



VYSOKÉ UČENÍ FAKULTA  
TECHNICKÉ INFORMAČNÍCH  
V BRNĚ TECHNOLOGIÍ



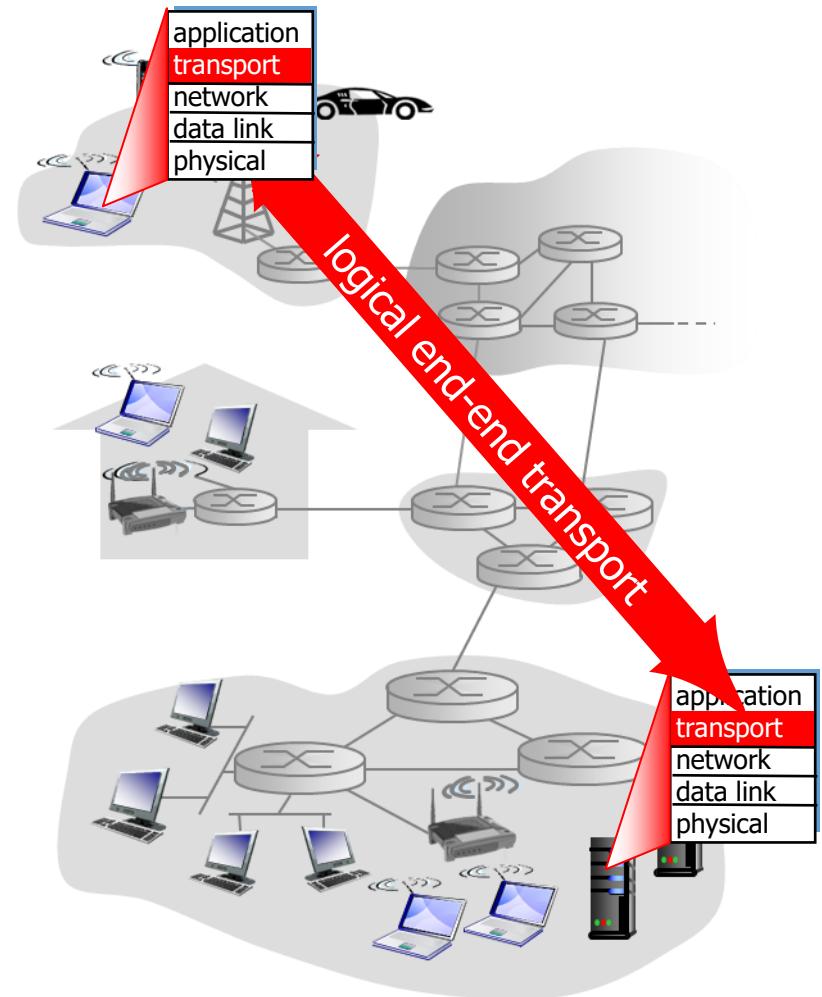
3

## Transportní vrstva

IPK 2021/2022 L

# Služby transportní vrstvy

- Logická komunikaci mezi aplikačními procesy
- L4 implementována především koncovými stanicemi
  - odesilatel „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)
    - = odesilatel/příjemce detekují ztráty a reagují na ně
  - řízení zahlcení (congestion control)
    - = odesilatel 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)



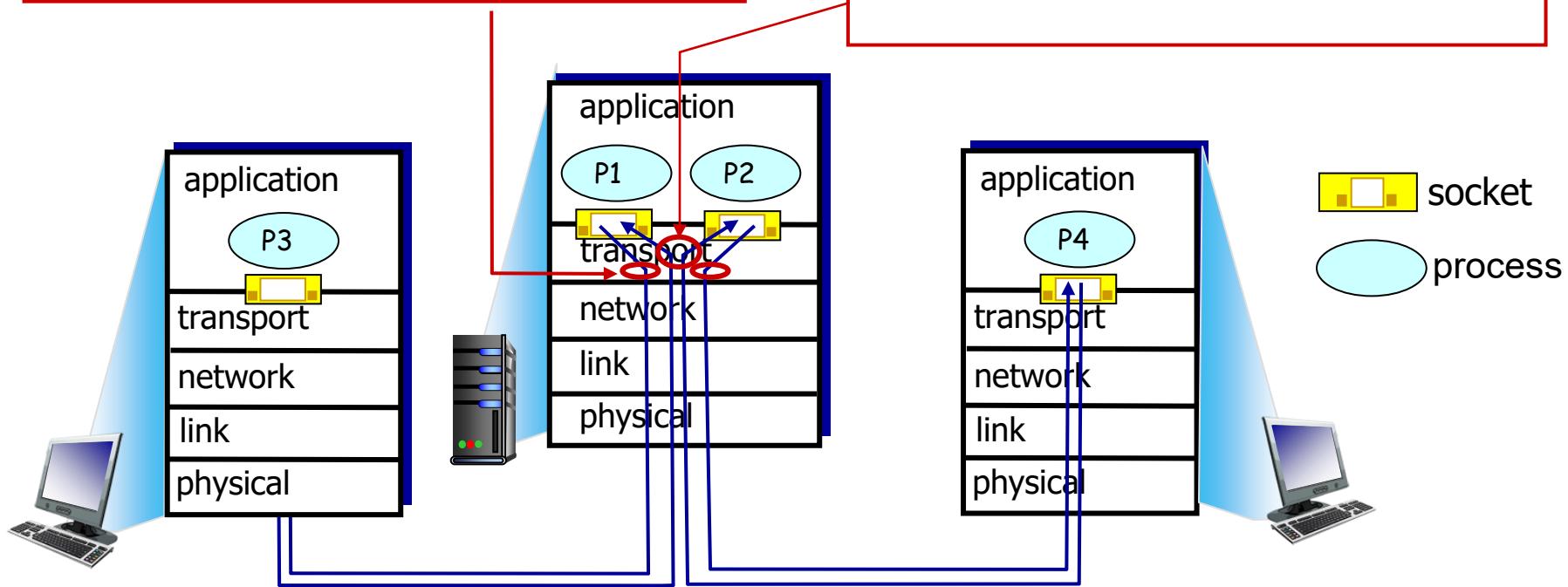
# Multiplexing

- *multiplexing odesilatel:*

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]

[Search] [txt|html|pdf|bibtex] [Tracker] [Email] [Diff1] [Diff2] [Nits]

RFC 768 Internet Standard  
J. Postel ISI  
28 August 1980

**User Datagram Protocol**

-----

Introduction

-----

This User Datagram Protocol (UDP) is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks. This protocol assumes that the Internet Protocol (IP) [1] is used as the underlying protocol.

This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP) [2].

Format

-----

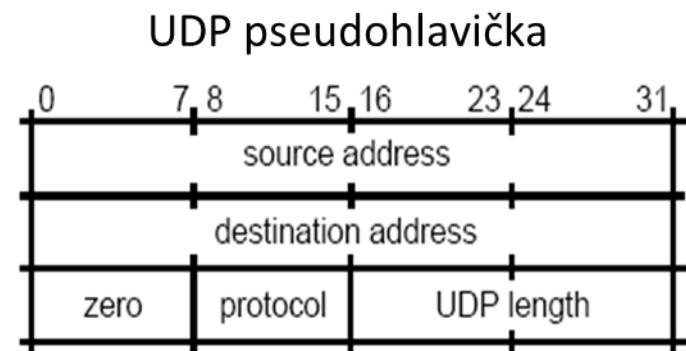
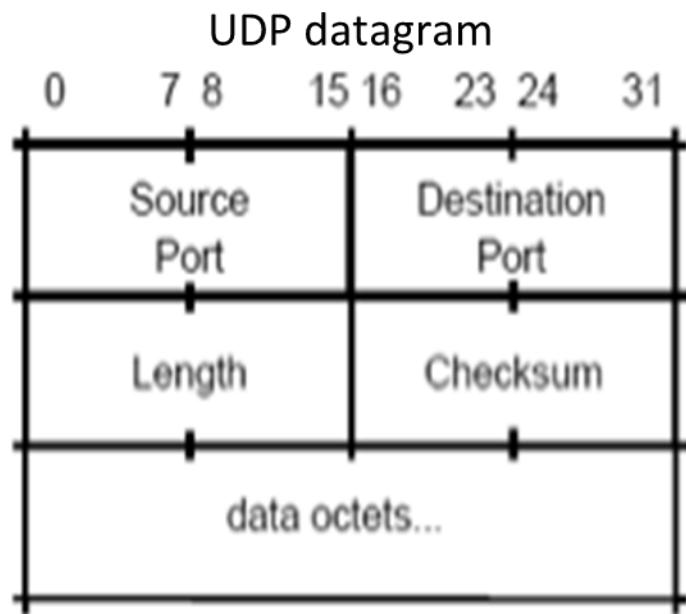
0	7	8	15	16	23	24	31
+-----+-----+	-----+-----+	-----+-----+	-----+-----+	-----+-----+	-----+-----+	-----+-----+	-----+-----+
	Source		Destination				
	Port		Port				
+-----+-----+	-----+-----+	-----+-----+	-----+-----+	-----+-----+	-----+-----+	-----+-----+	-----+-----+
	Length		Checksum				
+-----+-----+	-----+-----+	-----+-----+	-----+-----+	-----+-----+	-----+-----+	-----+-----+	-----+-----+
	data octets ...						
+-----+-----+-----+-----+-----+-----+-----+-----+	...						

# Protokol UDP

- 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 odesilatele (**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

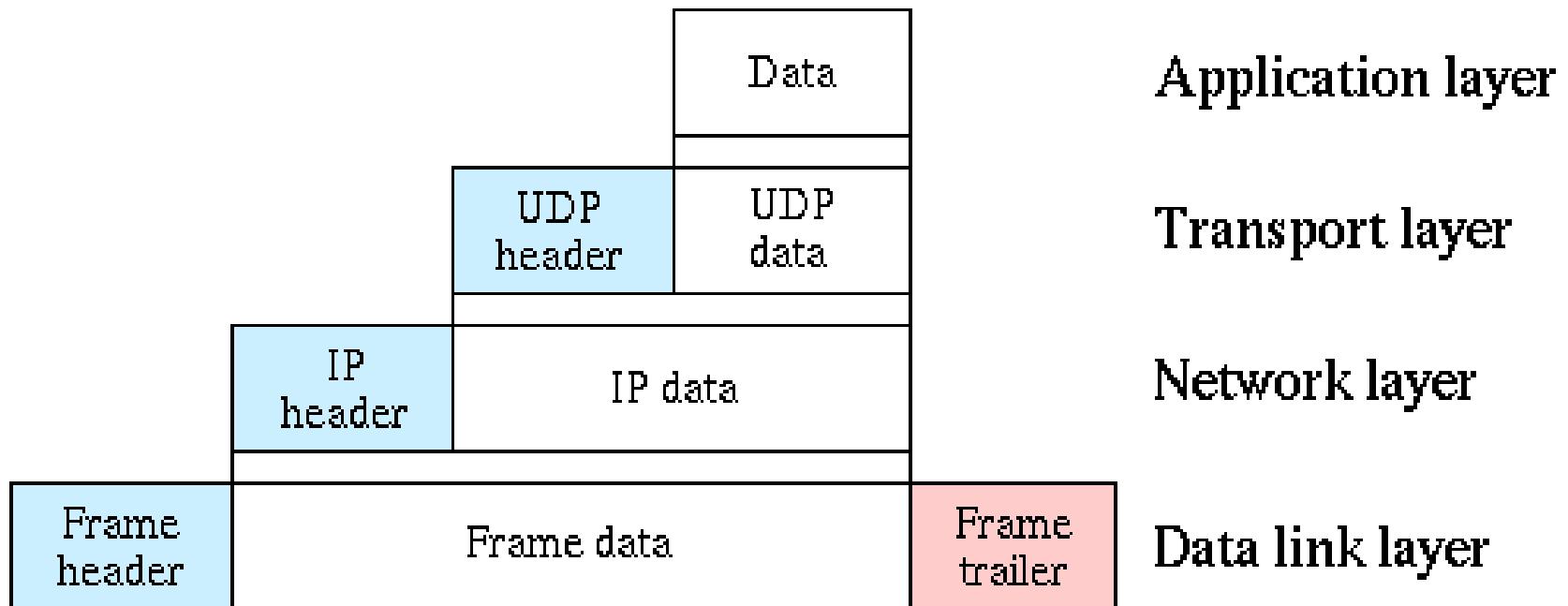


The screenshot shows a Windows file viewer window titled "Lister - [c:\WINDOWS\System32\drivers\etc\services]". The file contains a list of well-known services with their port numbers and protocols. The file starts with a copyright notice and a comment about IANA. It then defines the service format and lists various services like echo, discard, sysstat, daytime, qotd, chargen, ftp-data, etc. Many entries include comments describing the service's purpose.

```
## 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
sysstat   11/tcp   users          #Active users
sysstat   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   ttyst source  #Character generator
chargen  19/udp   ttyst source  #Character generator
ftp-data 20/tcp
ftp       21/tcp
ssh       22/tcp
telnet   23/tcp
smtp     25/tcp   mail           #Simple Mail Transfer
Protocol
time     37/tcp   timserver
time     37/udp   timserver
rlp      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   53/udp
bootps   67/udp   dhcps
bootpc   68/udp   dhcpc
tftp     69/udp
gopher   70/tcp
finger   79/tcp
http     80/tcp   www www-http  #World Wide Web
hosts2-ns 81/tcp
hosts2-ns 81/udp
kerberos 88/tcp   krb5 kerberos-sec #Kerberos
kerberos 88/udp   krb5 kerberos-sec #Kerberos
hostname 101/tcp  hostnames
iso-tsap 102/tcp
rtelnet   107/tcp
pop2     109/tcp  postoffice
Version 2
pop3     110/tcp
Version 3
```

# Zapouzdření UDP datagramu

- [https://upload.wikimedia.org/wikipedia/en/d/d8/UDP\\_encapsulation.png](https://upload.wikimedia.org/wikipedia/en/d/d8/UDP_encapsulation.png)



# UDP ve Wiresharku

No.	Time	Source	Destination	Protocol	Length	Info
718	9.812536000	2001:67c:1220:8t	2001:4860:4860::8888	DNS	89	Standard query 0xcf47 A seznam.cz
719	9.812901000	2001:67c:1220:8t	2001:4860:4860::8888	DNS	89	Standard query 0xd449 AAAA seznam.cz
723	9.833512000	2001:4860:4860::2001:67c:1220:8b5:749	DNS		105	Standard query response 0xcf47 A 77.7
724	9.834195000	2001:4860:4860::2001:67c:1220:8b5:749	DNS		117	Standard query response 0xd449 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)

    Version: 6

    Traffic class: 0x00000000

    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)

Hex	Dec	Text
0000	00 04 96 1d 4e 30 78 24	af 8a 95 6e 86 dd 60 00
0010	00 00 00 23 11 40 20 01	06 7c 12 20 08 b5 74 97
0020	89 fc 6b 44 66 19 20 01	48 60 48 60 00 00 00 00
0030	00 00 00 00 88 88 ce 0a	00 35 00 23 4a c2 4d 49
0040	01 00 00 01 00 00 00 00	00 00 06 73 65 7a 6e 61
0050	6d 02 63 7a 00 00 1c 00	01 .....

