Odesílatel

- 1) data od vyšší vrstvy:
 - je-li volné místo v okně, pošli paket a nastav časovač
- 2) timeout(n):
 - znovu pošli paket n
 - restart časovače
- 3) ACK(n):
 - označ paket n jako doručený, když je n nejmenší nepotvrzený, posuň okno

Příjemce - příchozí paket # n

- 1) $n \in (\text{rcv_base}, \text{rcv_base} + N - 1)$:
 - pošli ACK(n)
 - ulož do bufferu, jedná-li se o první, potom doruč data a posuň okno
- 2) $n \in (\text{rcv_base} - N, \text{rcv_base} - 1)$:
 - pošli ACK(n)
- 3) jinak:
 - ignoruj paket



Selective Repeat Demo

sender window (N=4)

012345678

012345678

012345678

012345678

012345678

012345678

012345678

012345678

012345678

012345678

sender

send pkt0

send pkt1

send pkt2

send pkt3

(wait)

rcv ack0, send pkt4

rcv ack1, send pkt5

record ack3 arrived



pkt 2 timeout

send pkt2

record ack4 arrived

record ack5 arrived

receiver

receive pkt0, send ack0

receive pkt1, send ack1

receive pkt3, buffer,
send ack3

receive pkt4, buffer,
send ack4

receive pkt5, buffer,
send ack5

rcv pkt2; deliver pkt2,
pkt3, pkt4, pkt5; **send ack2**

X loss

Q: Co se stane, když ack2 nedorazí?

Spolehlivý přenos - principy

Mechanismus	Použití
Kontrolní součet	Detekce bitových chyb v přenášených segmentech.
Časovač	Pro detekci ztráty segmentů nebo jejich ACK.
Sekvenční číslo	Číslování segmentů. Detekce duplicit segmentů a segmentů, které chybi.
Potvrzení	Pro informaci odesílatele, že segment byl úspěšně doručen.
Negativní potvrzení	Upozornění odesílatele, že segment nebyl správně doručen.
Okno Zřetězení	Podporuje zasílání více paketů, bez nutnosti čekat na jejich potvrzení. Zvyšuje efektivitu přenosu.

Obsah

1) PROTOKOL UDP

- Funkce, formát rámce

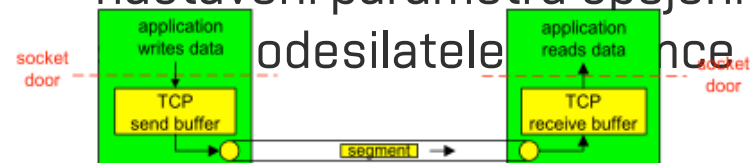
2) SPOLEHLIVÝ PŘENOS

3) PROTOKOL TCP

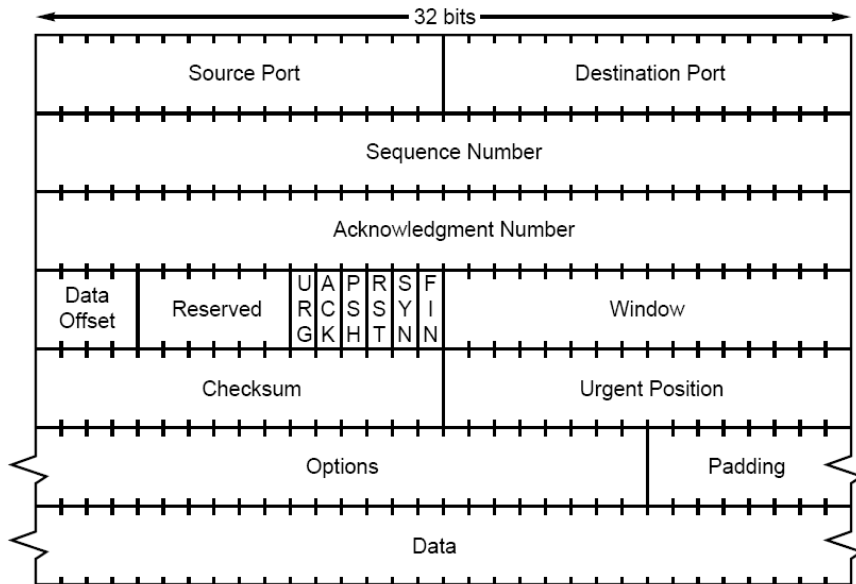
- Vlastnosti a funkce, formát segmentu
- Řízení spojení
- Řízení toku
- Řízení zahlcení

Transmission Control Protocol

- TCP [[RFC 793](#)]
- spojení **point-to-point**
 - 1 odesílatel a 1 příjemce
- spolehlivý přenos dat
 - **zachování pořadí bytů (in-order byte stream)**
 - bez omezení velikosti přenášených dat
- zřetězený přenos
 - window size zajišťuje řízení toku a obranu vůči zahlcení
- vyrovnávací paměť pro odesílaná i přijímaná data
- plně duplexní přenos
 - obousměrný (full-duplex) přenos dat v jednom spojení
- spojově orientovaná služba (**connection-oriented service**)
- vytvoření spojení výměnou inicializačních zpráv
- nastavení parametrů spojení na odesilatele a příjemce



Segment



Příznaky (Flags)

- **URG (Urgent data)**: indikuje přenášení urgentních dat
- **ACK (Acknowledge)**: indikuje platné číslo potvrzení
- **PSH (Push data)**: indikuje požadavek co nejrychlejšího doručení dat aplikační vrstvě
- **RST (Reset)**: indikuje požadavek na reset virtuálního spojení, aplikace mají spojení ukončit, nepředstavuje standardní prostředek ukončení komunikace
- **SYN (Synchronize)**: požadavek na vytvoření spojení
- **FIN (Finish)**: požadavek na ukončení spojení

- **Velikost hlavičky (Data offset)**: počet 32-bitových slov hlavičky, zahrnující volitelné volby (**Options**) a zarovnání (**Padding**)
 - Segment bez volby, data offset=5
- **Window**: velikost klouzavého okna
- **Kontrolní součet (Checksum)**: jedničkový doplněk (negace) součtu 16-bitových slov
- **Pozice urgentních dat (Urgent data)**: číslo oktetu v datové části, která mají být co okamžitě doručena aplikaci od začátku segmentu (bez ohledu na pořadí oktětů)
- **Volby (Options)**: volitelné parametry spojení, MSS: maximální velikost
- **Padding**: vycpávka do mocniny 2
- **Data**: aplikační data

TCP ve Wiresharku

Filter: tcp Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
101	1.444853000	147.229.181.83	147.229.181.30	TCP	54	59595→56216 [ACK] Seq=26535 Ack=1045 w
104	1.760224000	147.229.181.83	54.152.141.6	TCP	55	58483→80 [ACK] Seq=1 Ack=1 win=254 Len
106	1.909554000	147.229.181.30	147.229.181.83	TCP	83	56216→59595 [PSH, ACK] Seq=1045 Ack=26
107	1.909978000	54.152.141.6	147.229.181.83	TCP	66	80→58483 [ACK] Seq=1 Ack=2 win=136 Len
108	1.942186000	147.229.181.83	188.165.233.56	TCP	138	64878→8008 [PSH, ACK] Seq=1 Ack=1 win=
109	1.945318000	147.229.181.83	147.229.181.30	TCP	96	59595→56216 [PSH, ACK] Seq=26535 Ack=1
110	1.946122000	147.229.181.30	147.229.181.83	TCP	86	56216→59595 [PSH, ACK] Seq=1045 Ack=26

Frame 104: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface 0

Ethernet II, Src: AsustekC_8a:95:6e (78:24:af:8a:95:6e), Dst: ExtremeN_1d:4e:30 (00:04:96:1d:4e:30)

Internet Protocol Version 4, Src: 147.229.181.83 (147.229.181.83), Dst: 54.152.141.6 (54.152.141.6)

Transmission Control Protocol, Src Port: 58483 (58483), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 1

Source Port: 58483 (58483)
Destination Port: 80 (80)
[Stream index: 2]
[TCP segment Len: 1]
Sequence number: 1 (relative sequence number)
[Next sequence number: 2 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Header Length: 20 bytes

.... 0000 0001 0000 = Flags: 0x010 (ACK)
window size value: 254
[calculated window size: 254]
[window size scaling factor: -1 (unknown)]

Checksum: 0x0cf3 [validation disabled]
urgent pointer: 0

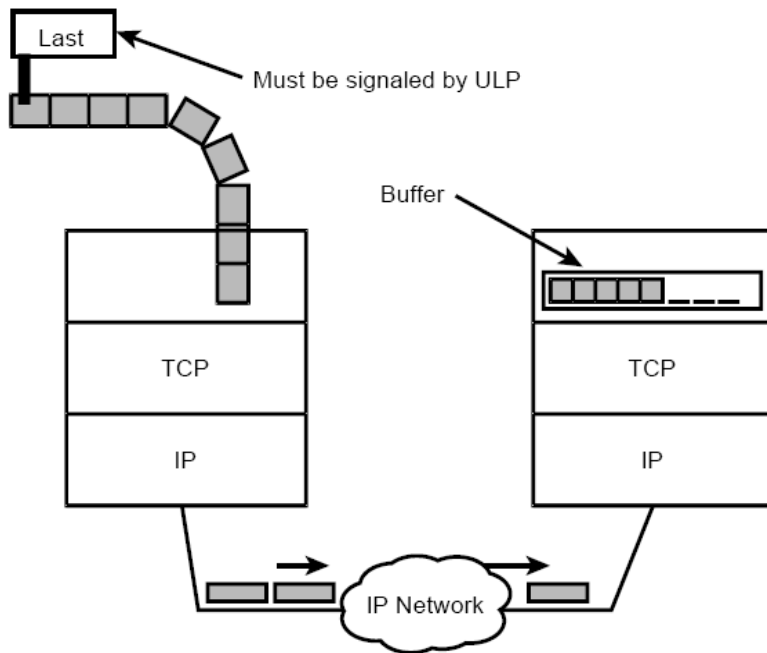
[SEQ/ACK analysis]

0000 00 04 96 1d 4e 30 78 24 af 8a 95 6e 08 00 45 00NOx\$...n..E.
0010 00 29 51 1d 40 00 80 06 00 00 93 e5 b5 53 36 98 ..)Q.@... ..S6.
0020 8d 06 e4 73 00 50 bf bf a4 6f 3a 47 a7 27 50 10 ...S.P.. .O:G..P.
0030 00 fe 0c f3 00 00 00
.....

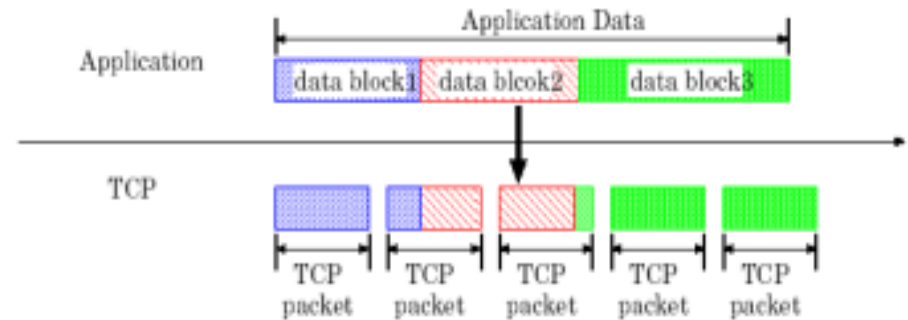
Transmission Control Protocol (tcp), 20 bytes Packets: 527 · Displayed: 477 (90,5%) · Dropped: 0 (0,0%) Profile: Default

In-order Byte Stream

- Data přenášená TCP segmentem nemají žádnou strukturu
 - představuje proud bitů/bytů
 - rozděluje data od aplikace do posloupnosti segmentů
 - velikost těchto segmentů je určena TCP vrstvou
 - TCP určuje vhodnou velikost segmentu pro každý směr komunikace.



- Octet-stream orientation
– No inherent notion of block of data

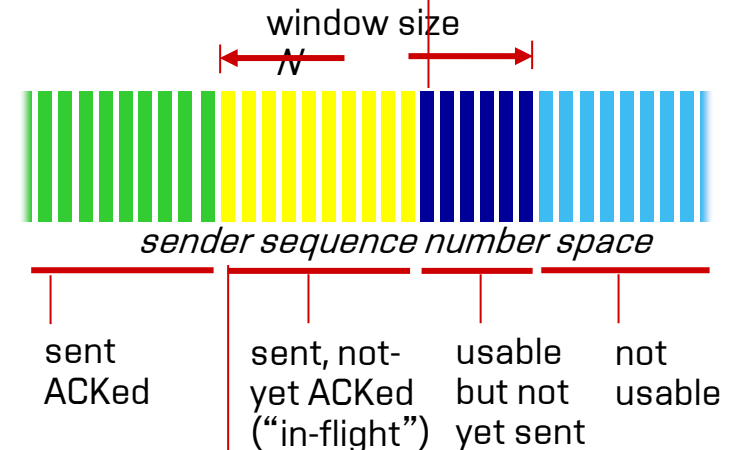


TCP SEQ a ACK čísla

- **Sekvenční číslo (sequence #)**
 - číslo prvního B odesílaného segmentu
- **Potvrzovací číslo (acknowledgement #)**
 - číslo prvního bytu očekávaného segmentu k přijetí
 - TCP umí kumulativní potvrzování

outgoing segment from sender

source port #	dest port #
sequence number	
acknowledgement number	
	rwnd
checksum	urg pointer

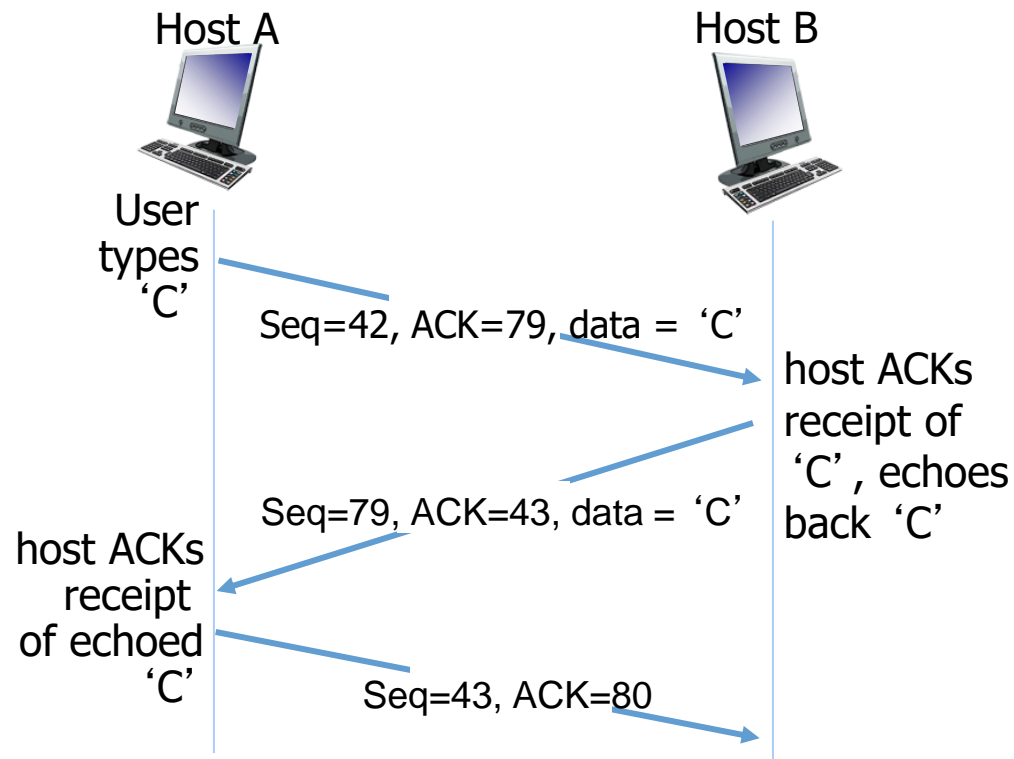


incoming segment to sender

source port #	dest port #
sequence number	
acknowledgement number	
	A
checksum	urg pointer

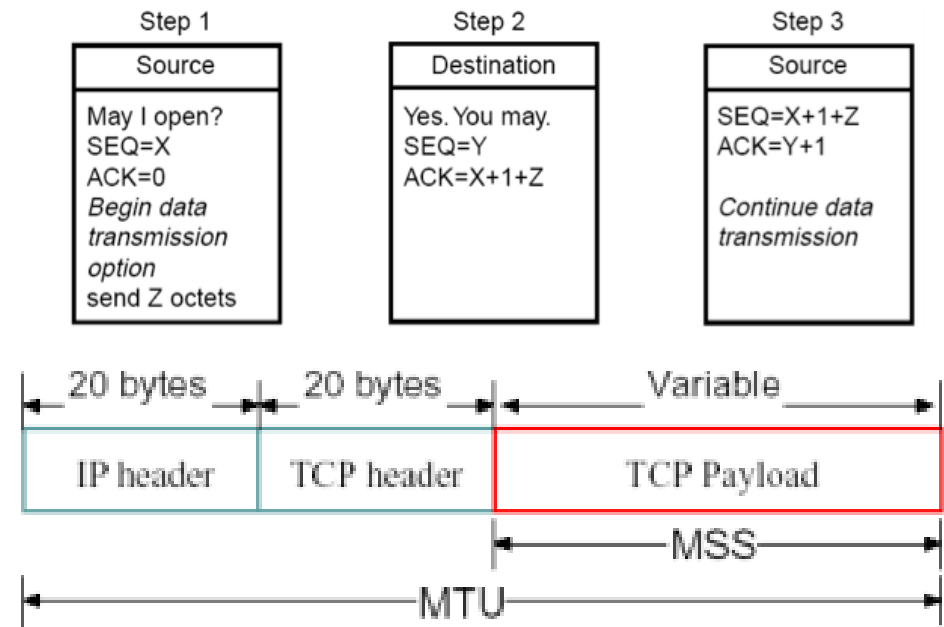
Příklad potvrzování TCP

- Telnet posílá/potvrzuje právě jeden znak



Úvodní parametry ppojení

- **Initial Sequence Number (ISN)** na obou stranách
 - RFC určuje inkrement ISN každé 4 mikrosekundy
 - Každých 4,5 hodiny přetečení 32-bitového čítače
- **MSL (Maximum Segment Lifetime)**
 - 2 minuty, často 30 s
- **MSS (Maximum Segment Size)**
 - Maximální hodnota pro data TCP segmentu podle typu LAN technologie



Volby

- RFC 793 definuje pouze tři volby
- RFC 1323 definuje další
- Jednooktetová vs. víceoktetové
- TLV design of protocols

➤ End of option list

Kind	Len	Values...
------	-----	-----------

0

➤ No operation

Options are usually 4 byte aligned with leading NOPs

1

➤ Maximum (Receive) Segment Size [SYN only]

2	4	MSS
---	---	-----

➤ Window Scale Factor [SYN only]

NOP	3	3	shift
-----	---	---	-------

➤ Timestamp

NOP	NOP	8	10	Timestamp Value	Timestamp EchoReply
-----	-----	---	----	-----------------	---------------------

➤ Selective ACK Permitted [SYN packet only]

NOP	NOP	4	2
-----	-----	---	---

➤ Selective ACK block

NOP	NOP	5	L	Left Edge	Right Edge
-----	-----	---	---	-----------	------------

...

$L = 2 + N * 8, N \text{ is number of left-right pairs}$



Obsah

1) PROTOKOL UDP

- Funkce, formát rámce

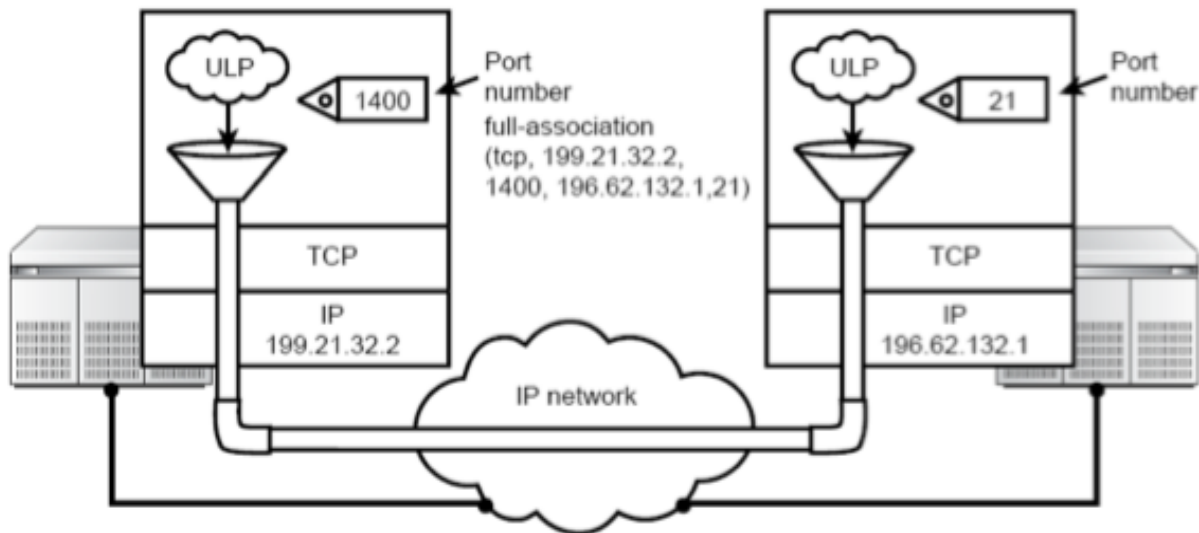
2) SPOLEHLIVÝ PŘENOS

3) PROTOKOL TCP

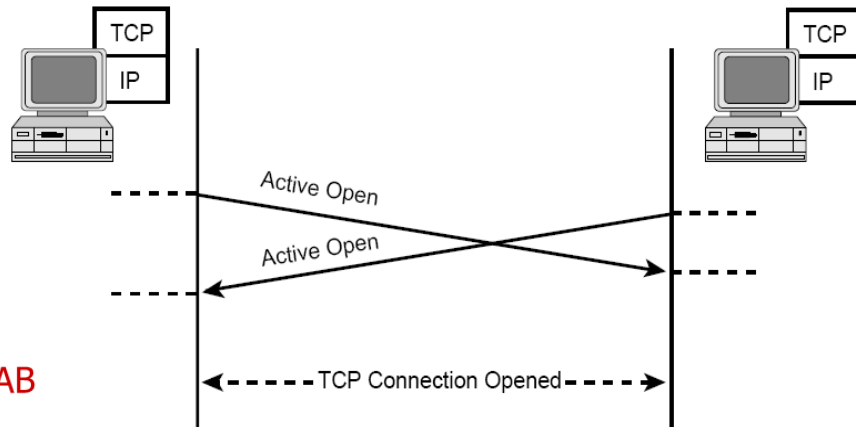
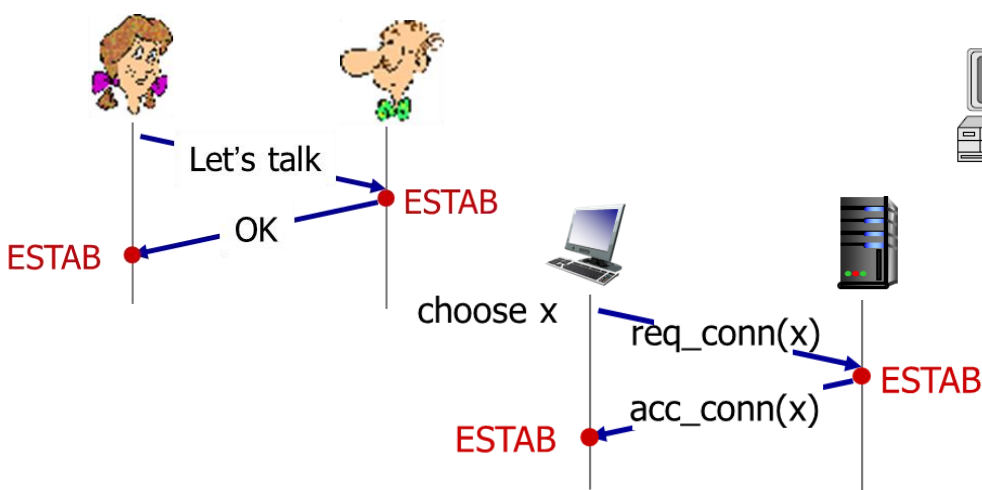
- Vlastnosti a funkce, formát segmentu
- Řízení spojení
- Řízení toku
- Řízení zahlcení

Datový tok (flow)

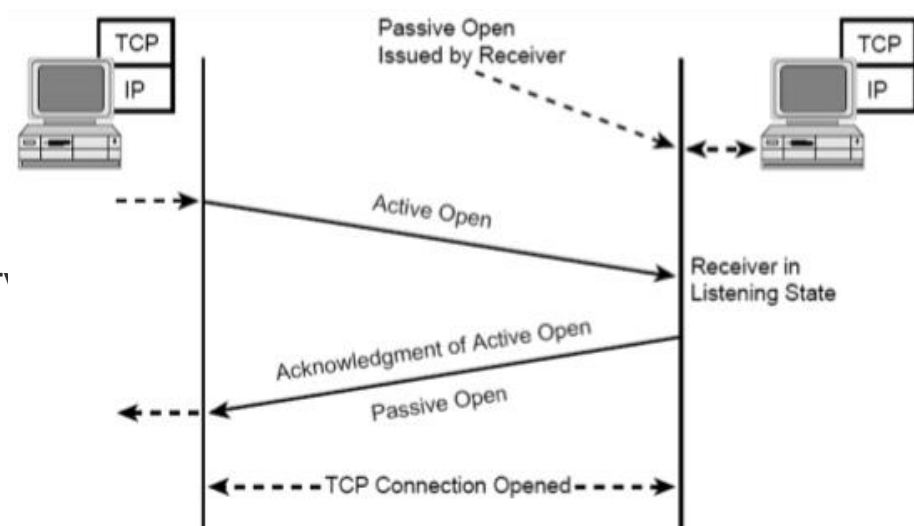
- Poloviční asociace (half association)
 - protokol transportní vrstvy + IP adresa + číslo portu
- Plná asociace (full association)
 - protokol transportní vrstvy + zdrojová IP adresa + zdrojové číslo portu + cílová IP adresa + cílové číslo portu