User Datagram Protocol (udp), 8 bytes

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

Profile: Default

# Obsah

## 1) PROTOKOL UDP

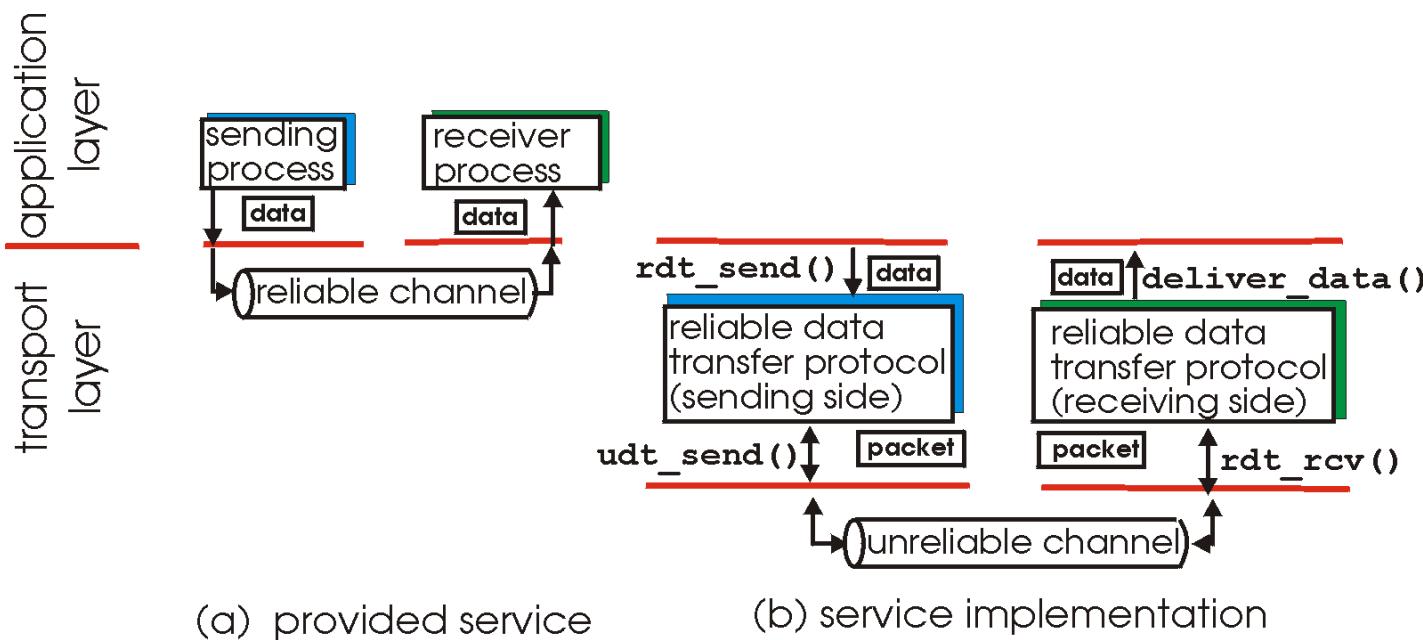
## 2) SPOLEHLIVÝ PŘENOS

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

## 3) PROTOKOL TCP

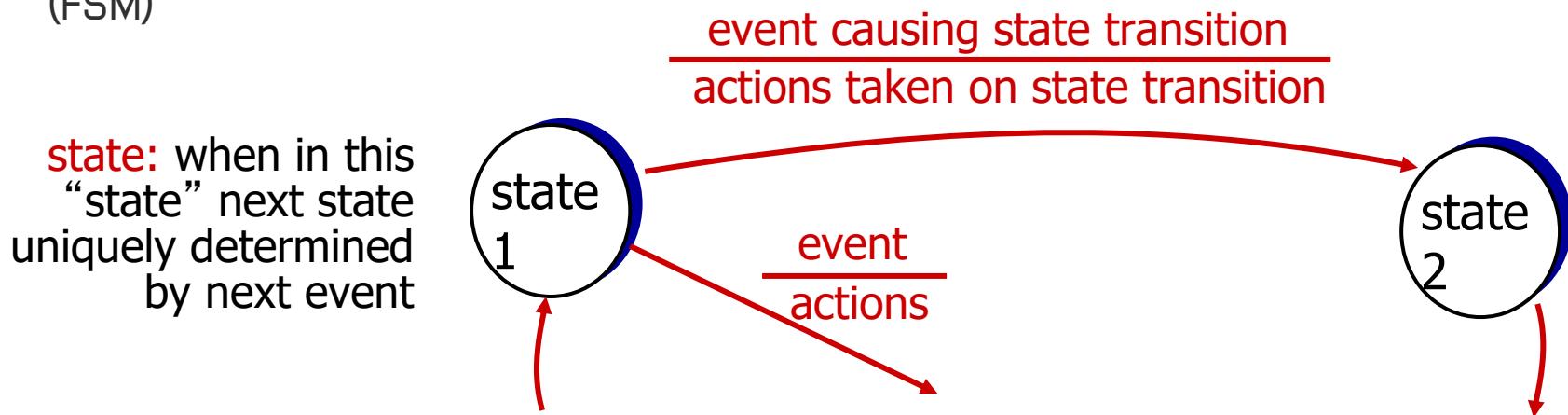
# Úvod

- 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

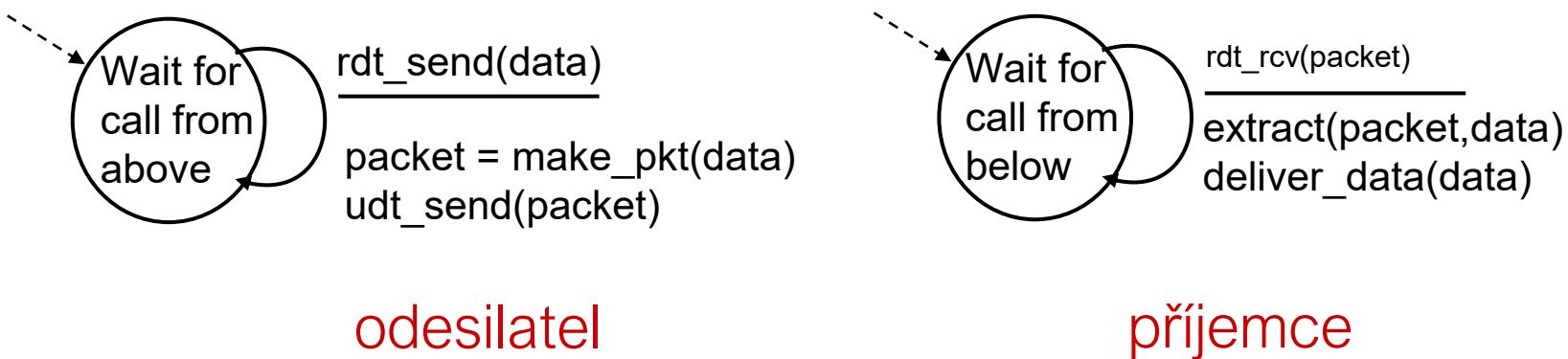


# Plán

- 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 odesilatele a příjemce bude popsáno konečným stavovým automatem (FSM)



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



# 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

rdt\_send(data)

sndpkt = make\_pkt(data, **checksum**)

udt\_send(sndpkt)

rdt\_rcv(rcvpkt) &&  
isNAK(rcvpkt)

Wait for  
call from  
above

Wait for  
ACK or  
NAK

rdt\_rcv(rcvpkt) && isACK(rcvpkt)

$\Lambda$

**odesilateł**

**příjemce**

rdt\_rcv(rcvpkt) &&  
**corrupt(rcvpkt)**

udt\_send(NAK)

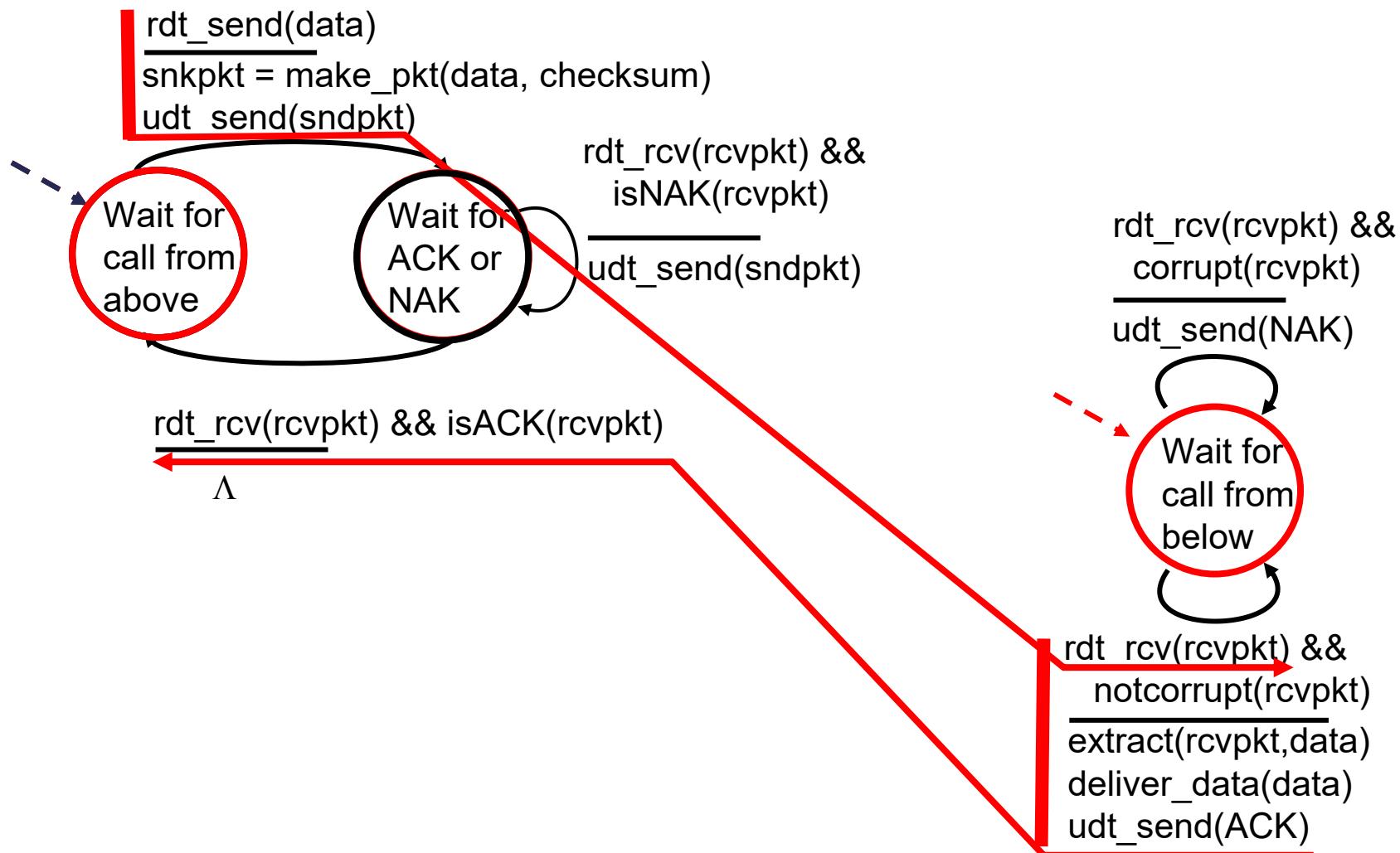
Wait for  
call from  
below

rdt\_rcv(rcvpkt) &&  
**notcorrupt(rcvpkt)**

extract(rcvpkt,data)  
deliver\_data(data)  
udt\_send(ACK)