Zahájení spojení

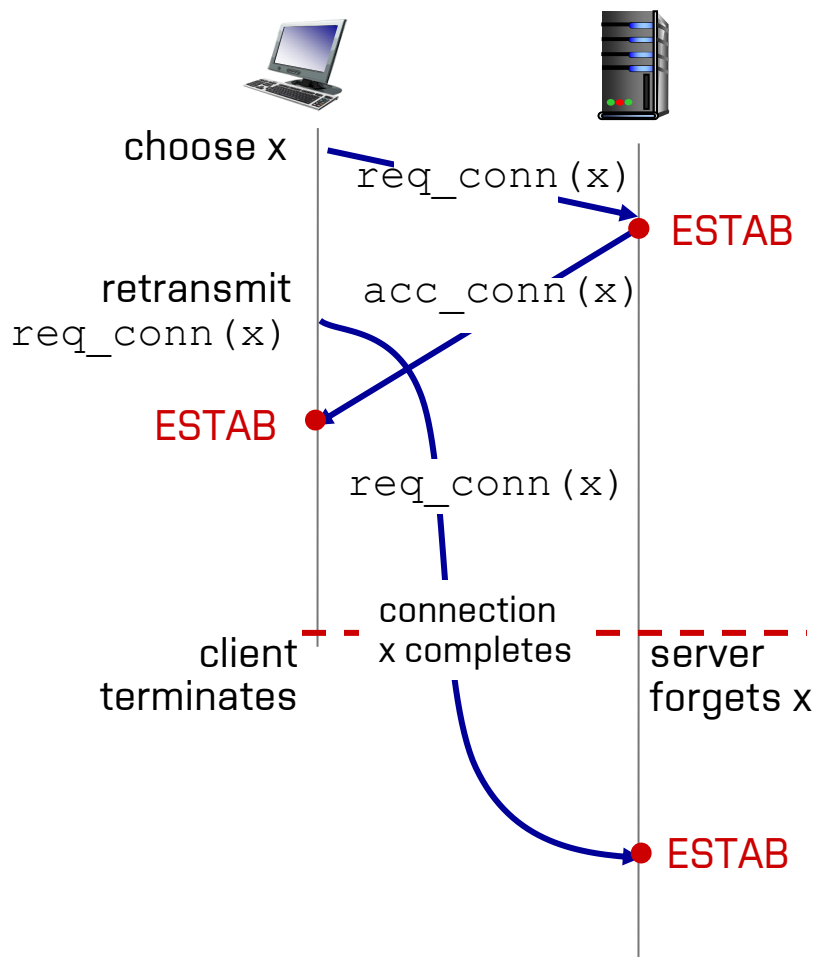


- Operace otevření spojení OPEN
 - Aktivní (Active OPEN)
iniciuje spojení (typicky klient)
 - Pasivní (Passive OPEN)
čeká na vytvoření spojení (typicky ser)
- Možné vytvořit spojení
 - [Active-Passive]
 - [Active-Active] (distribuované systémy)



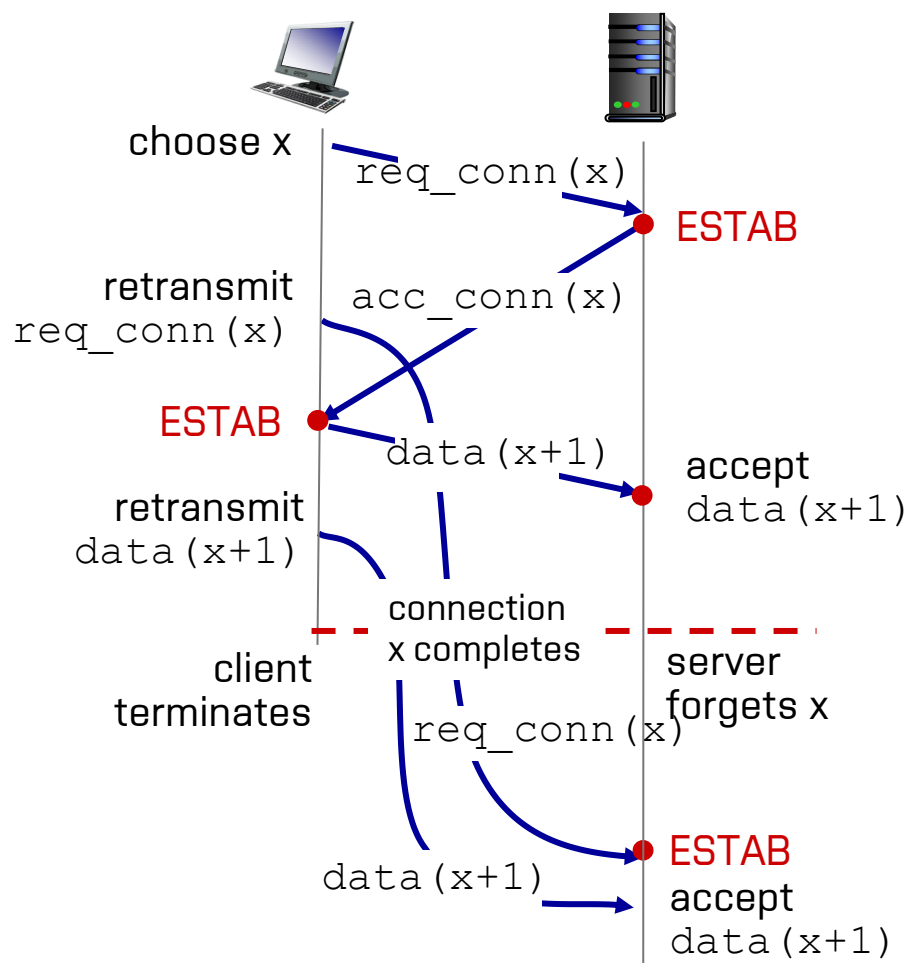
2-Way Handshake (je problematický)

a) zatoulaný retransmit



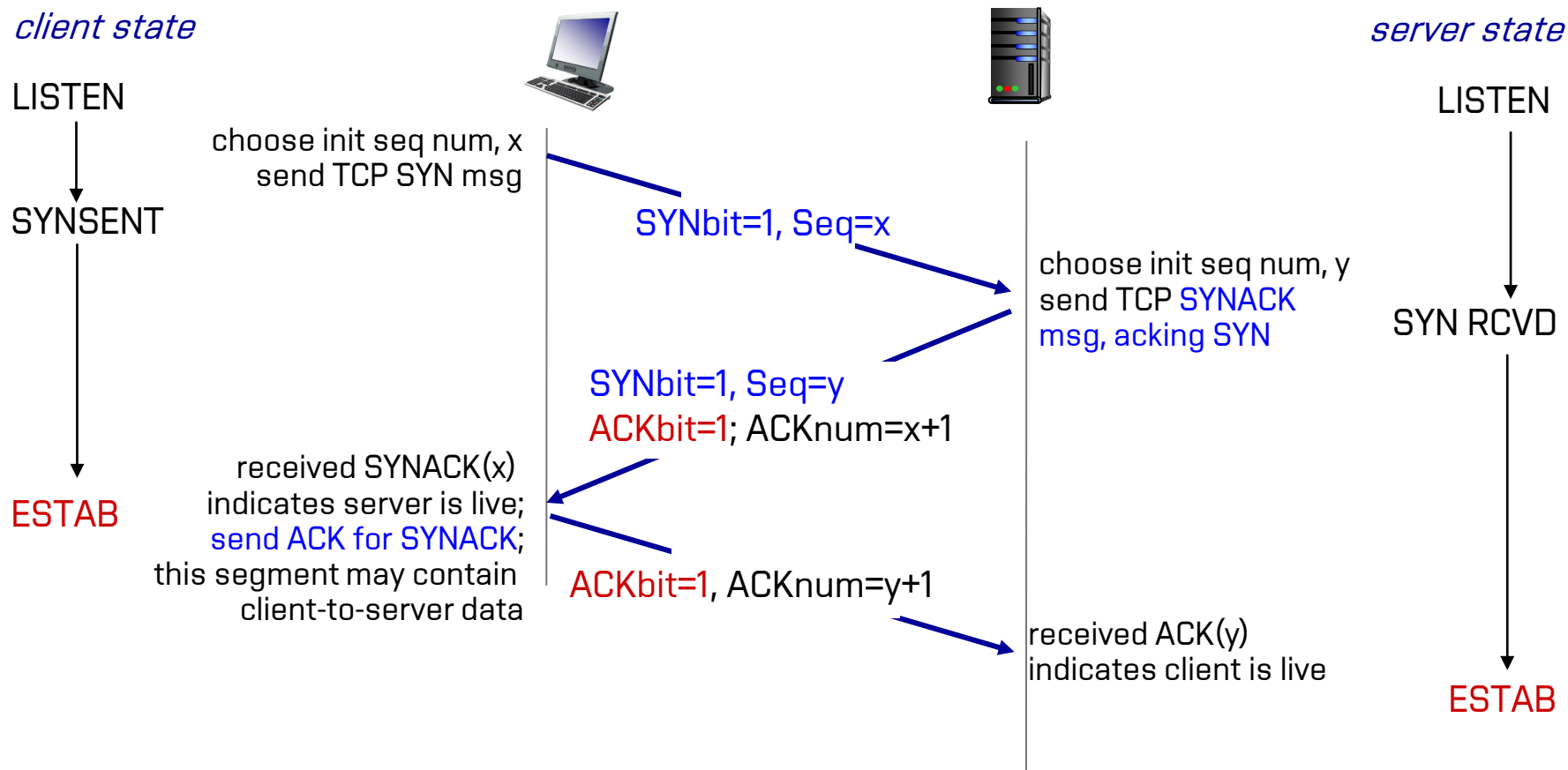
Napůl otevřené spojení!

b) zatoulaná data



Vložení paketu!

3-Way Handshake (příznak SYN)



Ukončení spojení (příznak FIN)

client state

ESTAB

`clientSocket.close()`

FIN_WAIT_1

can no longer
send but can
receive data

FIN_WAIT_2

wait for server
close

TIMED_WAIT

timed wait
for $2 \cdot \text{max}$
segment lifetime

CLOSED



FINbit=1, seq=x

ACKbit=1; ACKnum=x+1

FINbit=1, seq=y

ACKbit=1; ACKnum=y+1

can still
send data

can no longer
send data

server state

ESTAB

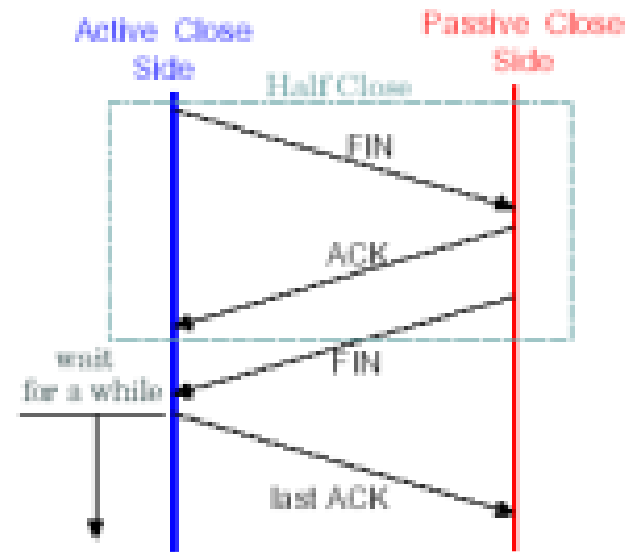
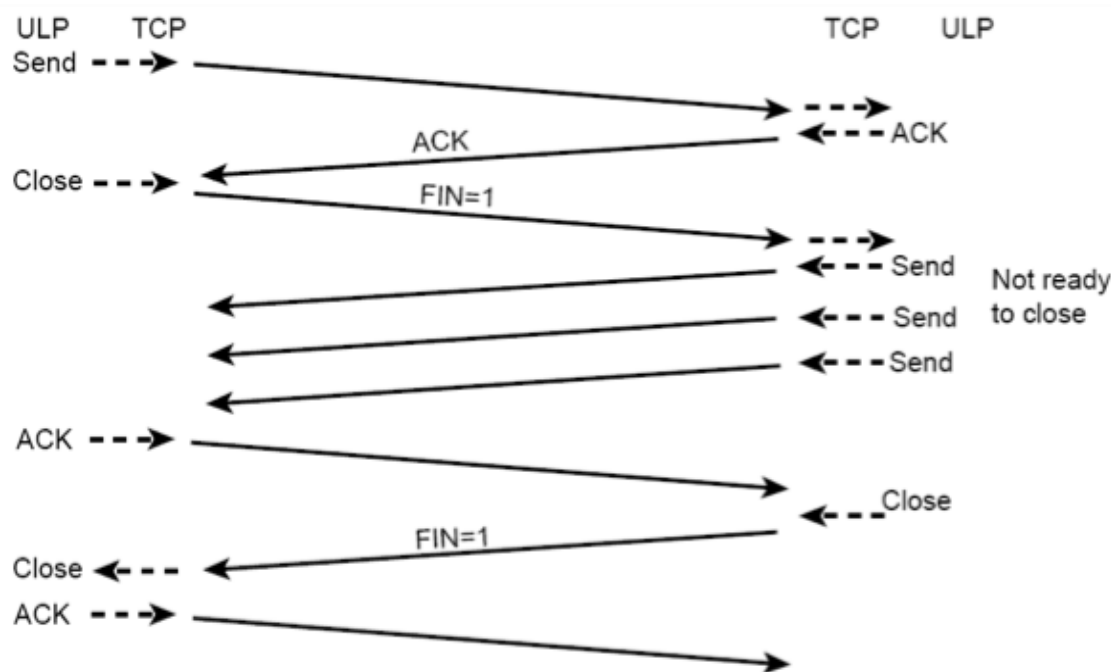
CLOSE_WAIT