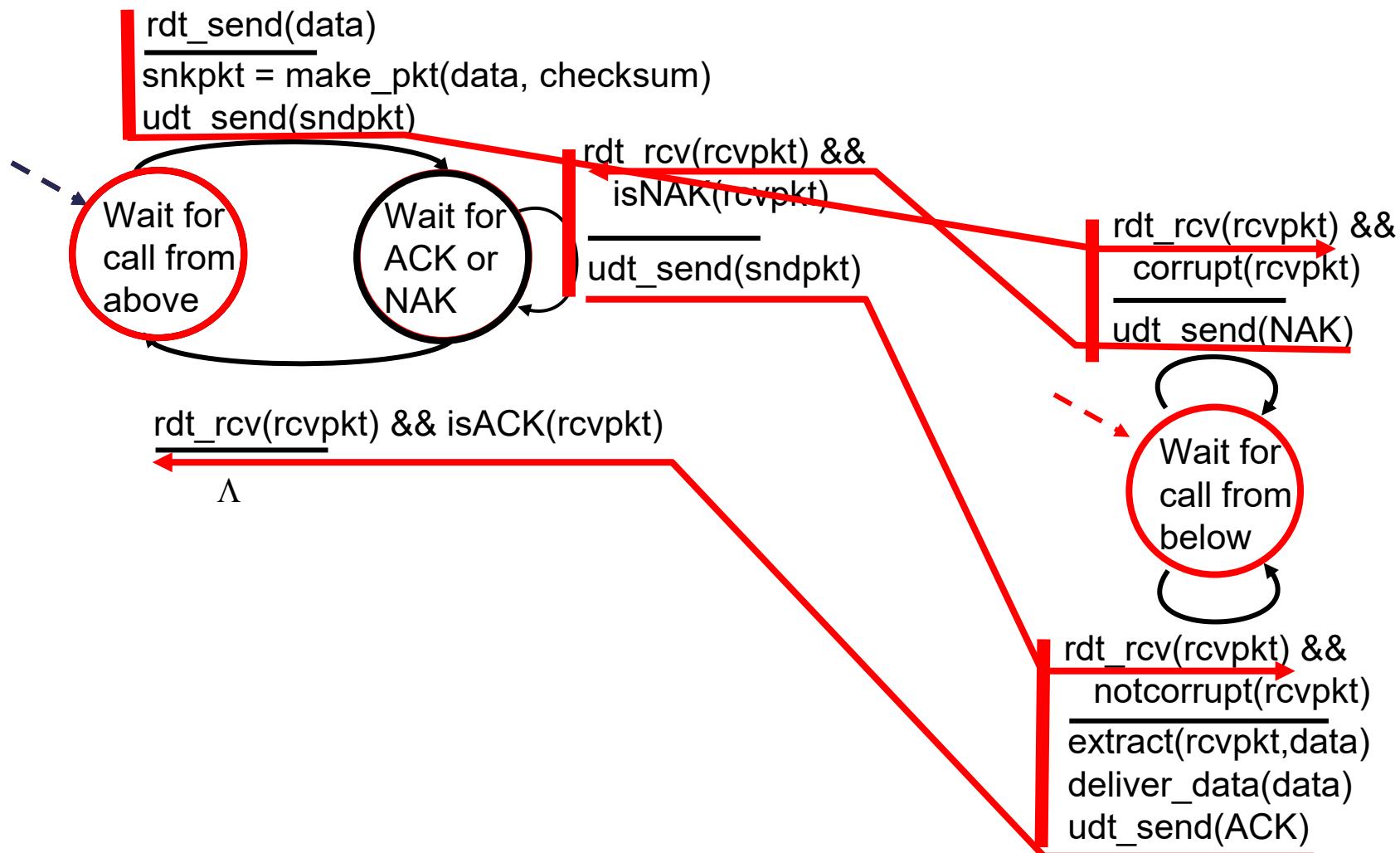
# RDT 2.0

## Normální chování



# RDT 2.0

## Příklad s chybou



## Nedostatek

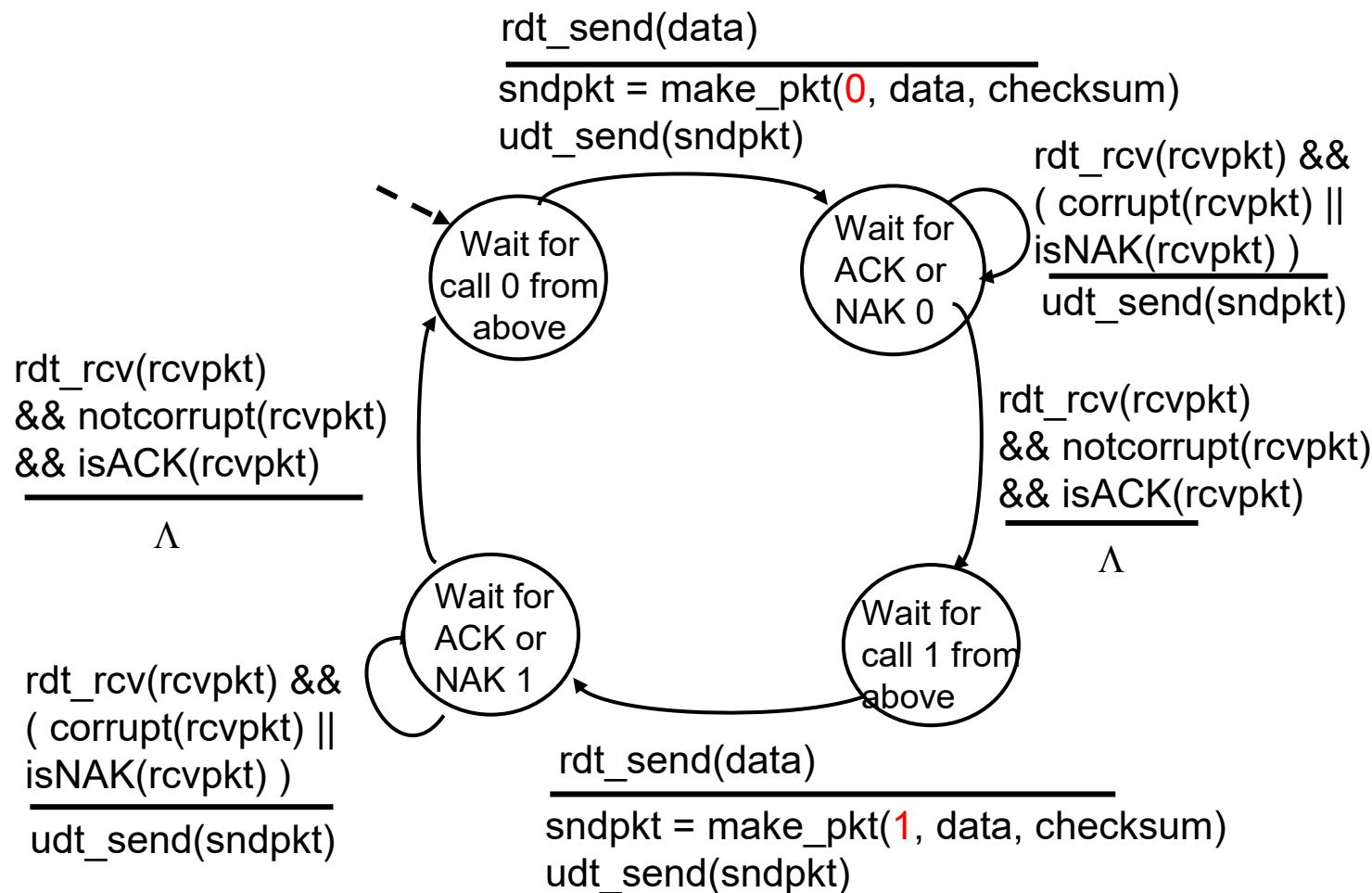
- *Co když je potvrzení (ACK/NAK) poškozeno?*
  - odesilatel neví, co se na straně příjemce stalo
  - není možné jednoduše poslat znovu paket
    - vznikla by duplikace paketů (dat)
- Možné řešení:
  - přidání pořadového čísla do paketu
  - odesilatel pošle znovu paket, jestliže je potvrzení poškozeno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

## Vlastnosti

- Odesilatel
  - 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 odesilatele