LAST_ACK

CLOSED

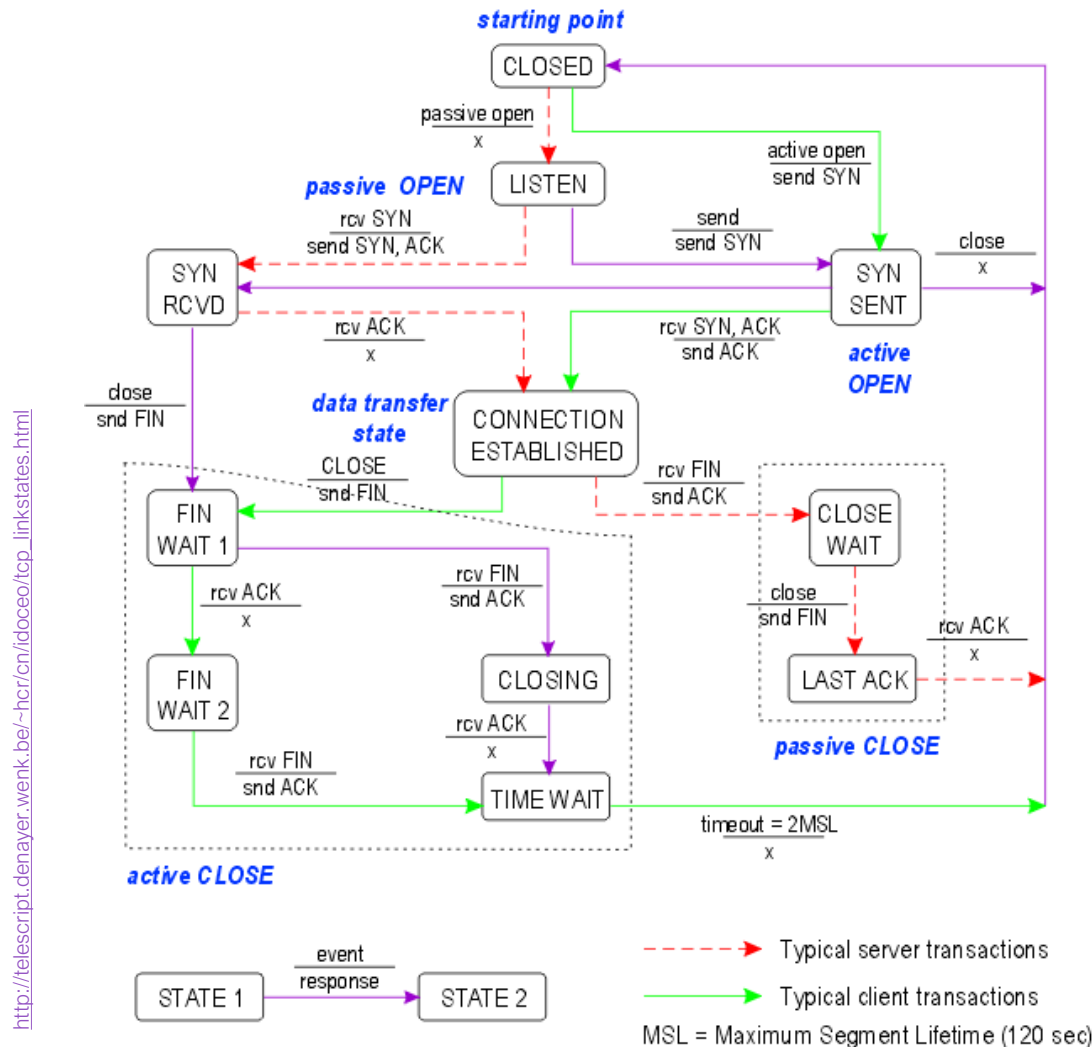
Druhy ukončení spojení

- FIN příznak nastaven -> následné potvrzení
- Aktivní a Pasivní uzavření spojení
- Poloviční uzavření spojení (half close)
 - data posílána pouze jedním směrem

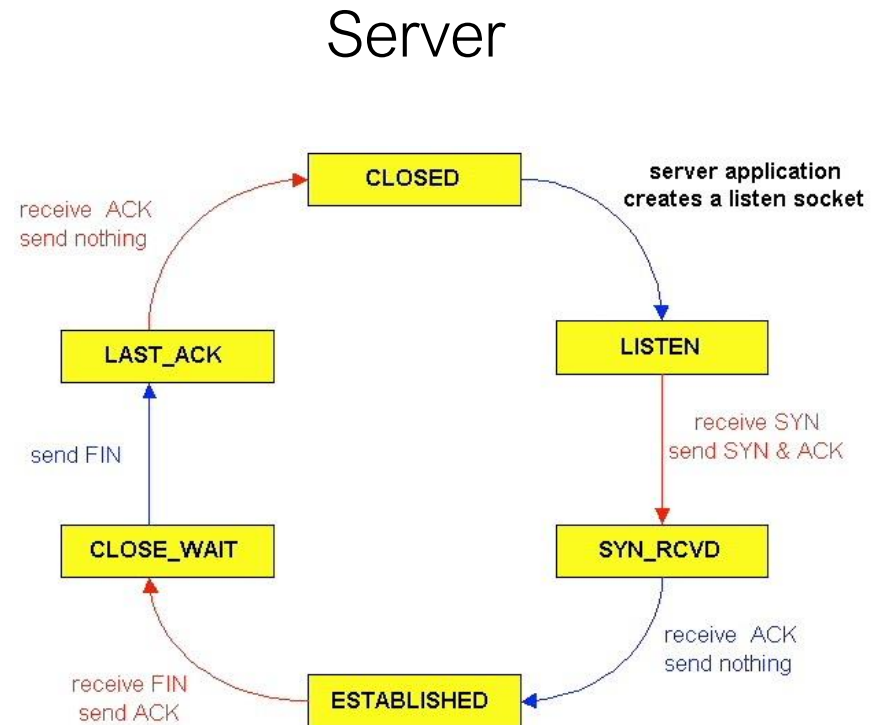
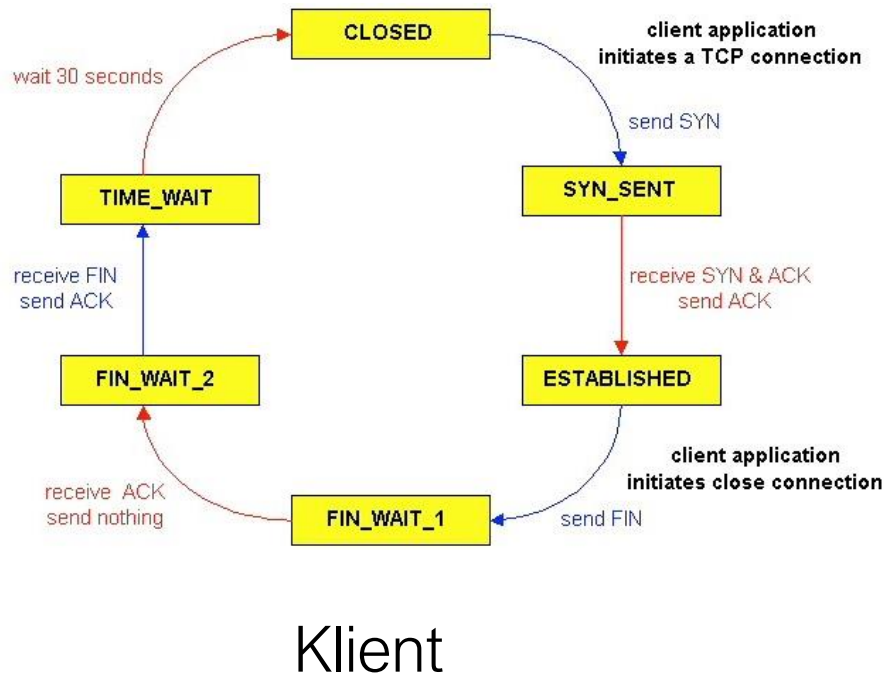


Zjednodušený FSM od TCP

TCP finite state machine



Überzjednodušený FSM od TCP



Obsah

1) PROTOKOL UDP

- Funkce, formát rámce

2) SPOLEHLIVÝ PŘENOS

3) PROTOKOL TCP

- Vlastnosti a funkce, formát segmentu
- Řízení spojení
- Řízení toku
- Řízení zahlcení

Spolehlivý přenos dat TCP

- TCP garantuje spolehlivé doručování nad „nespolehlivým“ (a.k.a. „best-effort“) IP
 - zřetězený přenos segmentů
 - **kumulativní ACK** potvrzování
 - jen jeden časovač (retransmit timer) znovuzasílání
- K znovuzasílání (**retransmission**) dojde:
 - když vyprší časovač
 - když se obdrží duplicitní ACK

```
NextSeqNum = InitialSeqNum
```

```
SendBase = InitialSeqNum
```

```
loop (forever) {
```

```
  switch(event):
```

```
    event: data received from application above  
           create TCP segment with  
           sequence numberNextSeqNum
```

```
    if (timer currently not running)
```

```
      start timer
```

```
      pass segment to IP
```

```
      NextSeqNum = NextSeqNum+length(data)
```

```
    event: timer timeout
```

```
      retransmit not-yet-acknowledged
```

```
      segment with smallest sequence number
```

```
      start timer
```

```
    event: ACK received, with ACK field value of y
```

```
      if (y > SendBase) {
```

```
        SendBase = y
```

```
        if (there are currently  
            not-yet-acknowledged segments)
```

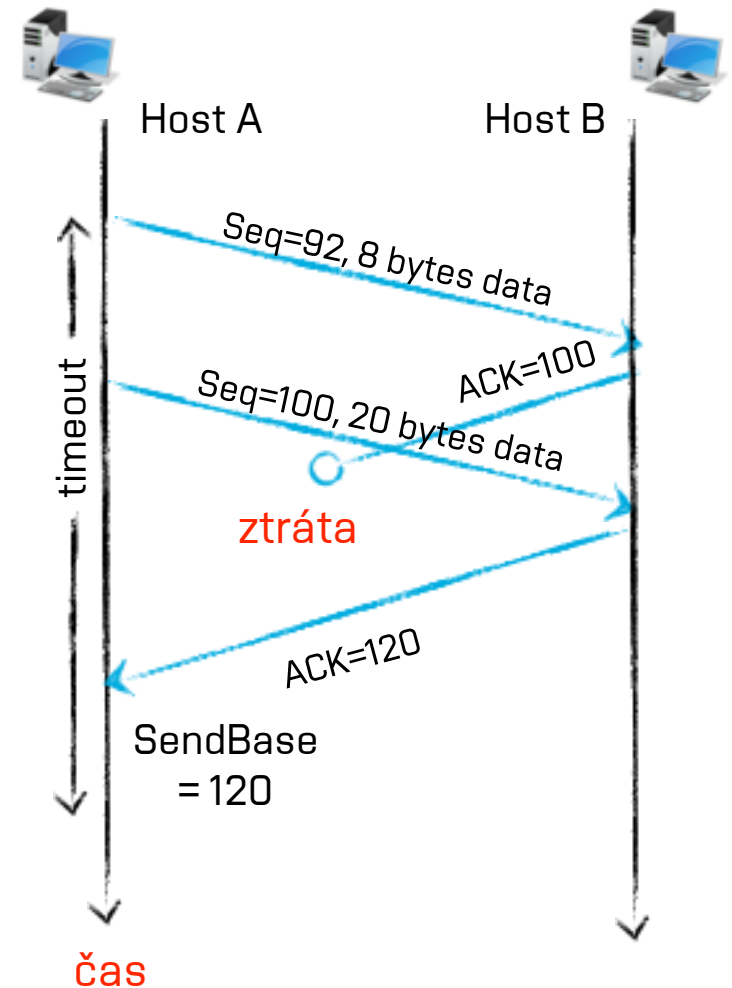
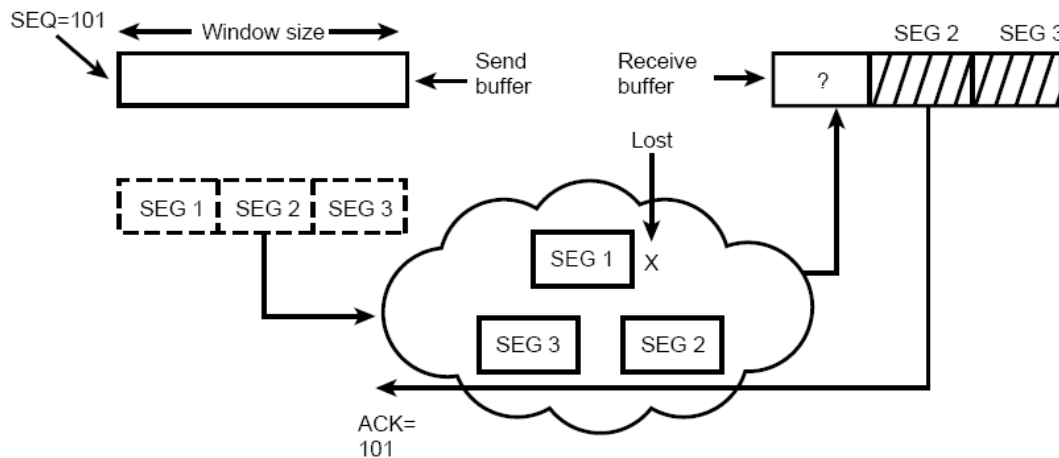
```
          start timer
```

```
      }
```

```
    } /* end of loop forever */
```

Kumulativní potvrzování

- Výhodou je, že ztracené ACK nemusíme znovu zasílat
- Co s out-of-order daty?
 - TCP neřeší, závisí na konkrétní implementaci