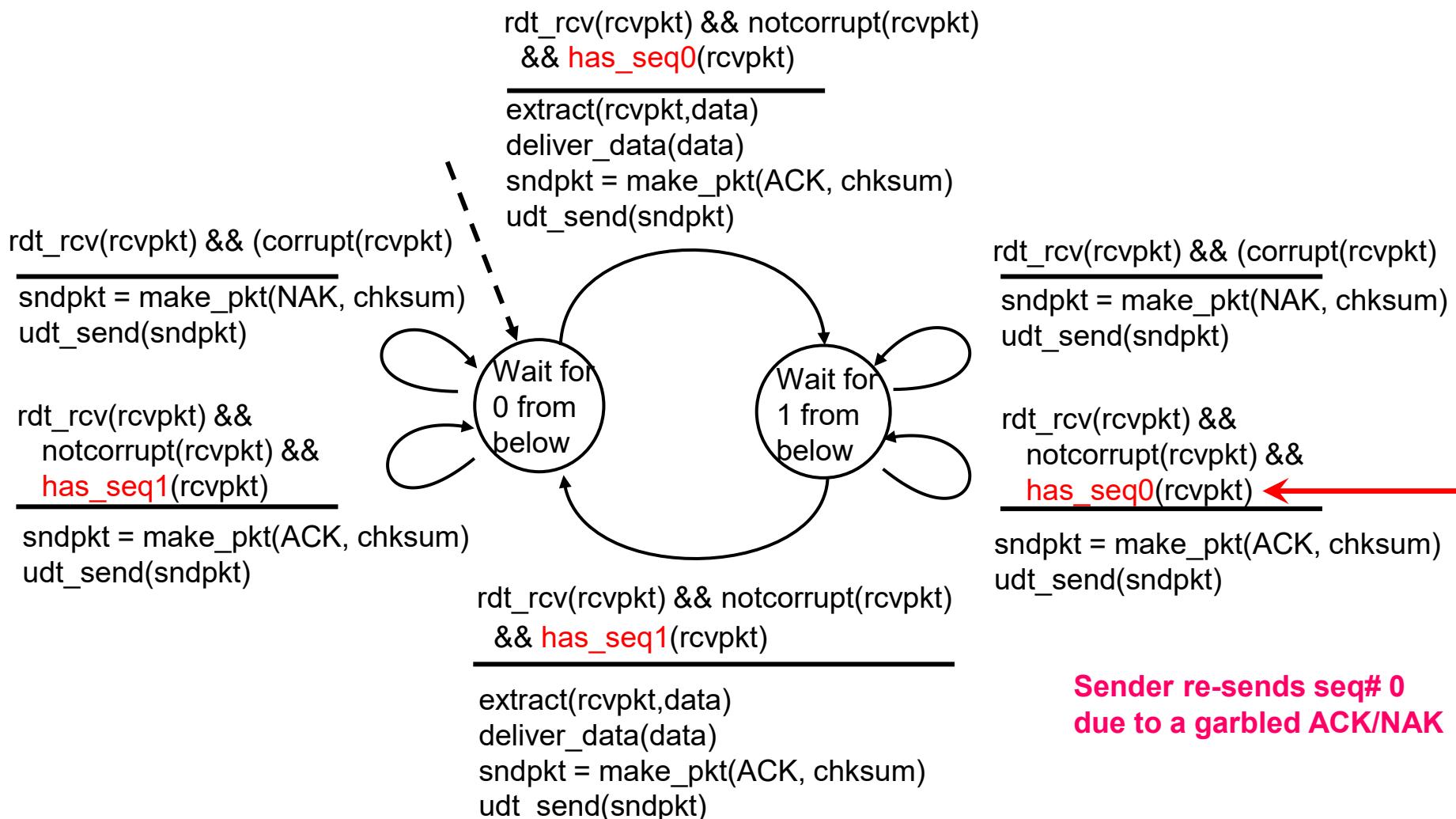
# RDT 2.1

## Odesilatel



# RDT 2.1

## Příjemce



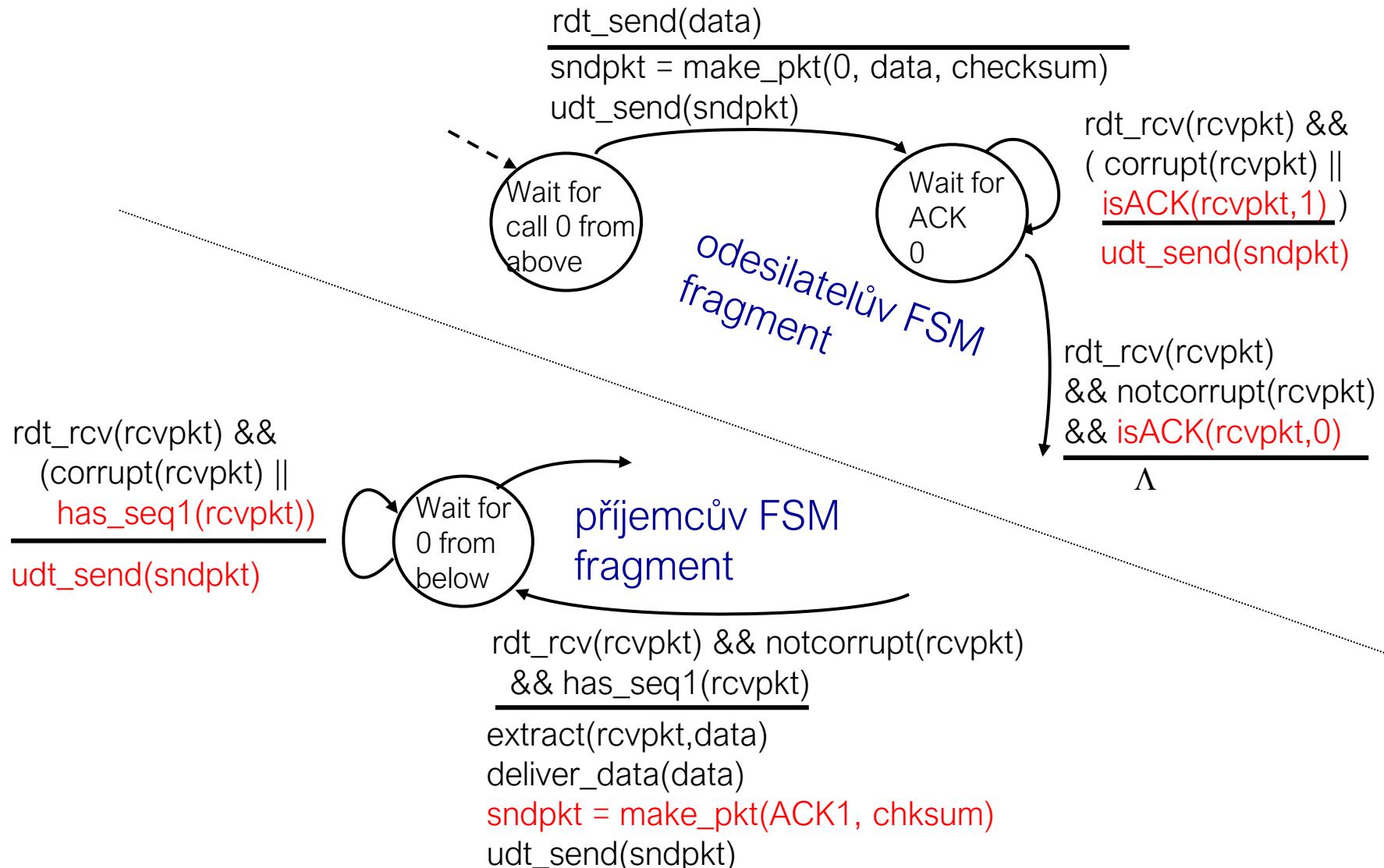
## RDT 2.2

### Pouze pozitivní potvrzování

- Stejná funkcionalita 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

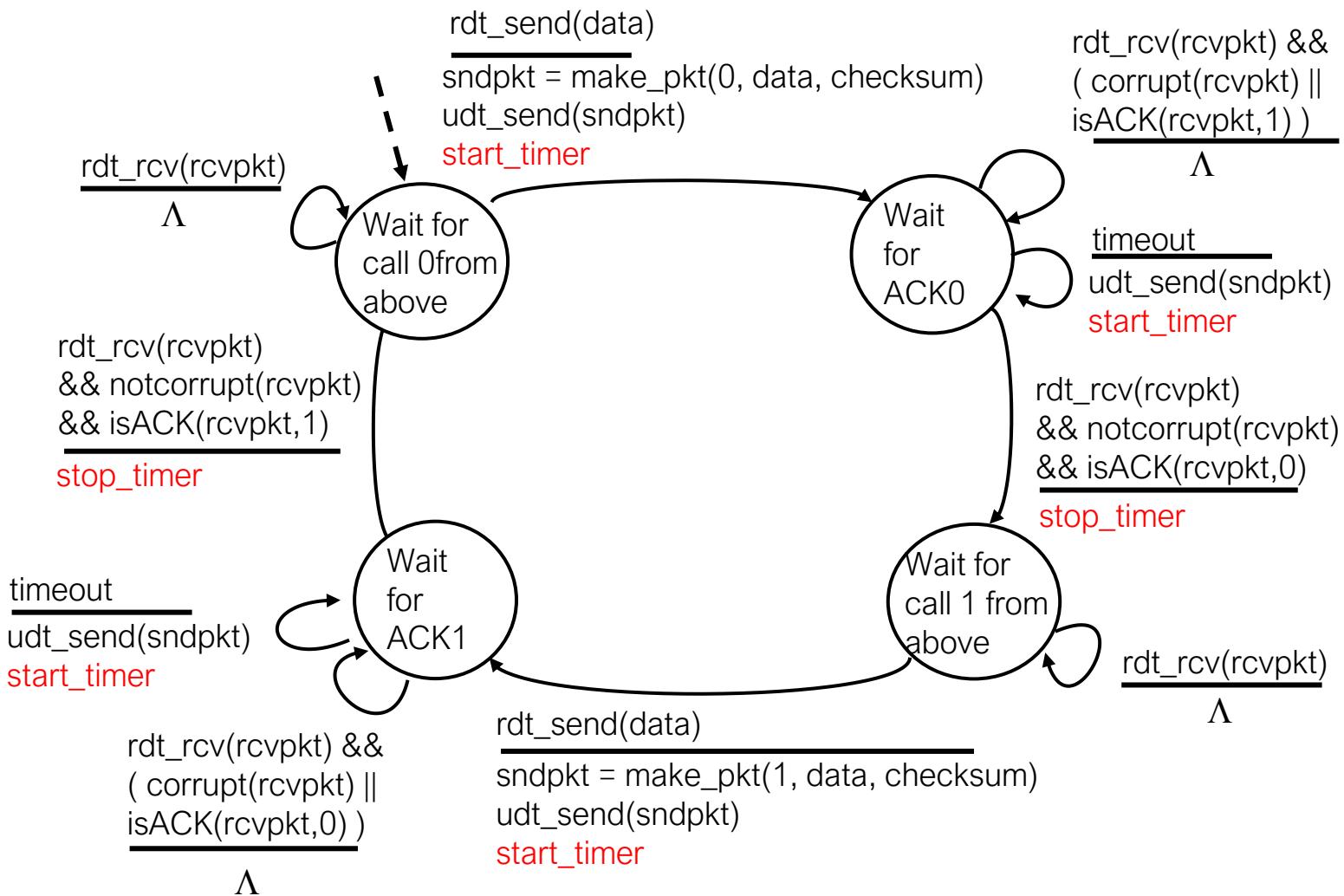


## (kanál s chybami)

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

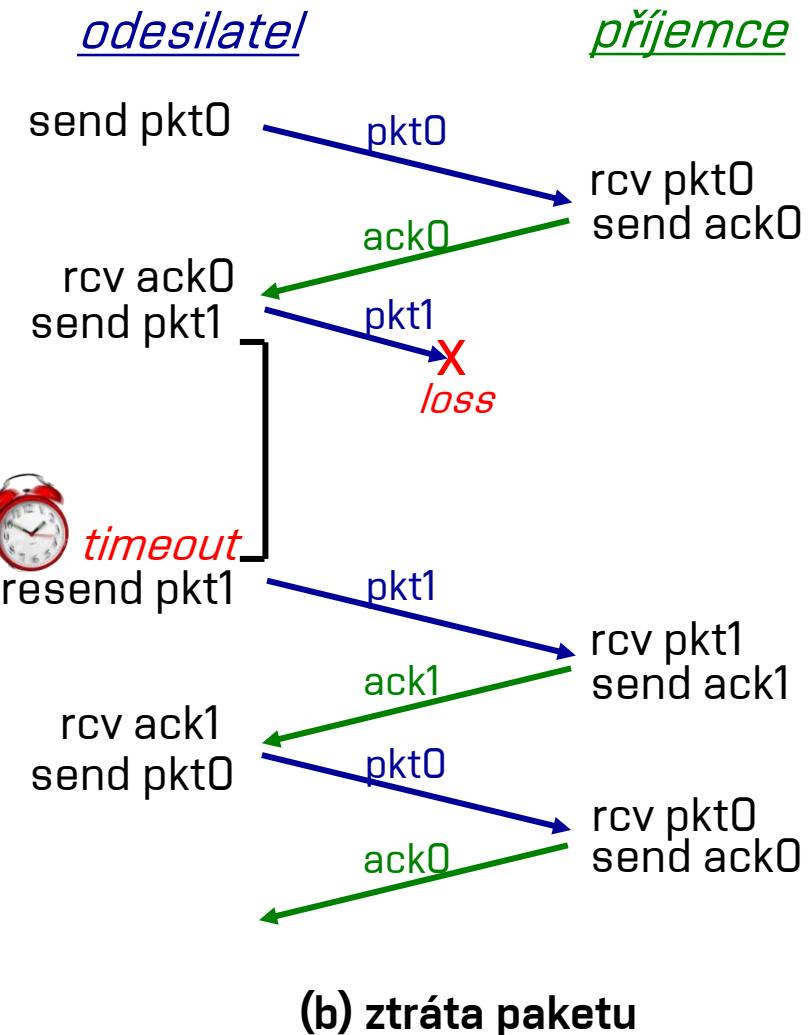
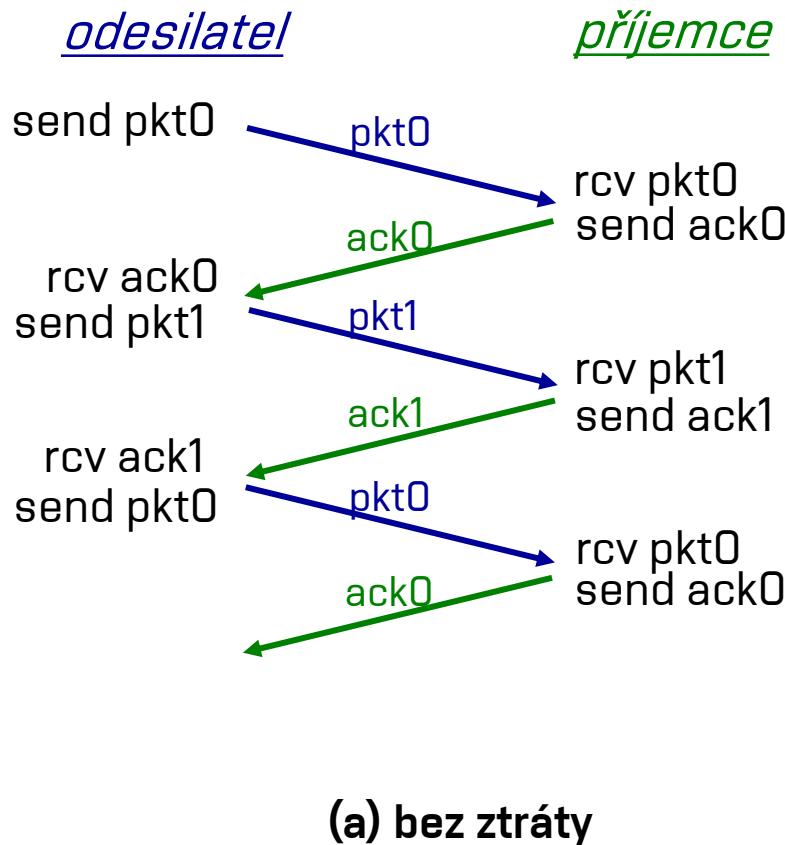
# RDT 3.0

## Odesilatel



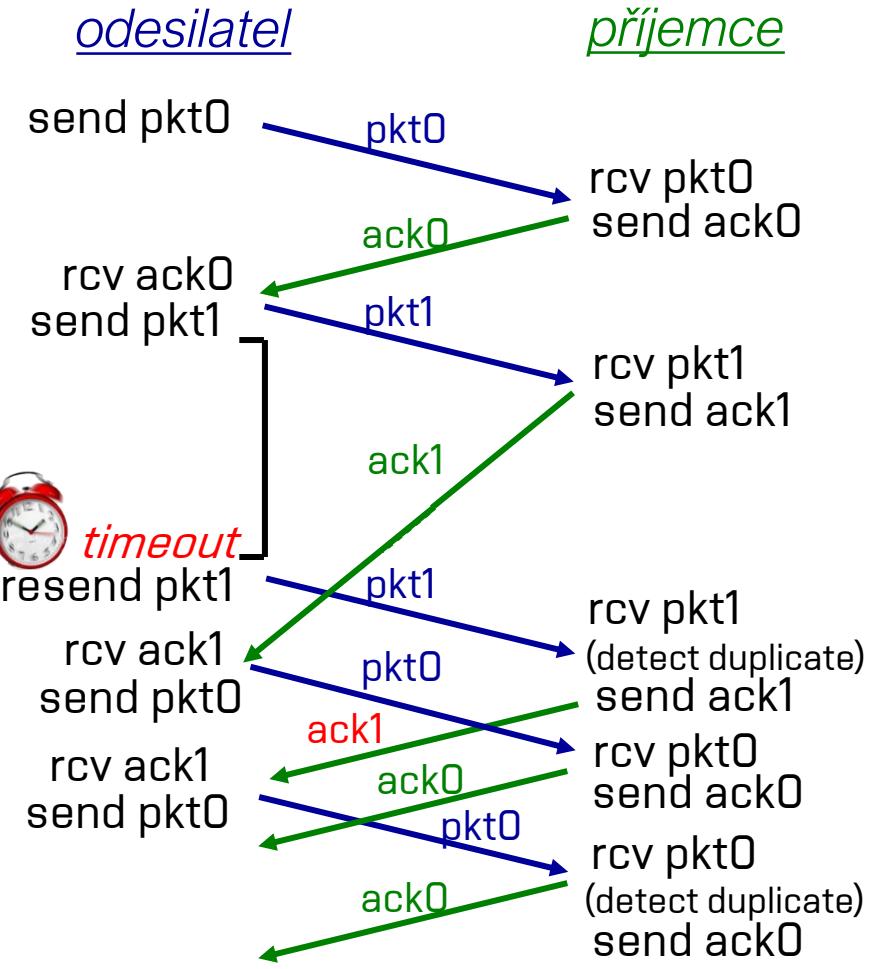
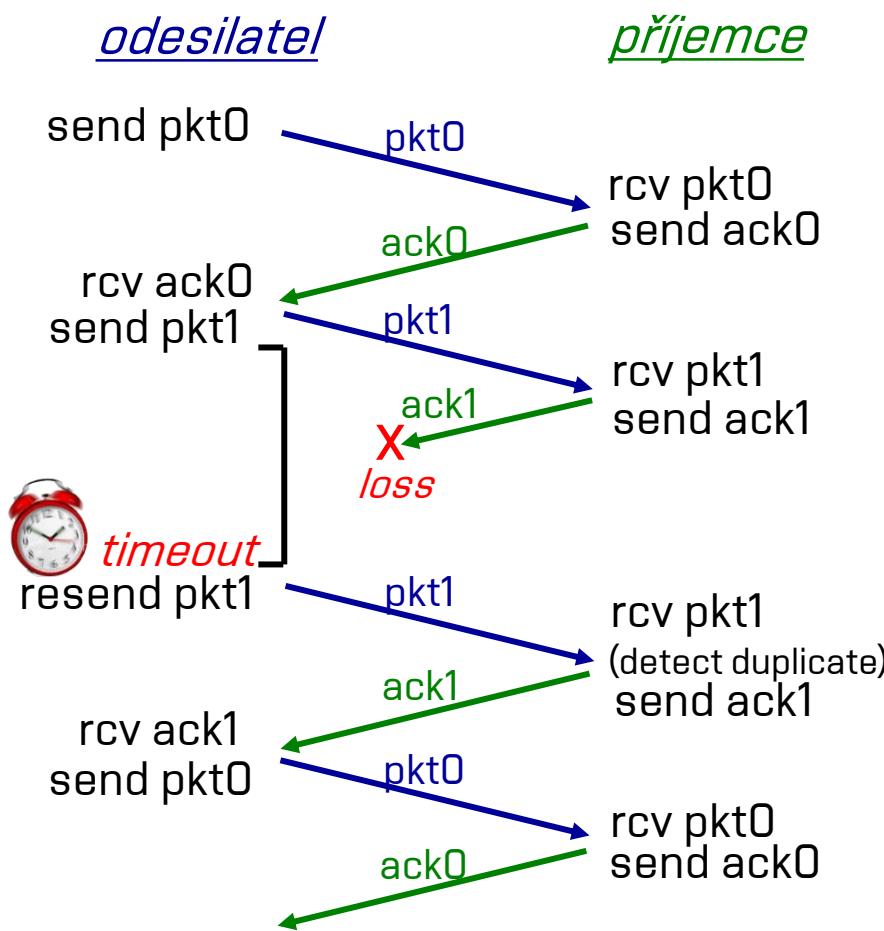
# RDT 3.0

## Scénáře (bez ztráty a se ztrátou)



# RDT 3.0

## Scénáře (ztráta potvrzení, zpoždění)



## 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]

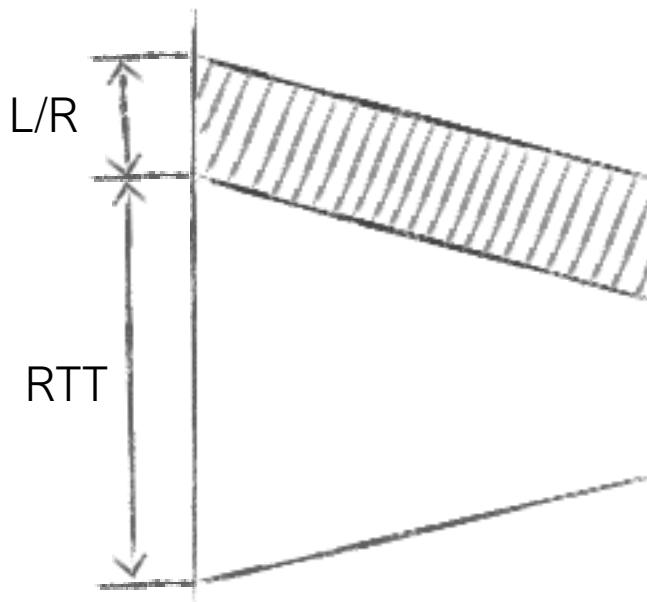
R - přenosová rychlosť [bps]

RTT – round trip time

U – využití

Například:

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



# Obsah

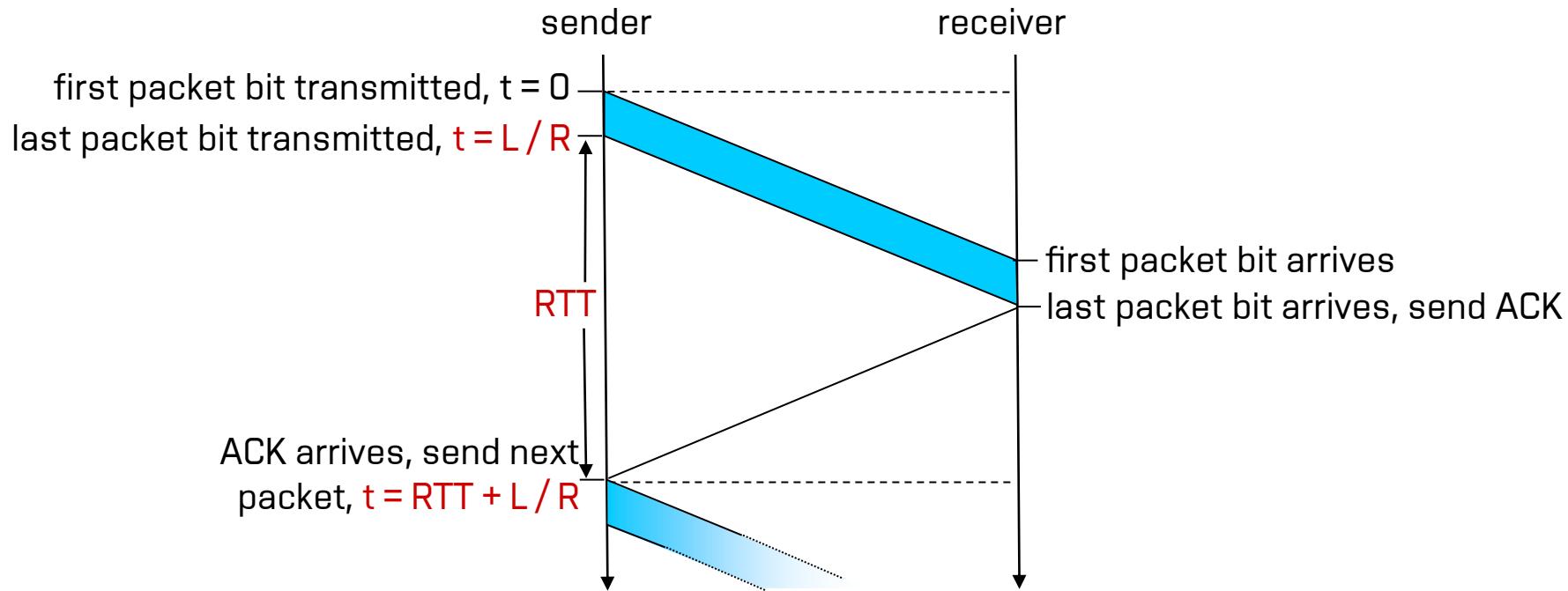
## 1) PROTOKOL UDP

## 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)

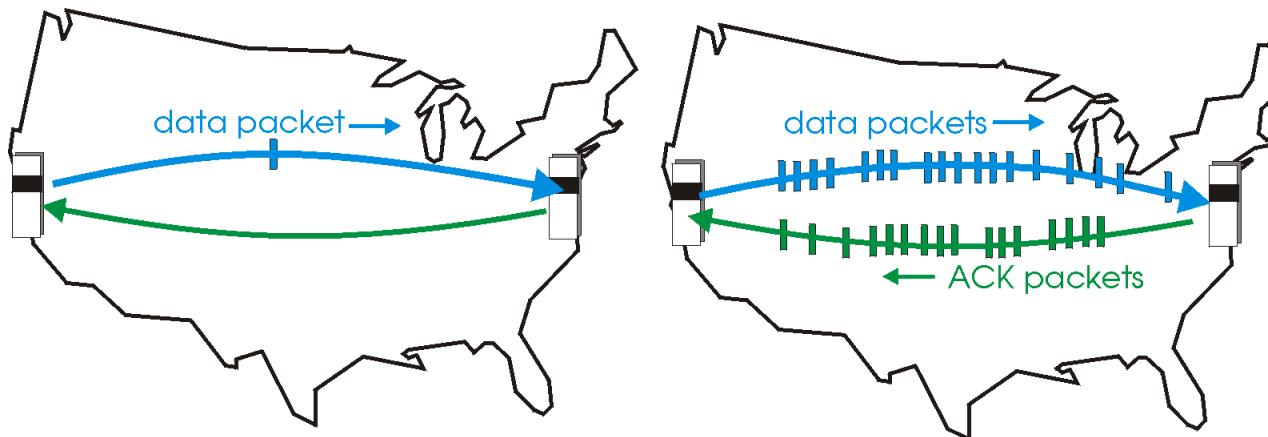


$$U = \frac{L/R}{RTT + L/R}$$

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

# Zřetězené protokoly

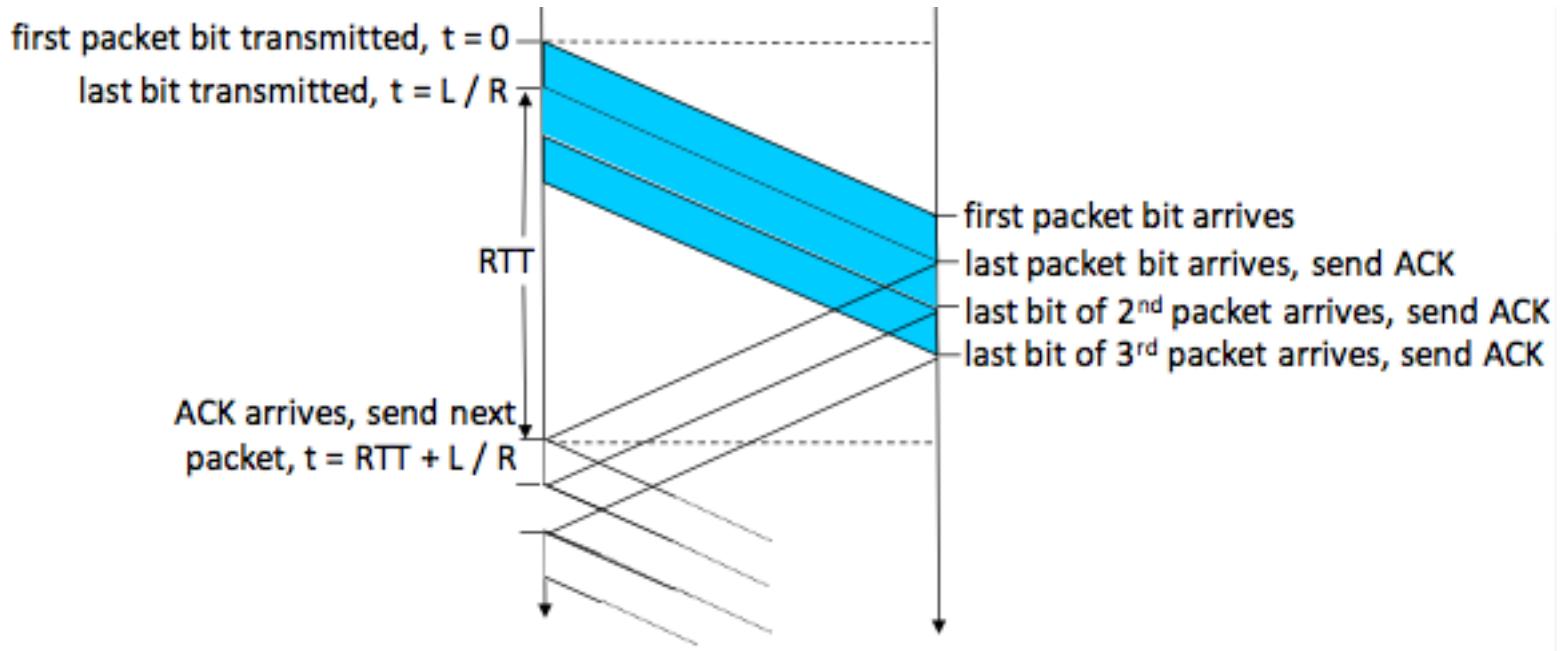
- Zřetězení (**pipelining**) umožňuje poslat více paketů, aniž přišlo potvrzení pro předchozí
  - číslování paketů nemůže být pouze 0 a 1
  - vyrovnávací paměti jsou potřeba u odesílatele i příjemce
- Dva druhy **zřetězených protokolů**:
  - go-back-N
  - selective-repeat



(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

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

## Selective Repeat

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

# Obsah

## 1) PROTOKOL UDP

## 2) SPOLEHLIVÝ PŘENOS

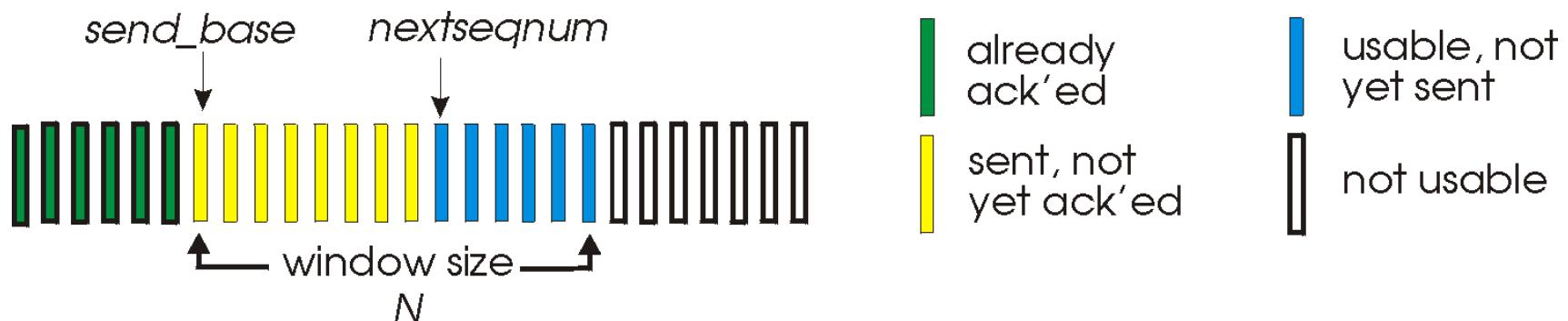
- Principy spolehlivého přenosu
- Stop-and-wait
- Pipelined protokoly
- Go-Back-N
- Selective Repeat

## 3) PROTOKOL TCP

# Go-back-N

Odesilatel:

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



# Go-back-N chování

sender window (N=4)

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

odesíatel

send pkt0  
send pkt1  
send pkt2  
send pkt3  
(wait)

příjemce

receive pkt0, send ack0  
receive pkt1, send ack1

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

rcv ack0, send pkt4  
rcv ack1, send pkt5

receive pkt3, discard,  
(re)send ack1

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

ignore duplicate ACK  
 *pkt 2 timeout*

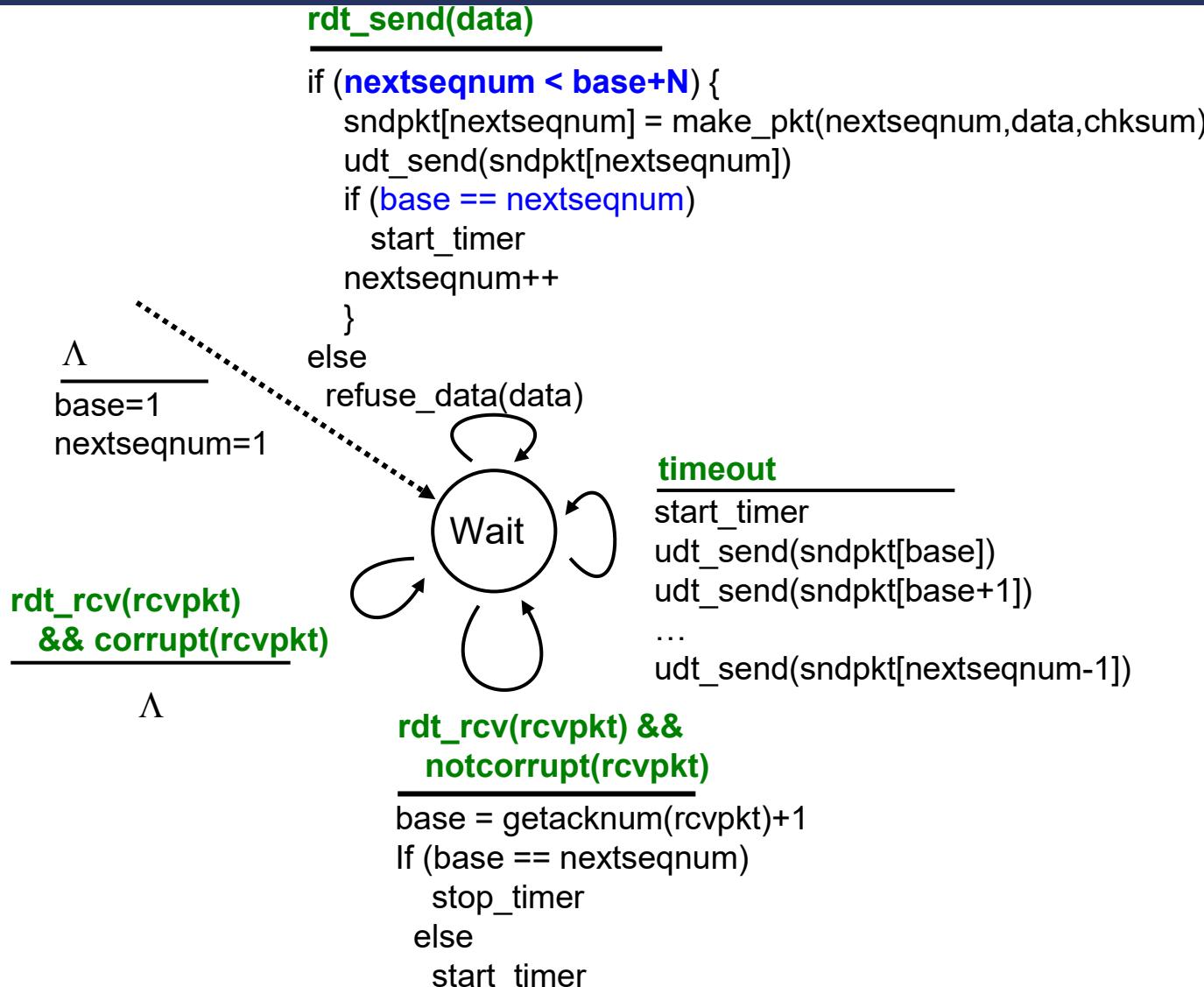
receive pkt4, discard,  
(re)send ack1  
receive pkt5, discard,  
(re)send ack1