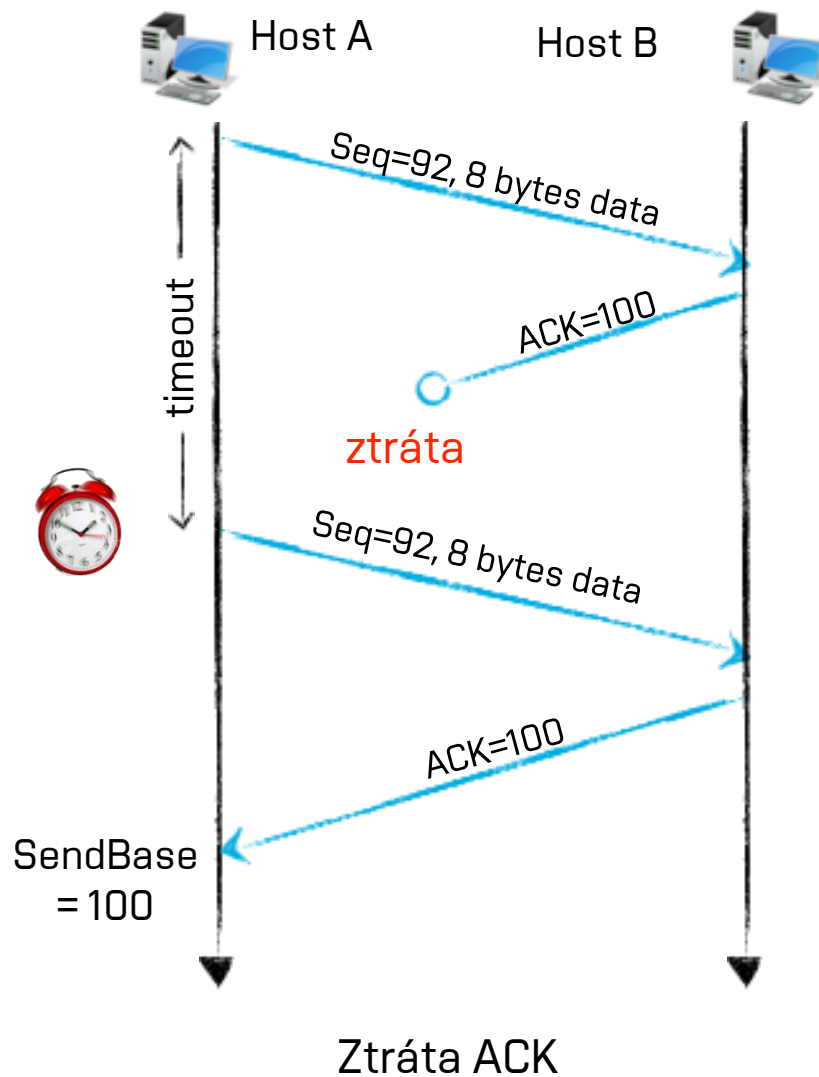


Kumulativní ACK

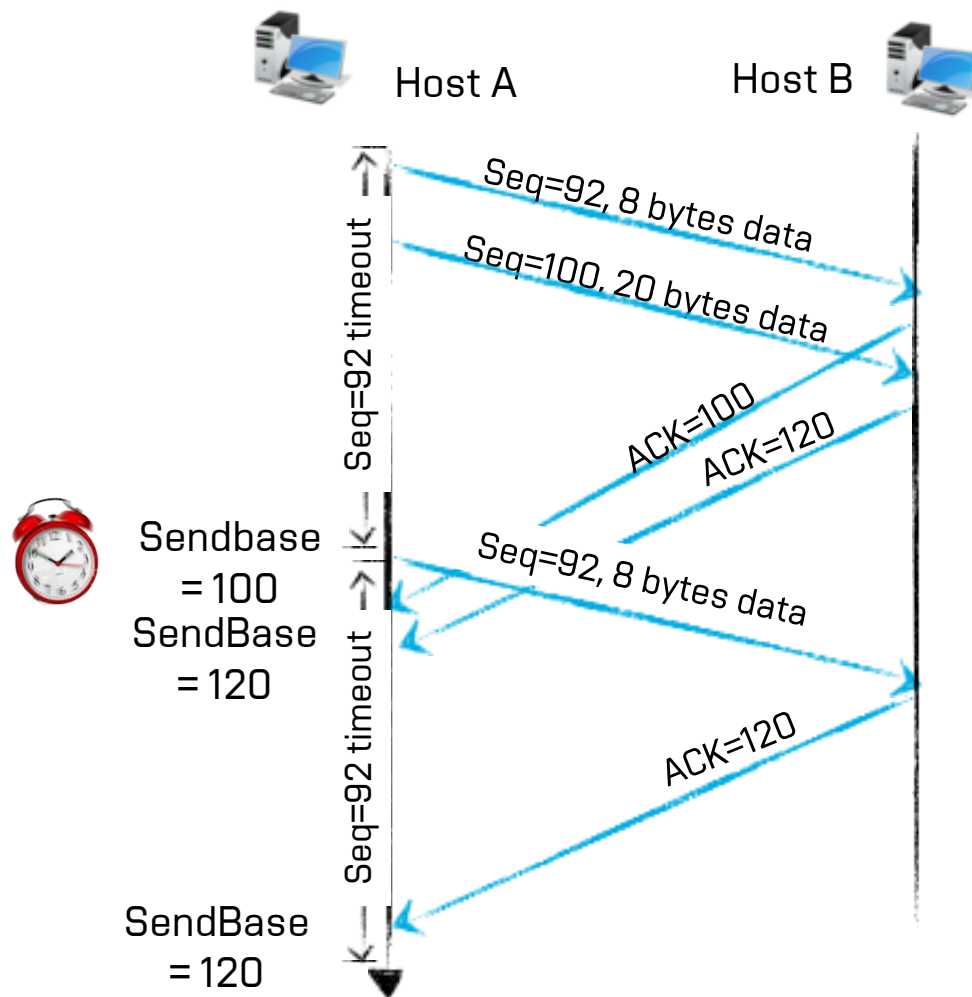
Znovuzasílání

- *Musí pořešit všechny možné i nemožné situace, ke kterým v průběhu přenosu může dojít*
- Možné případy:
 - Ztráta dat
 - Zpoždění dat
 - Ztráta ACK
 - Zpoždění ACK
- **Ztráta** vs. **Zpoždění** (*co je kdy?*)
 - odesílatel čeká na potvrzení, nemůže ovšem čekat nekonečně dlouho

TCP Retransmission (ztráta ACK)

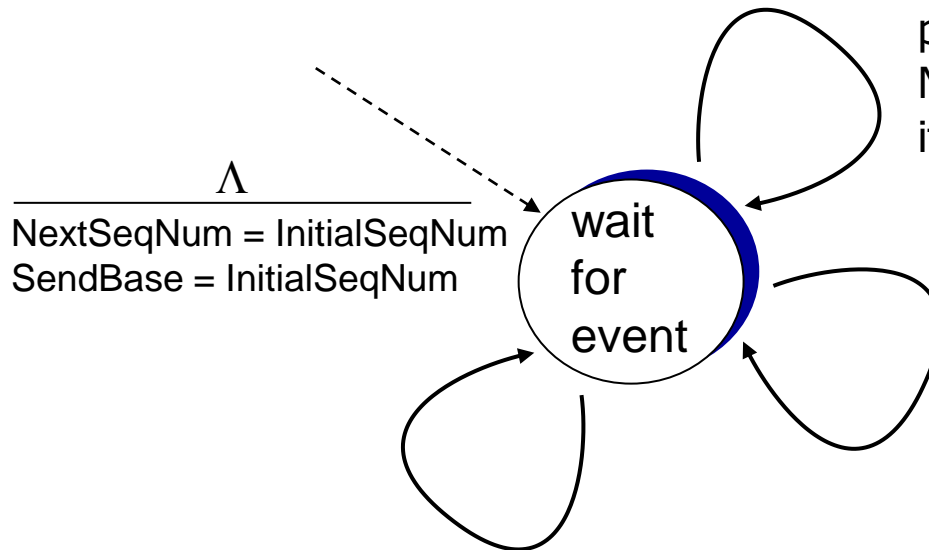


TCP Retransmission (zpožděné ACK)



Předčasné vypršení časovače

FSM odesilatele



Λ
NextSeqNum = InitialSeqNum
SendBase = InitialSeqNum

data received from application above

create segment, seq. #: NextSeqNum
pass segment to IP (i.e., “send”)
NextSeqNum = NextSeqNum + length(data)
if (timer currently not running)
start timer

timeout

retransmit not-yet-acked segment
with smallest seq. #
start timer

ACK received, with ACK field value y

```
if (y > SendBase) {  
    SendBase = y  
    /* SendBase-1: last cumulatively ACKed byte */  
    if (there are currently not-yet-acked segments)  
        start timer  
    else stop timer  
}
```

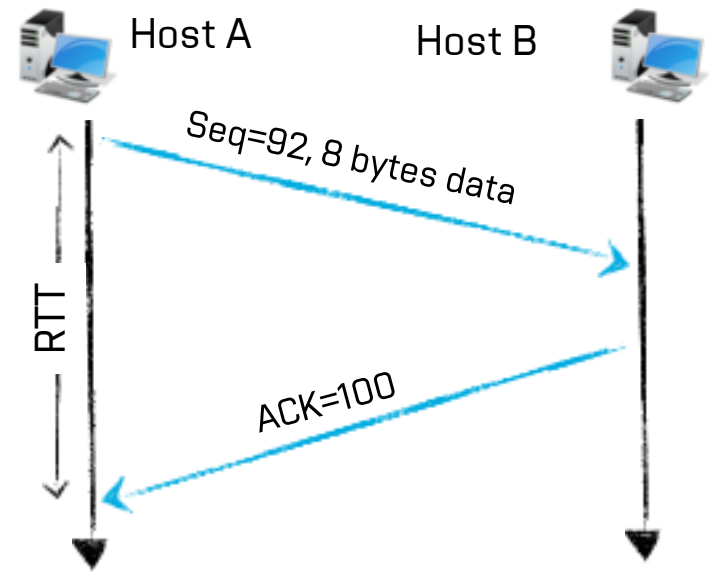

FSM příjemce (jak s ACK)

- RFC 1122, RFC 2581

Událost	Reakce
Přijetí dat v předpokladaném pořadí s očekávaným sekvenčním číslem. Předchozí data do tohoto sek. čísla již potvrzena.	Odlož odeslání potvrzení. Čekej až 500 ms na další segment. Jestliže nedorazí, pošli ACK.
Přijetí dat v předpokladaném pořadí s očekávaným sekvenčním číslem. Pro předchozí segment ještě nebylo odesláno potvrzení.	Okamžitě odešli kumulativní ACK, který potvrdí oba segmenty přijaté v očekávaném pořadí.
Přijetí dat mimo pořadí s vyšším sekvenčním číslem než je očekáváno. Vznik mezery v bufferu.	Okamžitě odešli duplikovaný ACK, který indikuje sekvenční číslo očekávaného oktetu segmentu.
Přijetí dat které zcela nebo částečně vyplní mezeru.	Okamžitě odešli ACK, který označuje další očekávaný oktet mezery.

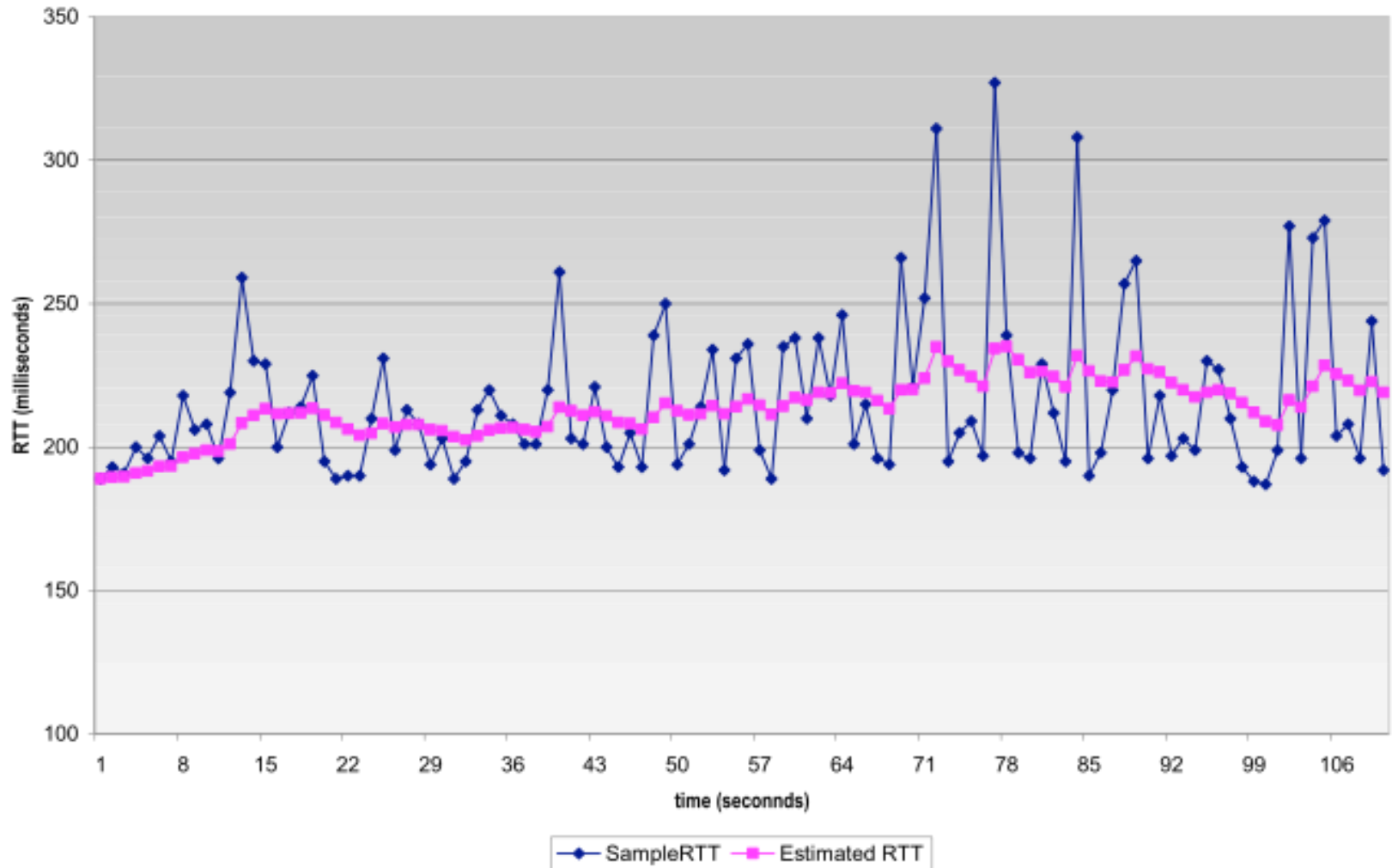
Vypršení timeoutu

- Q: *Jak nastavit timeout pro potvrzení TCP segmentu ?*
- A: *Na hodnotu vyšší než je RTT*
 - R: *Ale RTT se může měnit a mění !!!*
- **Hodnota < RTT = předčasný timeout**
 - Zbytečný opakovaný přenos segmentu
- **Hodnota > RTT = pomalá reakce na ztrátu segmentu nebo potvrzení**
- *Jak odhadnout RTT?*