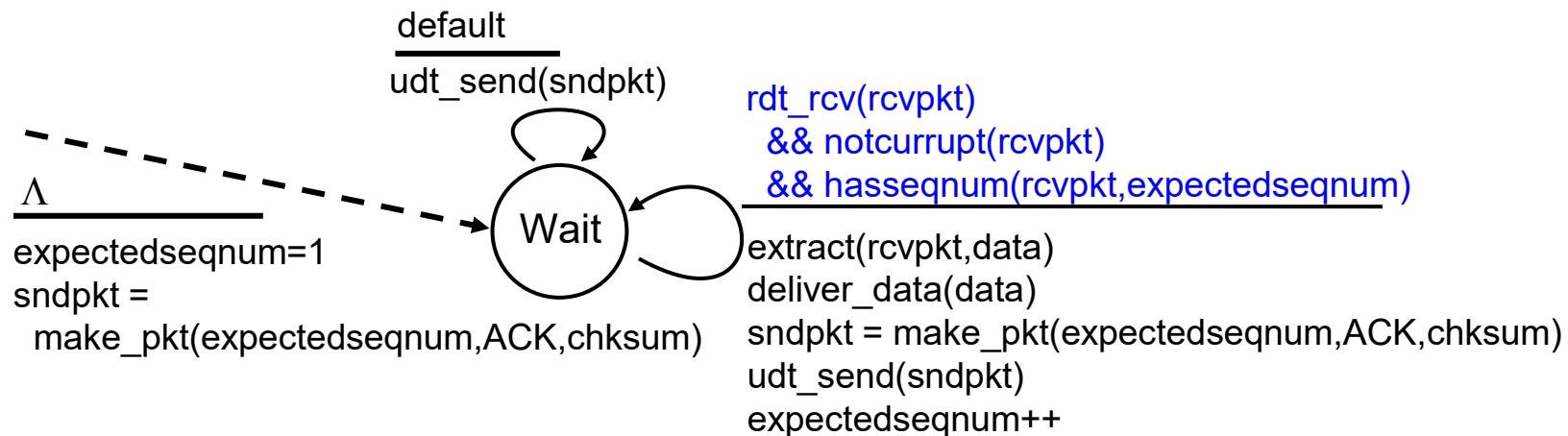
send pkt2  
send pkt3  
send pkt4  
send pkt5

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

# GBN

## Odesilatel





- 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

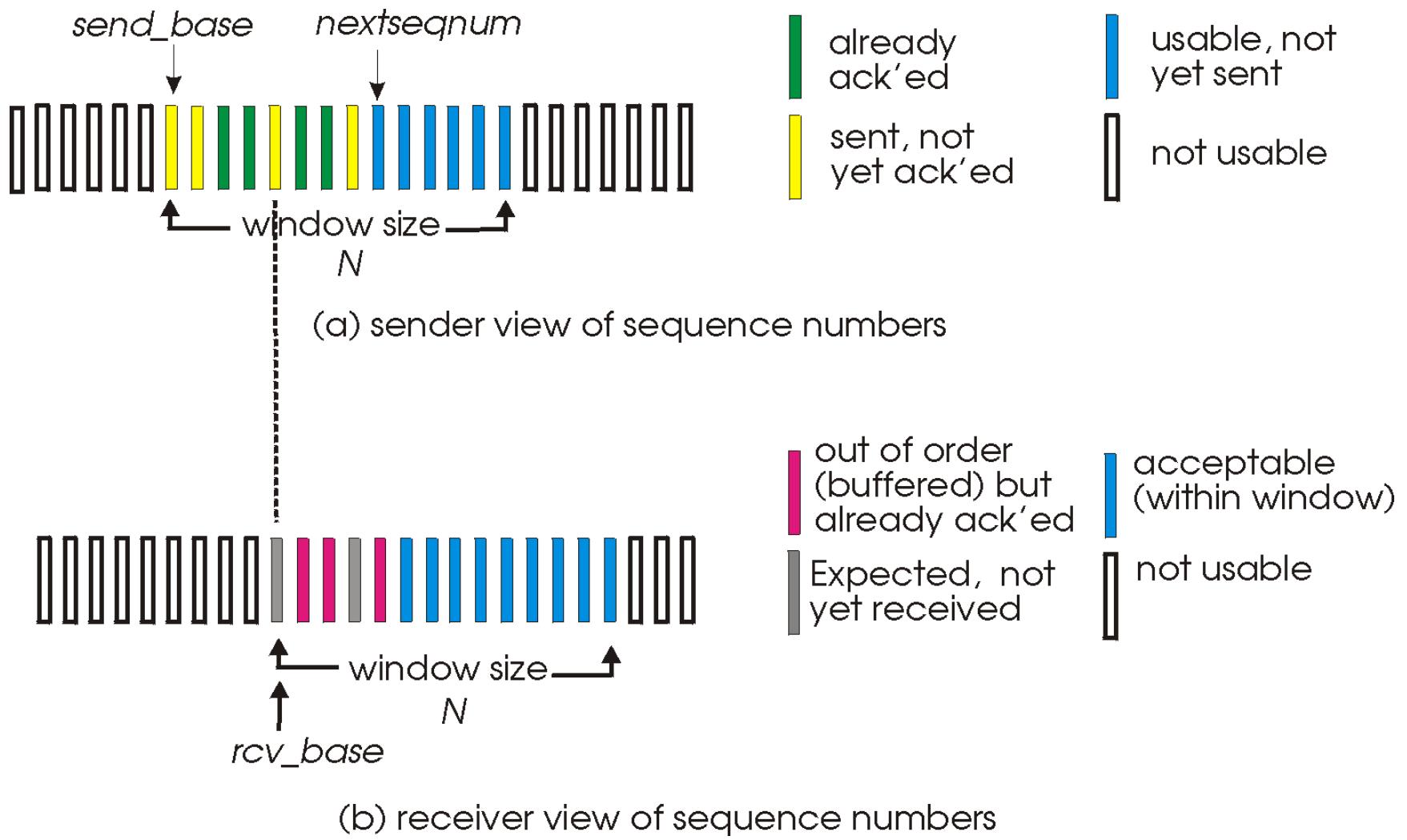
- Principy spolehlivého přenosu
- Stop-and-wait
- Pipelined protokoly
- Go-Back-N
- Selective Repeat

## 3) PROTOKOL TCP

## Obecně

- Příjemce selektivně potvrzuje všechny správně přijaté pakety
  - Pakety jsou uchovány ve vyrovnávací paměti, kde po doplnění jsou předány vyšší vrstvě
- Odesilatel znovuzasílá pouze pakety pro které neobdržel potvrzení
  - časovač pro každý nepotvrzený paket
- Okénko odesilatele
  - $N$  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

## Odesilatel

### 1) data od vyšší vrstvy:

- je-li volné místo v okně, pošli paket a nastav časovač

### 2) timeout( $n$ )

- znova 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) $n \in (rcv\_base, rcv\_base+N-1)$ :

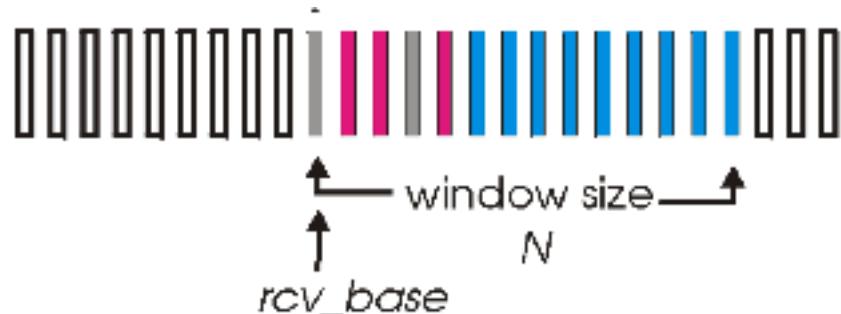
- pošli ACK( $n$ )
- ulož do bufferu, jedná-li se o první, potom doruč data a posuň okno

### 2) $n \in (rcv\_base-N, rcv\_base-1)$ :

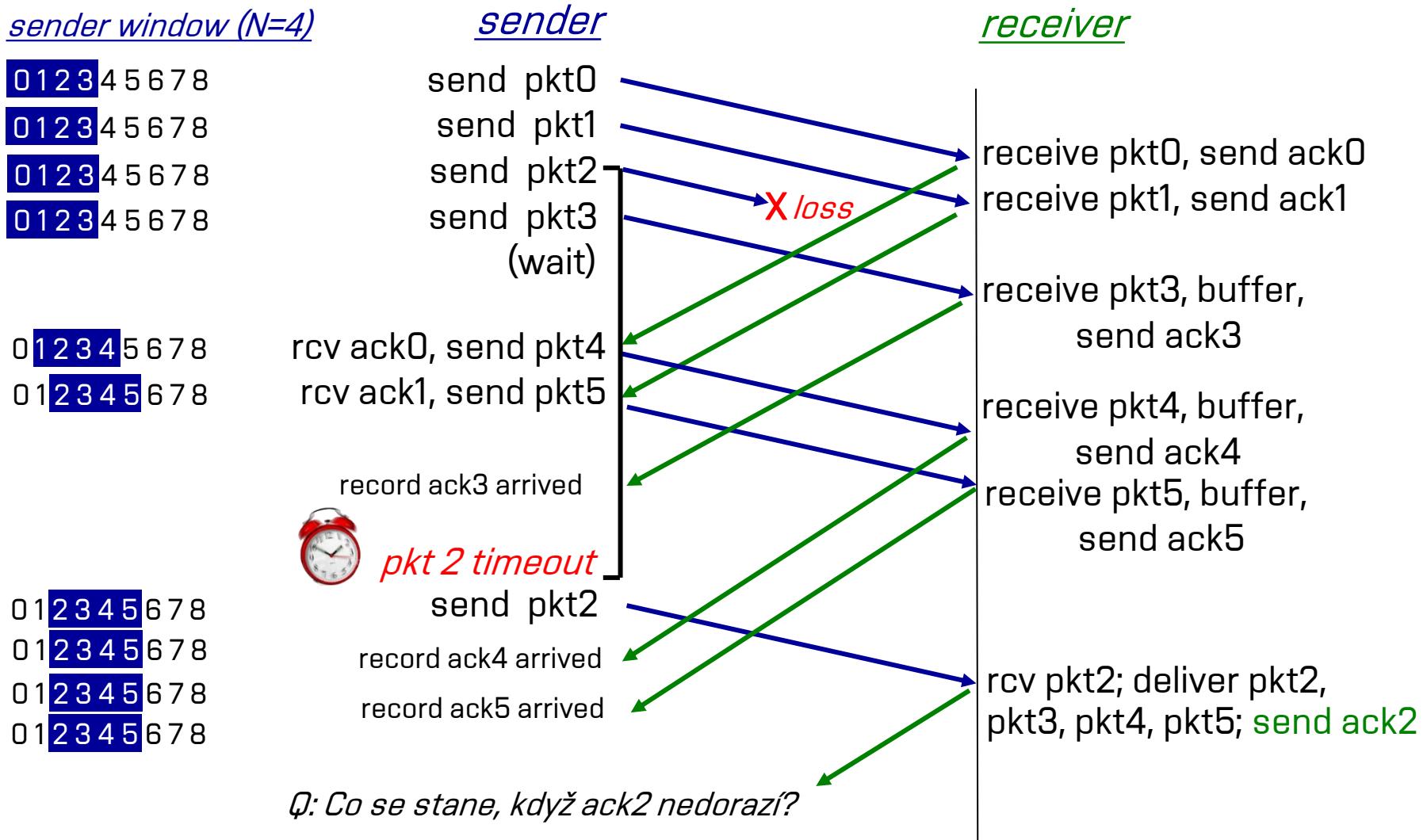
- pošli ACK( $n$ )

### 3) jinak:

- ignoruj paket



# Selective Repeat Demo



# Spolehlivý přenos - principy

Mechanismus	Použití
Kontrolní součet	Detekce bitových chyb v přenášených segmentech.
Časovač	Pro detekci ztráty segmentů nebo jejich ACK.
Sekvenční číslo	Cílování segmentů. Detekce duplicit segmentů a segmentů, které chybí.
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 odesilatel 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 straně odesilatele i příjemce



# Transmission Control Protocol

- TCP [RFC 793]

[Search] [txt|html|pdf|with errata|bibtex] [Tracker] [Email] [Diff1] [Diff2] [Nits]

Updated by: [1122](#), [3168](#), [6093](#), [6528](#)

RFC: 793

Internet Standard  
[Errata exist](#)

TRANSMISSION CONTROL PROTOCOL

DARPA INTERNET PROGRAM

PROTOCOL SPECIFICATION

September 1981

prepared for

Defense Advanced Research Projects Agency  
Information Processing Techniques Office  
1400 Wilson Boulevard  
Arlington, Virginia 22209

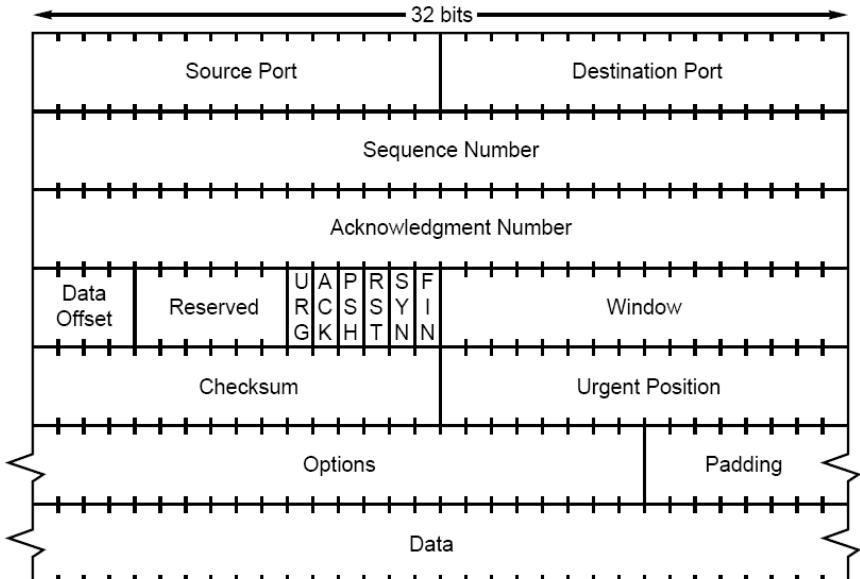
September 1981

Transmission Control Protocol

TABLE OF CONTENTS

PREFACE .....	<a href="#">iii</a>
1. INTRODUCTION .....	<a href="#">1</a>
1.1 Motivation .....	<a href="#">1</a>
1.2 Scope .....	<a href="#">2</a>
1.3 About This Document .....	<a href="#">2</a>
1.4 Interfaces .....	<a href="#">3</a>
1.5 Operation .....	<a href="#">3</a>
2. PHILOSOPHY .....	<a href="#">7</a>
2.1 Elements of the Internetwork System .....	<a href="#">7</a>
2.2 Model of Operation .....	<a href="#">7</a>
2.3 The Host Environment .....	<a href="#">8</a>
2.4 Interfaces .....	<a href="#">9</a>
2.5 Relation to Other Protocols .....	<a href="#">9</a>
2.6 Reliable Communication .....	<a href="#">9</a>
2.7 Connection Establishment and Clearing .....	<a href="#">10</a>
2.8 Data Communication .....	<a href="#">12</a>
2.9 Precedence and Security .....	<a href="#">13</a>
2.10 Robustness Principle .....	<a href="#">13</a>
3. FUNCTIONAL SPECIFICATION .....	<a href="#">15</a>
3.1 Header Format .....	<a href="#">15</a>
3.2 Terminology .....	<a href="#">19</a>
3.3 Sequence Numbers .....	<a href="#">24</a>
3.4 Establishing a connection .....	<a href="#">30</a>
3.5 Closing a Connection .....	<a href="#">37</a>
3.6 Precedence and Security .....	<a href="#">40</a>
3.7 Data Communication .....	<a href="#">40</a>
3.8 Interfaces .....	<a href="#">44</a>
3.9 Event Processing .....	<a href="#">52</a>
GLOSSARY .....	<a href="#">79</a>
REFERENCES .....	<a href="#">85</a>

# 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 aplikacní 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 oktetů)
- **Volby (Options):** volitelné parametry spojení, MSS: maximální velikost
- **Padding:** vypávkva do mocniny 2
- **Data:** aplikacní 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	96	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

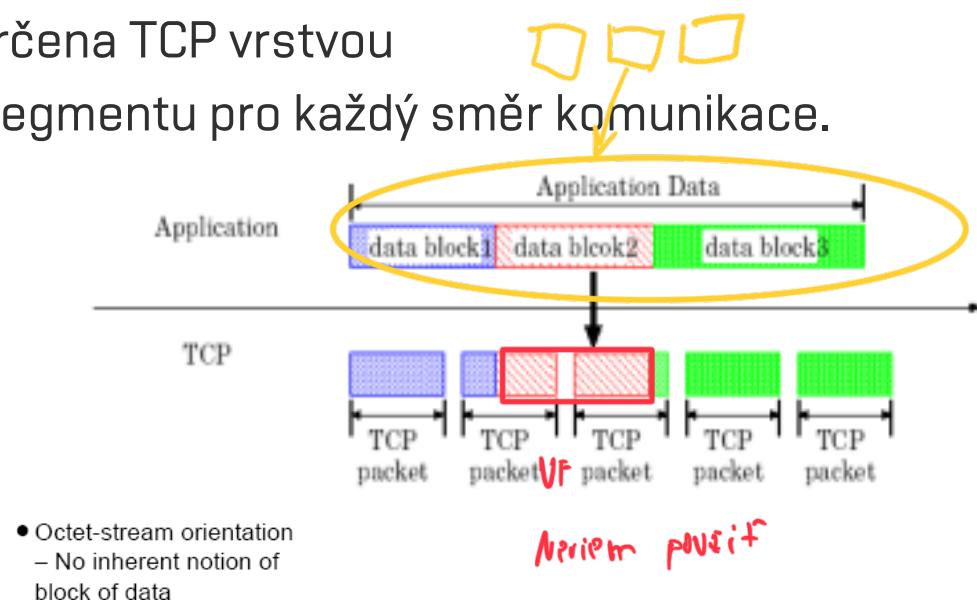
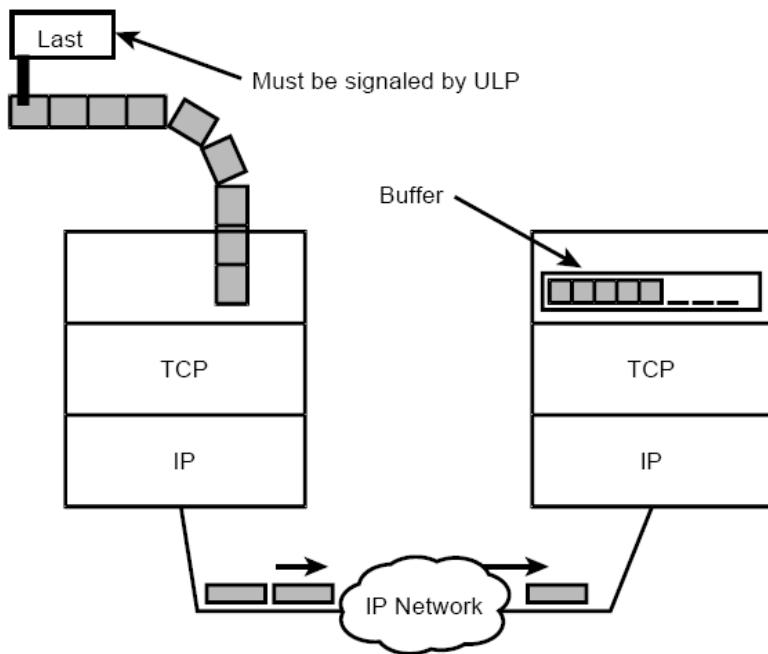
[SEQ/ACK analysis]

0000 00 04 96 1d 4e 30 78 24 af 8a 95 6e 08 00 45 00	....N0x\$ ...n..E.
0010 00 29 51 1d 40 00 80 06 00 00 93 e5 b5 53 36 98	.)Q.@.... ....56.
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 Service

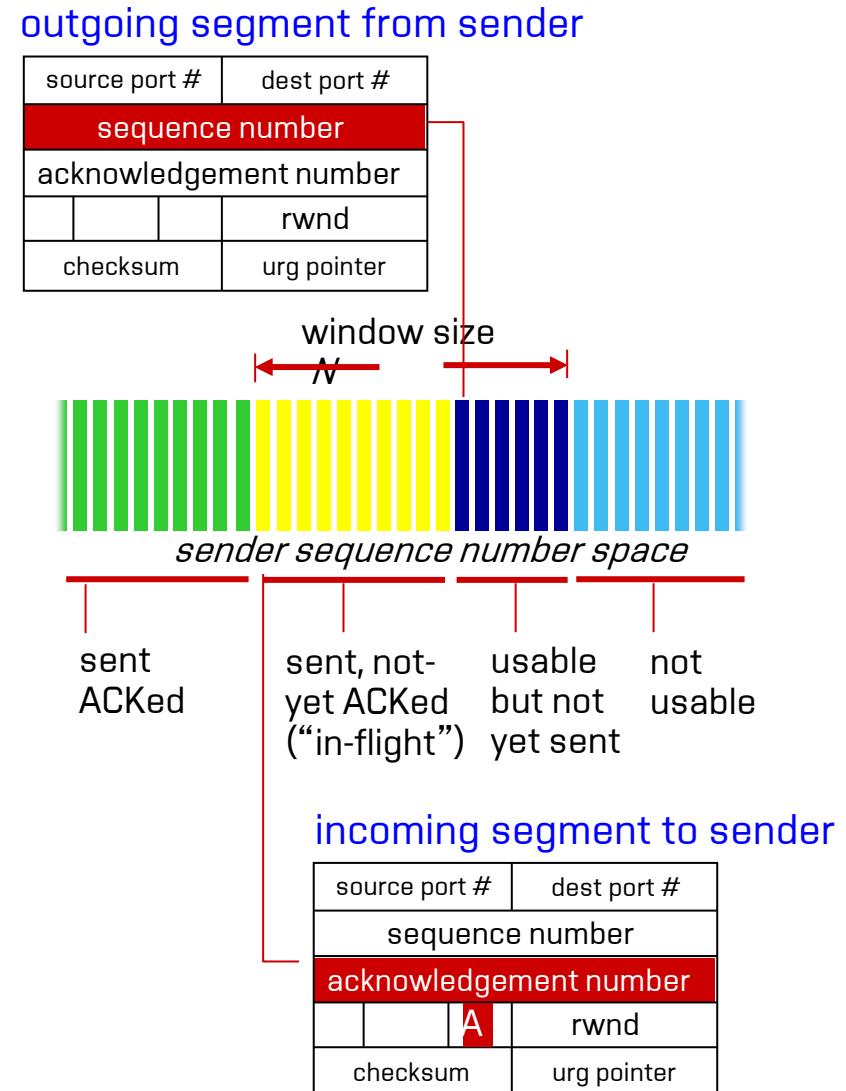
- Data přenášená TCP segmentem nemají žadnou strukturu
  - představuje proud bitů/bytů
  - rozděluje data od aplikace do posloupnosti segmentů
  - velikost těchto segmentů je určena TCP vrstvou
  - TCP určuje vhodnou velikost segmentu pro každý směr komunikace.



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

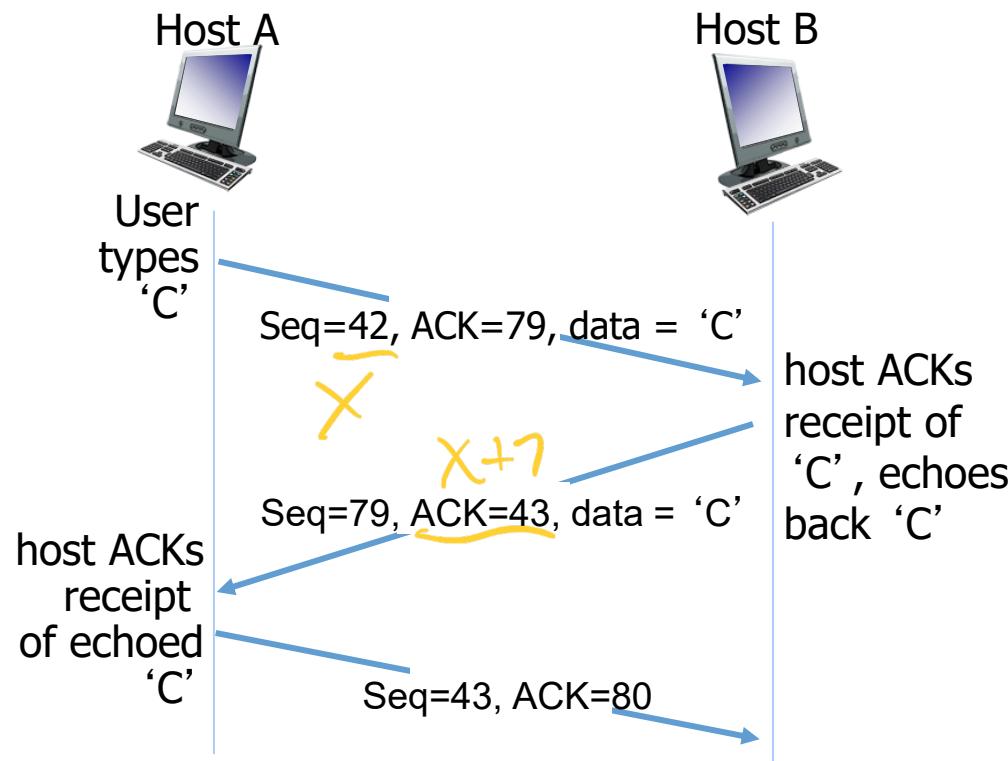
# TCP SEQ a ACK čísla

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



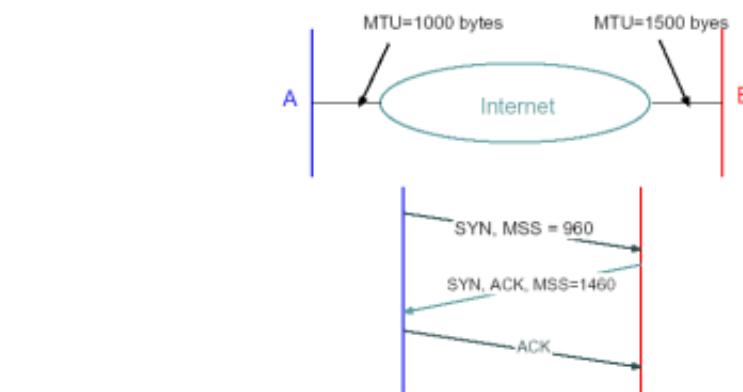
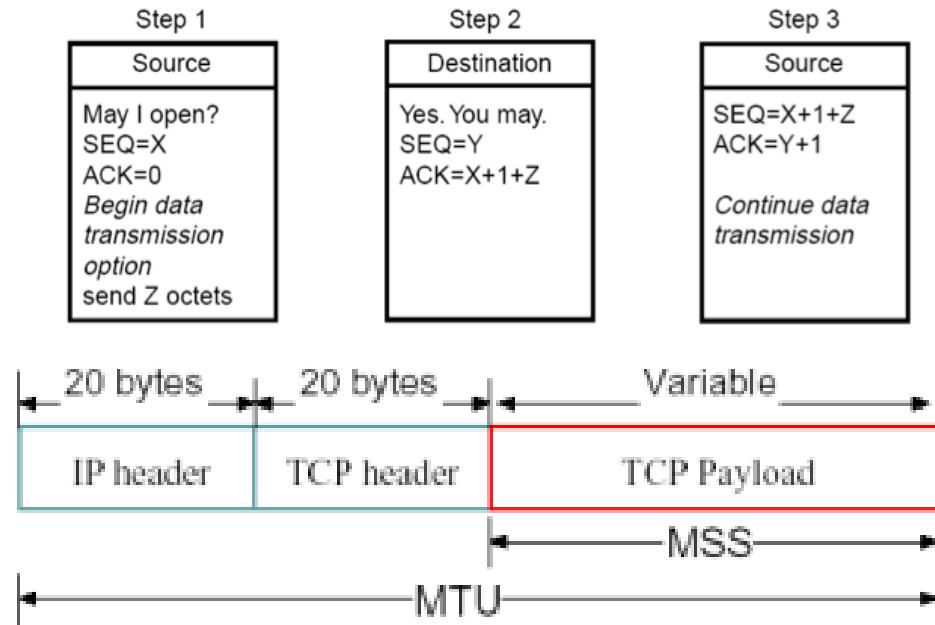
# Příklad potvrzování TCP