Reálné RTT a odhad SRTT

RTT: gaia.cs.umass.edu to fantasia.eurecom.fr



Fast Retransmit

- RFC 2581
- *RTO* je často relativně velký
 - což vede ke zbytečně dlouhé čekání na znovuvyslání ztraceného paketu
- Ztracené segmenty lze poznat pomocí duplikovaných ACKs
 - Odesílatel posílá příjemci vlivem zřetězení více segmentů naráz
 - Jestliže je segment ztracen, bude pravděpodobně několik duplikovaných ACKs
- Příklad:
 - Jestliže příjemce obdrží 3 ACKs pro stejný segment, znamená to, že segment(y) za ACK potvrzenými daty se ztratil(y)
- Fast retransmit
 - = znovu pošle segment než vyprší *RTO* na druhé straně

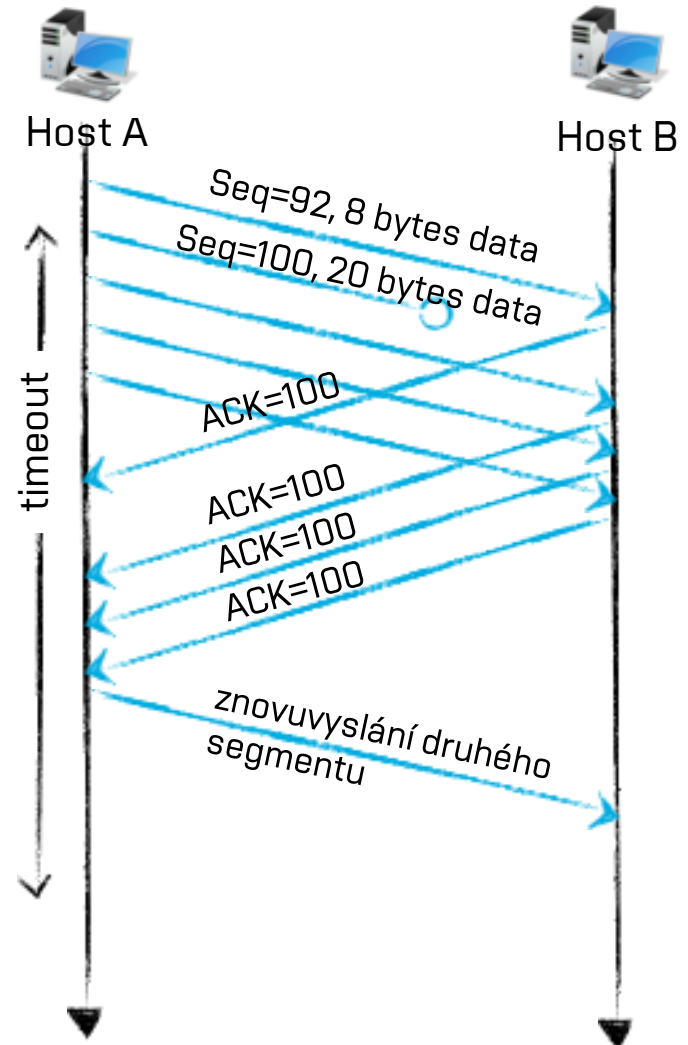
Fast Retransmit Příklad

event: ACK received, with ACK field value of y

```
if ( $y > \text{SendBase}$ ) {  
     $\text{SendBase} = y$   
    if (there are currently not-yet-acknowledged segments)  
        start timer  
}  
else {  
    increment count of dup ACKs received for  $y$   
    if (count of dup ACKs received for  $y = 3$ ) {  
        resend segment with sequence number  $y$   
    }  
}
```

Duplikovaný ACK

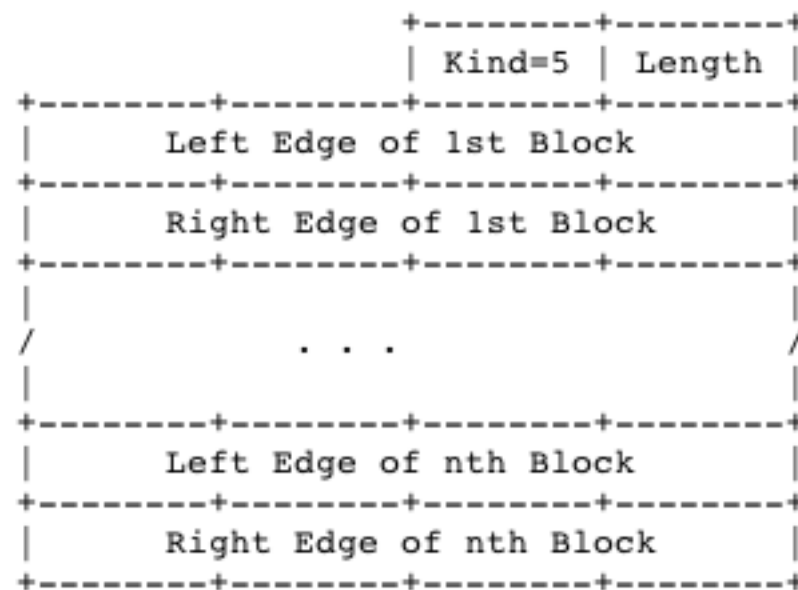
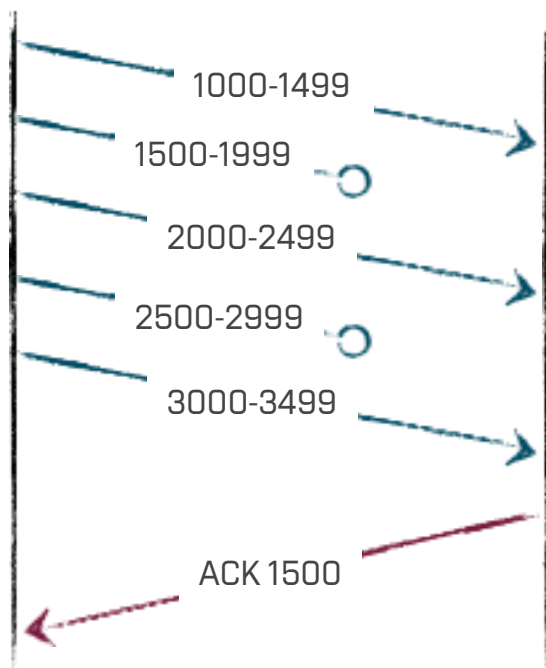
Fast retransmit



Selektivní potvrzování

- **Selective ACK** (RFC2018)

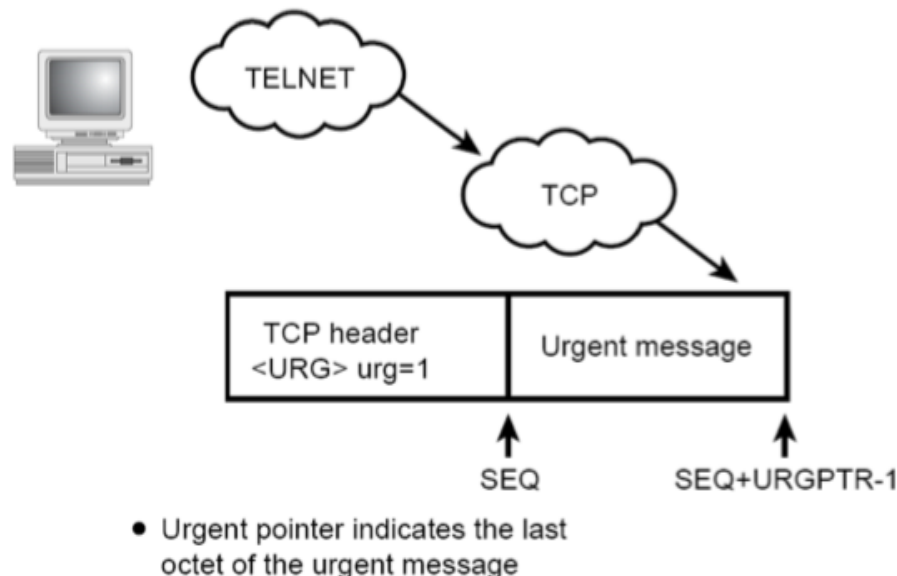
- Kumulativní ACK je nejednoznačný v případě ztráty více paketů
- TCP nemůže identifikovat přesně pakety, které nebyly doručeny
- Volba SACK při zahájení spojení



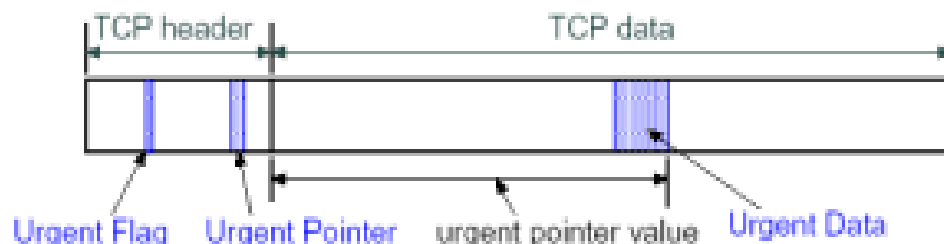
LeftEdge	RightEdge
2000	2500
3000	3500

Příznak URG

- TCP má jednoduchý (primitivní) mechanismus pro specifikaci prioritních dat
 - tzv. „urgentní“ data
- *K čemu slouží?*
 - Např: když aplikace potřebuje přerušit přenos dat a urychleně o tom informovat druhou stranu.
- Interaktivní aplikace využívají této vlastnosti: rlogin, telnet, ...

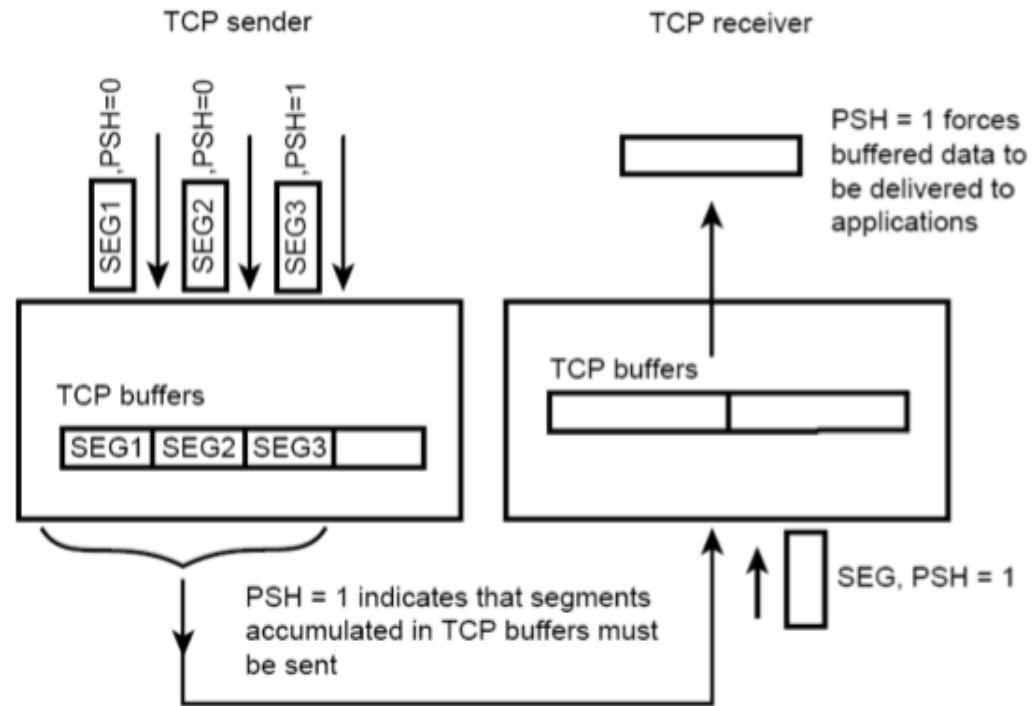


TCP packet with urgent data



Příznak PSH

- PSH příznak zabraňuje uváznutí komunikace (**communication deadlock**)
- Podpora je volitelná
- Minimálně však platí, že:
 - Nesmí uchovávat data v bufferu nekonečně dlouho (TCP se snaží maximálně využít MSS)
 - Musí nastavit PSH příznak v posledním bufferovaném segmentu (už není co více poslat)



Obsah

1) PROTOKOL UDP

- Funkce, formát rámce

2) SPOLEHLIVÝ PŘENOS

3) PROTOKOL TCP

- Vlastnosti a funkce, formát segmentu
- Řízení spojení
- Řízení toku
- Řízení zahlcení

Jak dojde k zahlcení sítě?

- **Zahlcení sítě (network congestion)**

- nastane když množství dat, která mají být přenesena je vyšší než je přenosová kapacita linky
- Směrovače disponují vyrovnávací pamětí
 - Pokud směrovač nemůže vyslat paket v danou dobu, uloží jej do vyrovnávací paměti (fronta) a čeká na další příležitost paket odeslat
 - Fronta má ale omezenou velikost → zvyšování prodlevy komunikace
 - Jestliže je fronta zaplněna, paket je zahozen → zvyšování ztrátovosti komunikace
- Zahlcení sítě má tendenci se zvyšovat/zhoršovat ☹
 - protokoly opakovaně znovuzasílají data, uživatelé se donekonečna pokouší o přenos → snižuje se počet korektně přenesených dat sítí, limitně až k nule → síť kolabuje → pakety po tisících umírají u bran bufferů a odebírají se do křemíkového nebe (tedy alespoň ty z paketů, jejichž majitelé v něj věří)

Ilustrace

- <http://kamery.praha.eu/ImageDetail.jsp?id=116804850&style=>



Přístupy k řízení zahltčení



End-end  *Tímto se budeme zabývat (Internet)*

- síťová vrstva neposkytuje žádné informace a podporu pro řízení zahltčení
- koncové stanice sledují charakteristiku sítě a aproximují aktuální stav
 - množství ztracených paketů
 - aktuální RTT
- například TCP

Network-assisted

- komponenty síťové vrstvy poskytují informace o stavu sítě
- například SNA, ATM ABR, TCP/IP ECN
- speciální „choke packet“ nebo označené datové pakety

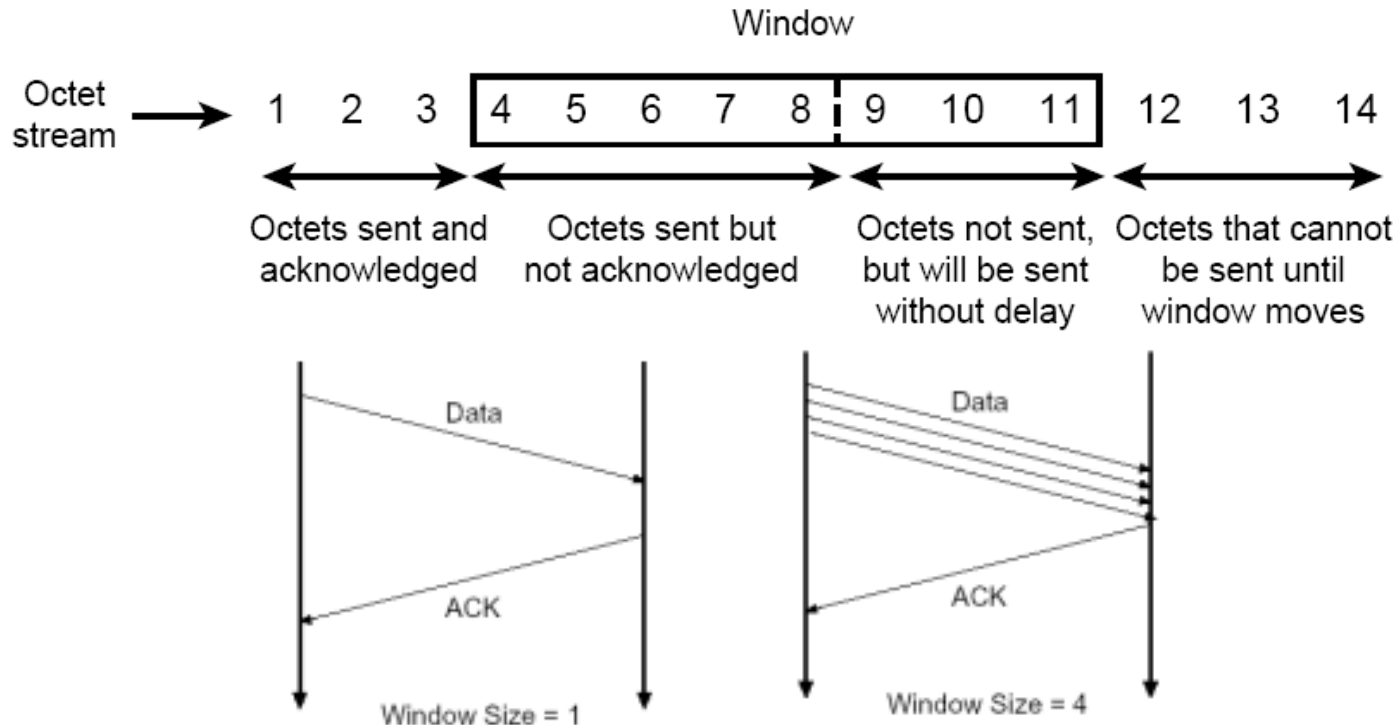
Základní princip

- Princip
 - *Protože nelze přesně zjistit stav sítě...*
 - *...tak TCP pohlíží na jakoukoliv ztrátu paketu jako na zahlcení*
- Znovuzaslání je řízeno jednoduchým způsobem:
 -  • Jestliže se pakety neztrácejí ...
 - TCP předpokládá že síť není zahlcena (odesílatel zvyšuje rychlost přenosu)
 -  • Jestliže se pakety ztrácejí ...
 - TCP předpokládá že síť je zahlcena (odesílatel snižuje rychlost přenosu)
- TCP zvyšuje rychlost přenosu až do detekce ztráty paketu
 - TCP se snaží odhadnout limit sítě ztrátou paketu

Klouzavé okno

- Velikost klouzavého okna (Sliding window)

- Změna velikosti okénka během spojení → přizpůsobení se aplikaci
- Nulová velikost indikuje zaplnění bufferu → odesílatel by měl přestat vysílat
- Cílem mechanismu je maximálně využít přenosové linky a minimalizovat zpoždění způsobená potvrzováním

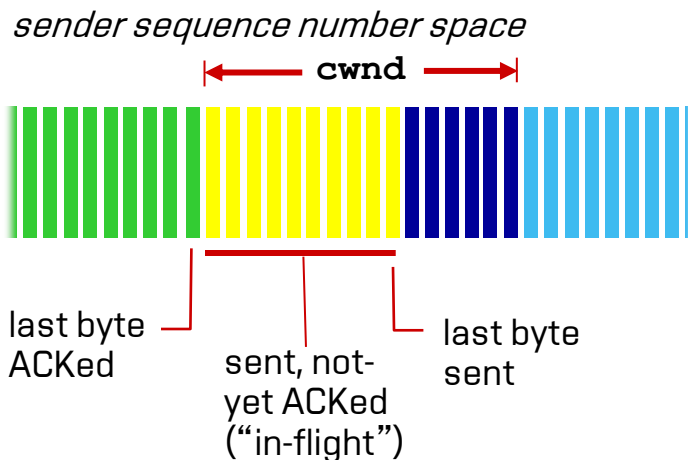


Detaily

Odesílatel limitován dle:

$$\text{LastByteSent} - \text{LastByteAcked} \leq \text{cwnd}$$

cwnd je dynamické, hodnota funkce předvídá zahltění sítě



TCP rychlost odesílání:

- 1) odešly *cwnd* B
- 2) počkej *RTO* na ACKs
- 3) pak pošli více B

$$\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

AIMD

- Additive Increase, Multiplicative Decrease

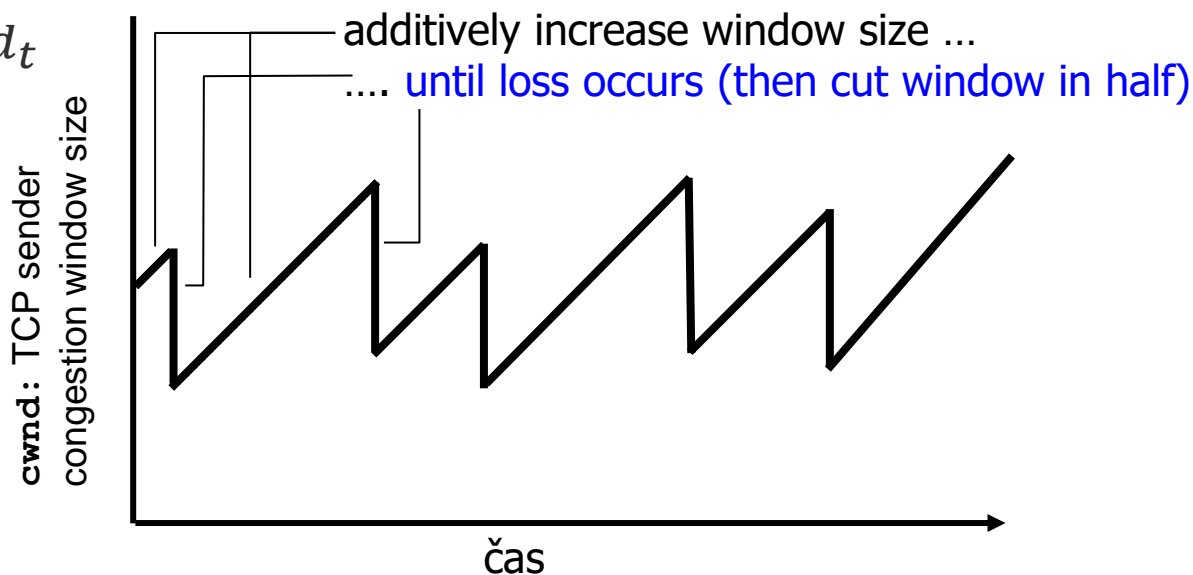
- [RFC 2581](#)

- Odesílatel zvedá rychlost odesílání lineárně

- $cwnd_{t+1} = cwnd_t + 1$

- V případě ztráty sníží rychlost na polovinu

- $cwnd_{t+1} = \frac{cwnd_t}{2}$

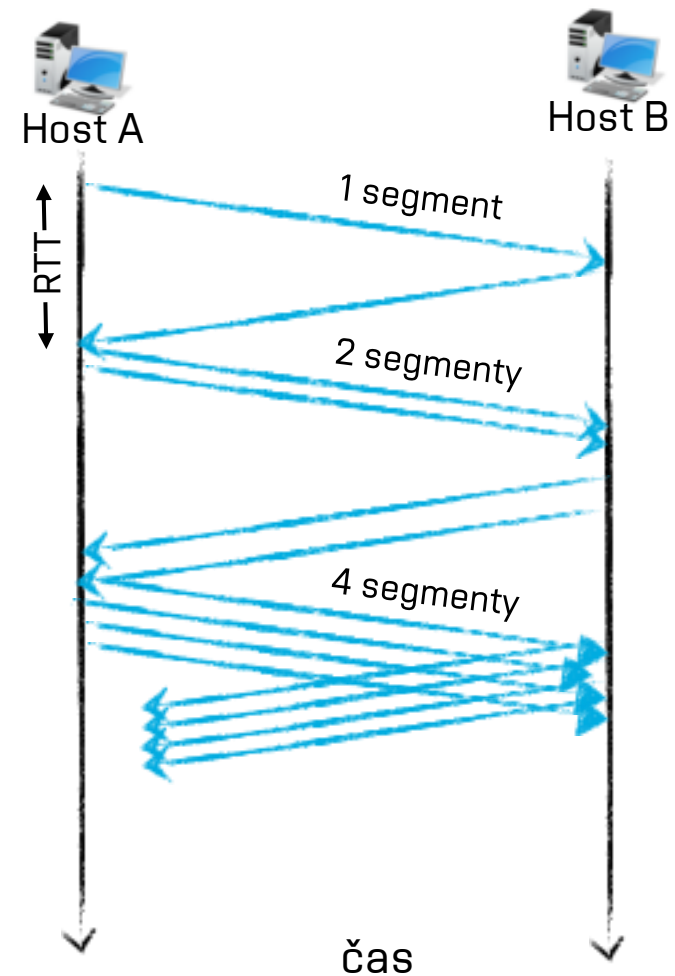


Dějepisné okénko

- V roce 1988 Van Jacobson publikoval článek [[Congestion Avoidance and Control](#)] na konferenci ACM SIGCOMM'88
 - Popisuje techniku jak reagovat na zahlcení sítě snížením toku dat od odesilatele. (ICMP Source Quench)
- Technika se nazývá [Slow Start with Congestion Avoidance](#)
 - Dostala se do RFC 1122 v roce 1989
- Obecně platné řízení zahlcení u TCP pracuje s:
 - Autonomní řízení koncovými systémy
 - Jednoduché algoritmy pro odhad stavu sítě
 - Zvolit vhodnou přenosovou rychlost: snaha vyhnout se co nejvíce zahlcení
 - Detekce zahlcení: předejít kolapsu jak to jen jde

TCP SS a CA

- *Po vytvoření spojení exponenciálně zrychluj...*
 - počáteční $cwnd_{t=0} = 1 \text{ MSS}$
 - $cwnd_{t+1} = 2cwnd_t$
- *...až do první ztráty, pak pokračuj lineárně*
 - $cwnd_{t+1} = cwnd_t + 1$
- Ideálně $BW = \frac{\text{MSS}}{\text{RTT}}$
- Snaha se rychle přiblížit této rychlosti

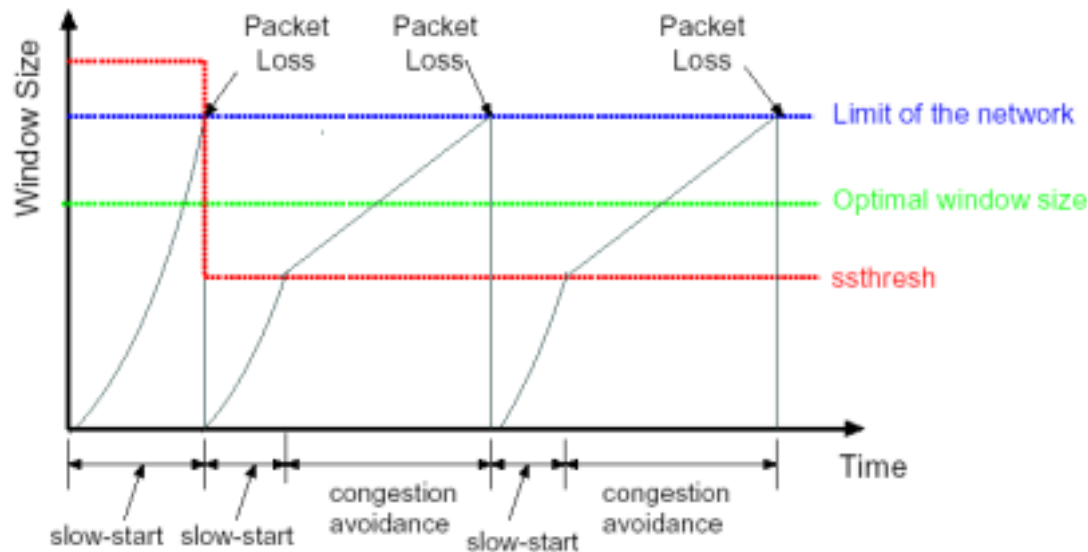


Historie řízení zahltčení v TCP

- Tři hlavní verze řízení zahltčení TCP – později už jen dílčí úpravy
 - Řízení zahltčení v TCP je spojeno s implementacemi distribuované v rámci BSD Unixu (pro VAX).
- **Tahoe**
 - Implementováno v 4.3BSD Tahoe, Net/1 (červen 1988)
 - Slow Start a Congestion Avoidance
 - Fast Retransmit
- **Reno**
 - Implementováno v 4.3BSD Reno, Net/2 (červen 1990), pre-release 4.4BSD
 - Fast Recovery následuje Fast Retransmit
- **NewReno**
 - Bez referenční implementace (1996)
 - Nový Fast Recovery algoritmus

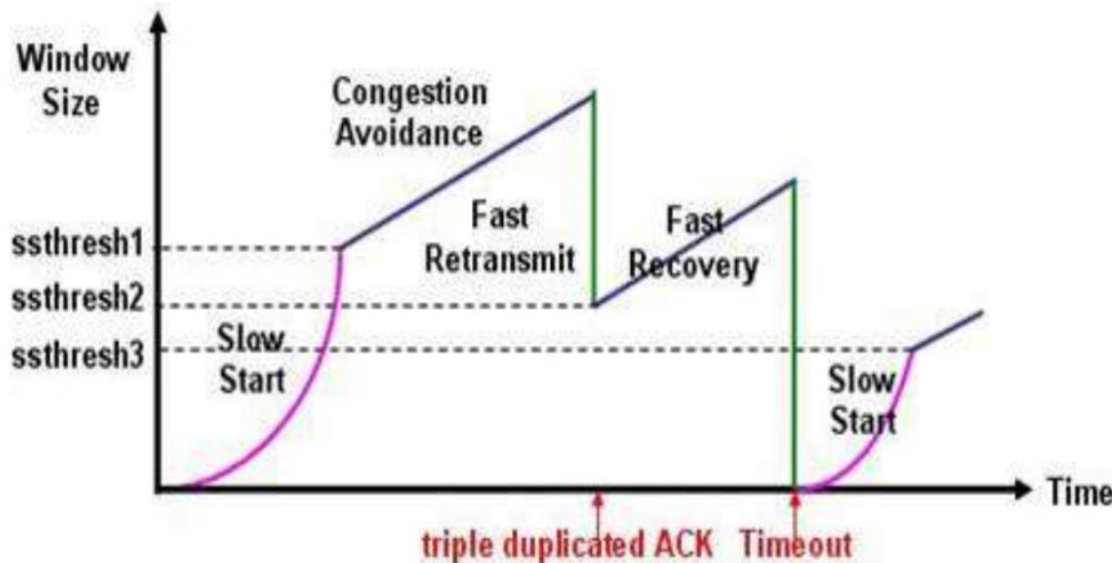
Tahoe (SS+CA)

- TCP si udržuje proměnnou *ssthresh*, aby věděl jaký algoritmus použít:
 - $cwnd < ssthresh$ potom slow start
 - $cwnd > ssthresh$ potom congestion avoidance
 - Detekce ztráty paketu $\rightarrow ssthresh = cwnd/2$

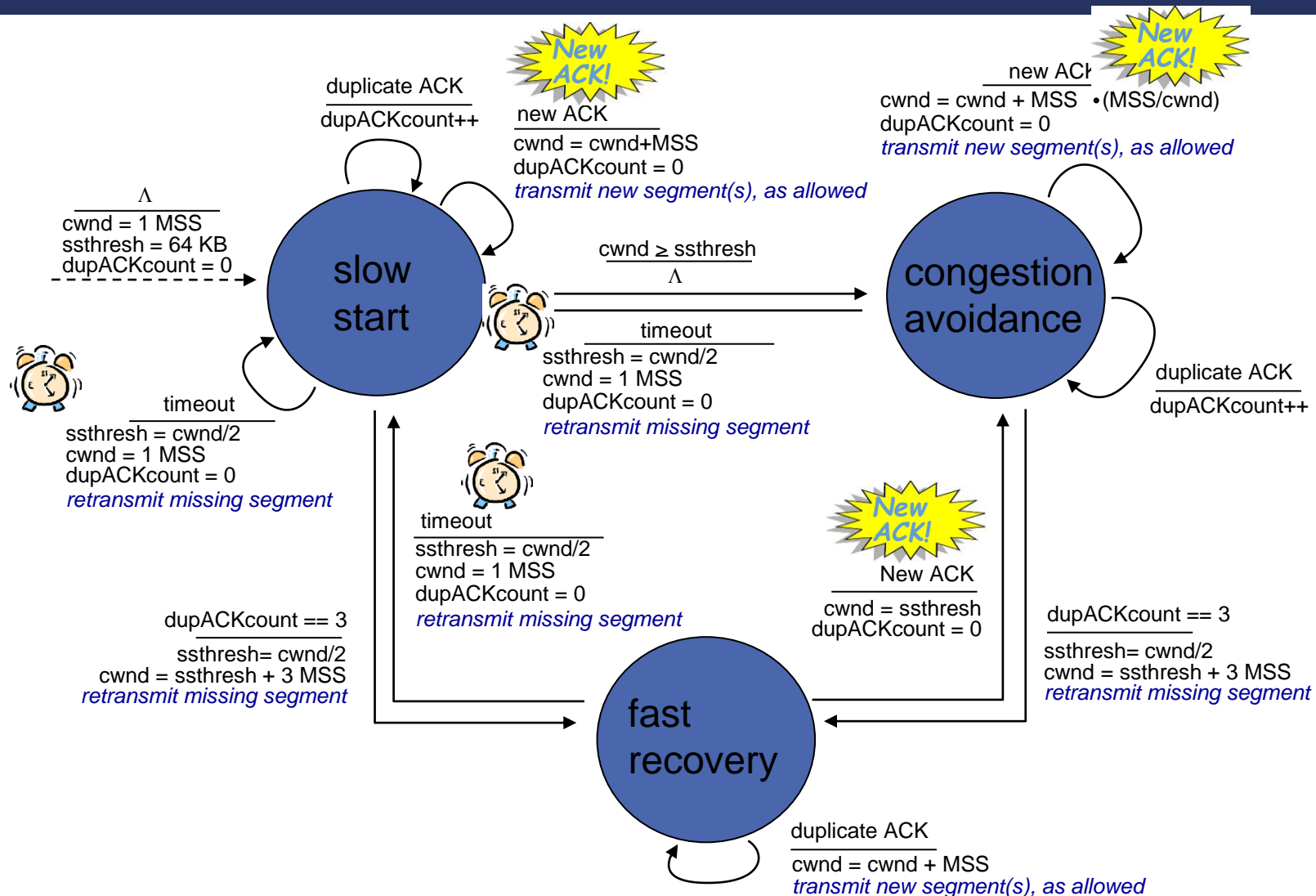


Reno (SS+CA+FR)

- Tahoe je citlivé na ztrátu paketu
 - 1% míra ztráty paketů může způsobit 50-75% snížení propustnosti
- **Fast Recovery**
 - Timeout – vážné zahlcení, *cwnd* zmenšeno na min a slow start
 - Duplikovaný ACK (konkrétně 3 duplikáty) – zahlcení není kritické, *cwnd* na polovinu a congestion avoidance

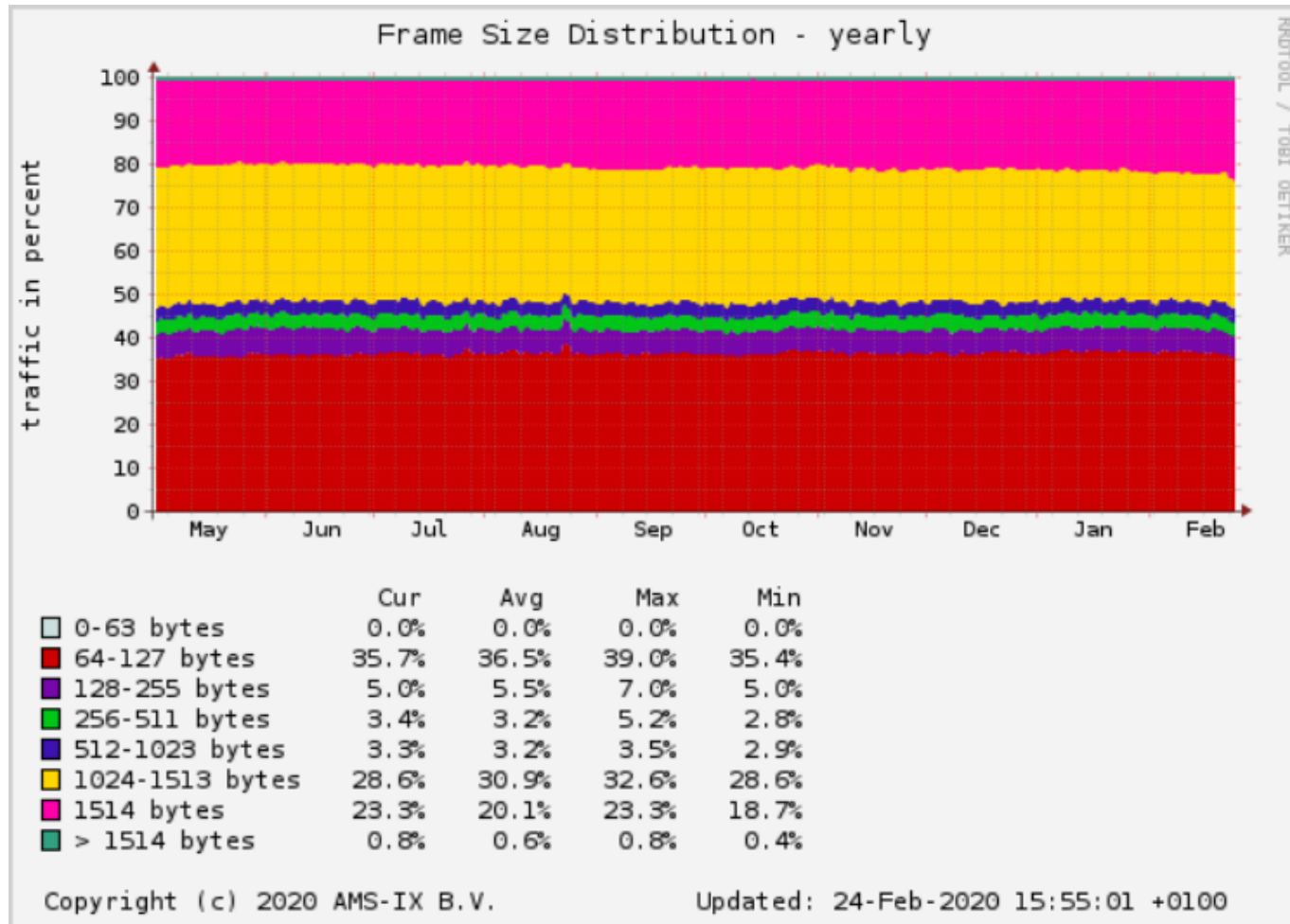


Reno FSM



Kolik TCP je na Internetu?

- <https://stats.ams-ix.net/sflow/index.html>



Studijní materiály

- Kurose J.F., Ross K.W.: [Computer Networking, A Top-Down Approach Featuring the Internet](#). Addison-Wesley, 2003. {kapitola 3.4}
- White, J.: [TCP/IP inside Datacenter](#),
<https://www.slideshare.net/datacenters/tcpip-inside-the-data-center-and-beyond>
- V. Jacobson, S.Braden, D.Borman: [TCP Extensions for High Performance](#). RFC 1323, May 1992.
- M. Mathis, J.Mahdavi, S.Floyd, A.Romanow: [TCP Selective Acknowledgment Options](#). RFC 2018, October 1996.
- Halsall, F.: [Computer Networking and the Internet](#). Addison-Wesley, 2005.
- W.R. Stevens: [TCP/IP Illustrated, Volume 1: The Protocols](#). Addison-Wesley, Reading, MA, 1994.
- RFC 768, 793, 1072, 1122, 1323, 2018, 2960, 2581, 2988, 3649, 3782