- 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 Echo Reply
-----	-----	---	----	-----------------	----------------------

## ➤ Selective ACK Permitted [SYN packet only]

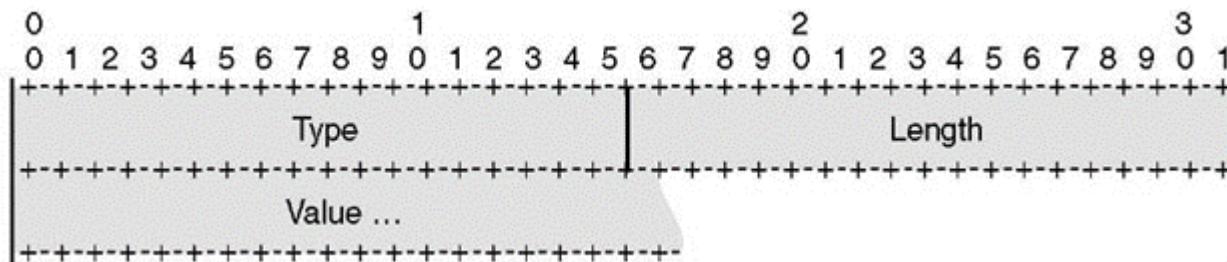
NOP	NOP	4	2
-----	-----	---	---

## ➤ 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

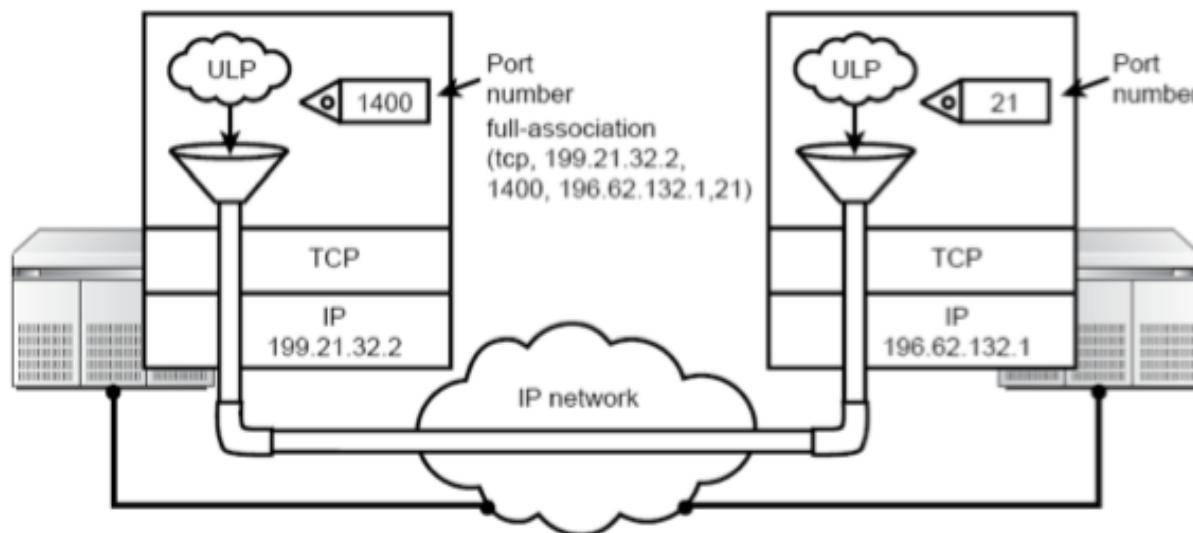
## 2) SPOLEHLIVÝ PŘENOS

## 3) PROTOKOL TCP

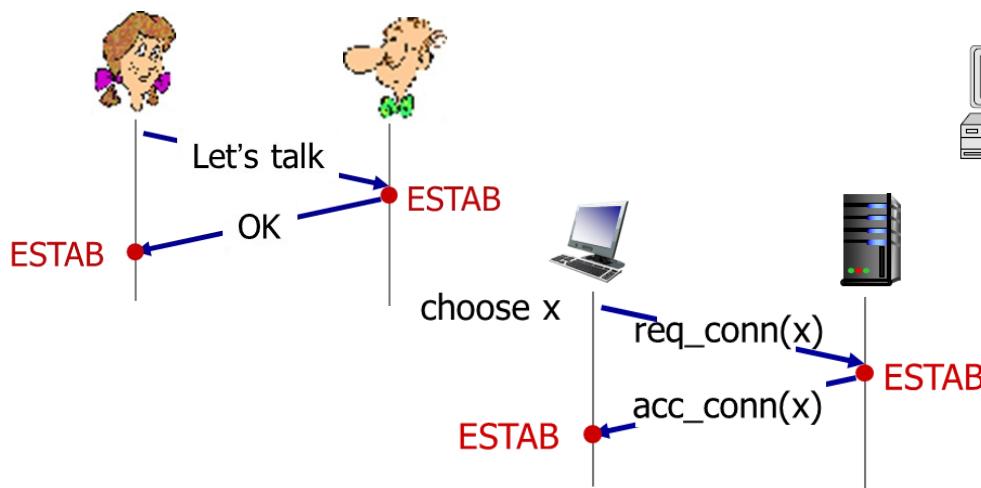
- Vlastnosti a funkce, formát segmentu
- Řízení spojení
- Řízení toku
- Řízení zahlcení

# Datový tok (flow)

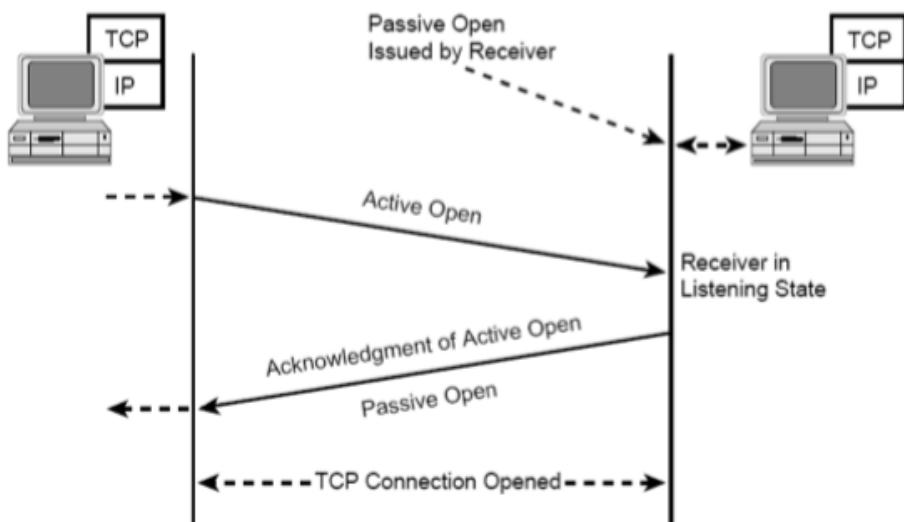
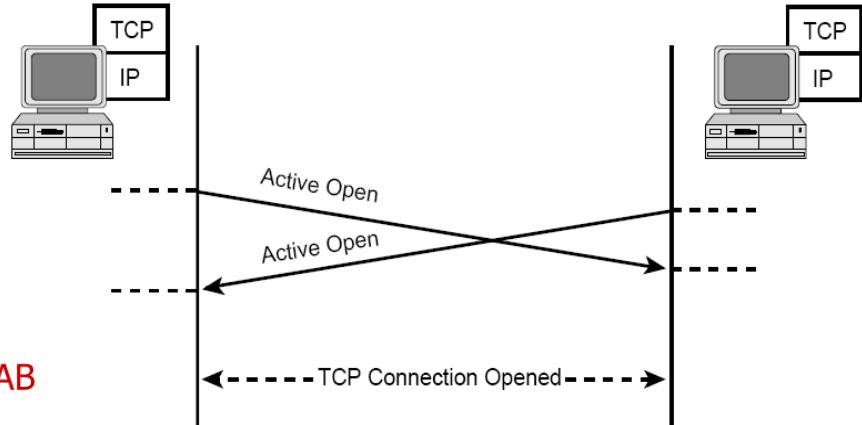
- Poloviční asociace (half association)
  - protokol transportní vrstvy + IP adresa + číslo portu
- Plná asociace (full association)
  - protokol transportní vrstvy + zdrojová IP adresa + zdrojové číslo portu + cílová IP adresa + cílové číslo portu



# Zahájení spojení

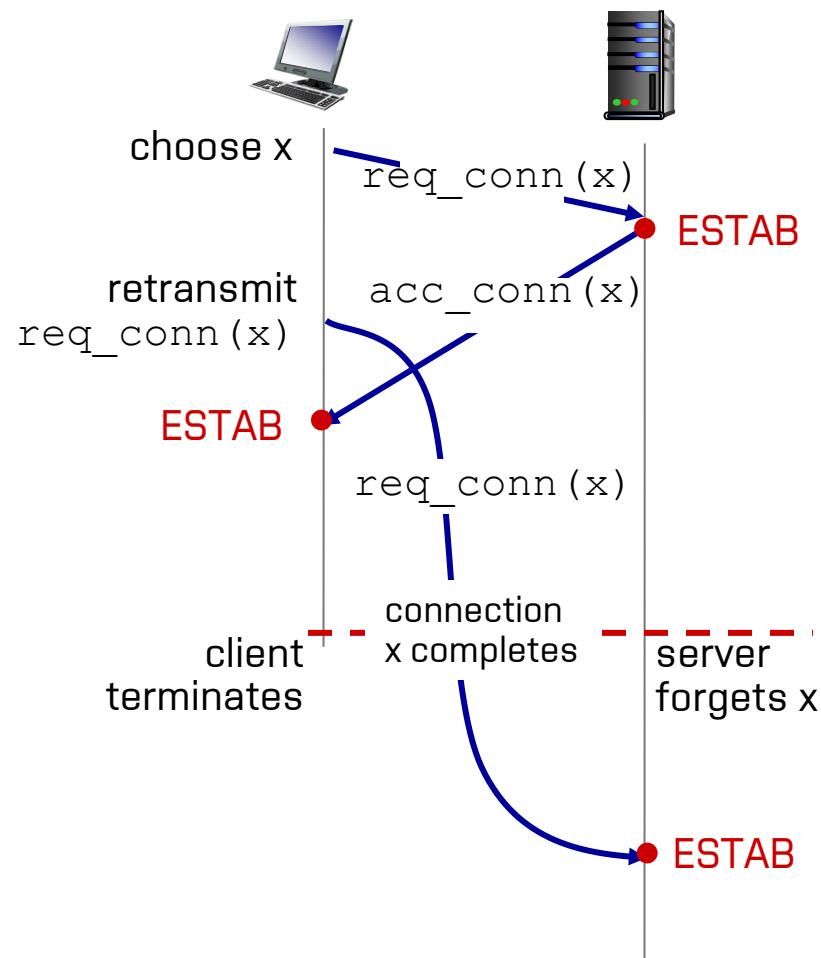


- Operace otevření spojení OPEN
  - Aktivní (Active OPEN)  
iniciuje spojení (typicky klient)
  - Pasivní (Passive OPEN)  
čeká na vytvoření spojení (typicky servis)
- Možné vytvořit spojení
  - [Active-Passive]
  - [Active-Active] (distribuované systémy)



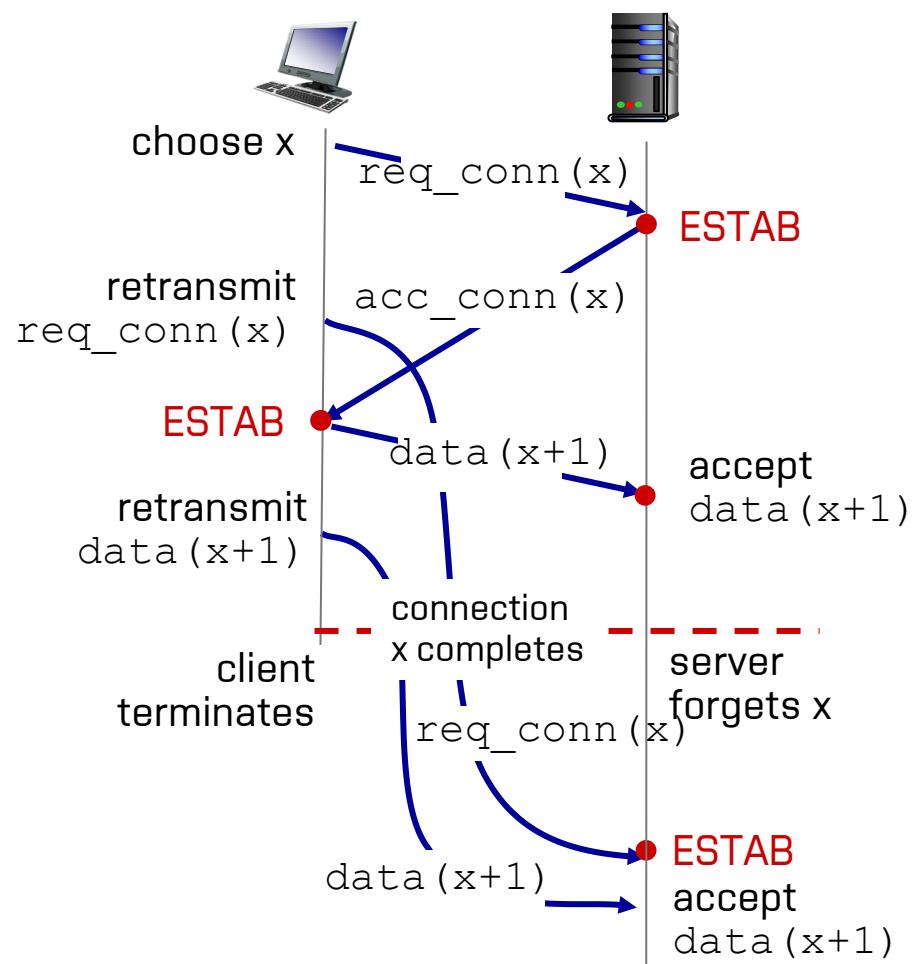
# 2-Way Handshake (je problematický)

a) zatoulaný retransmit



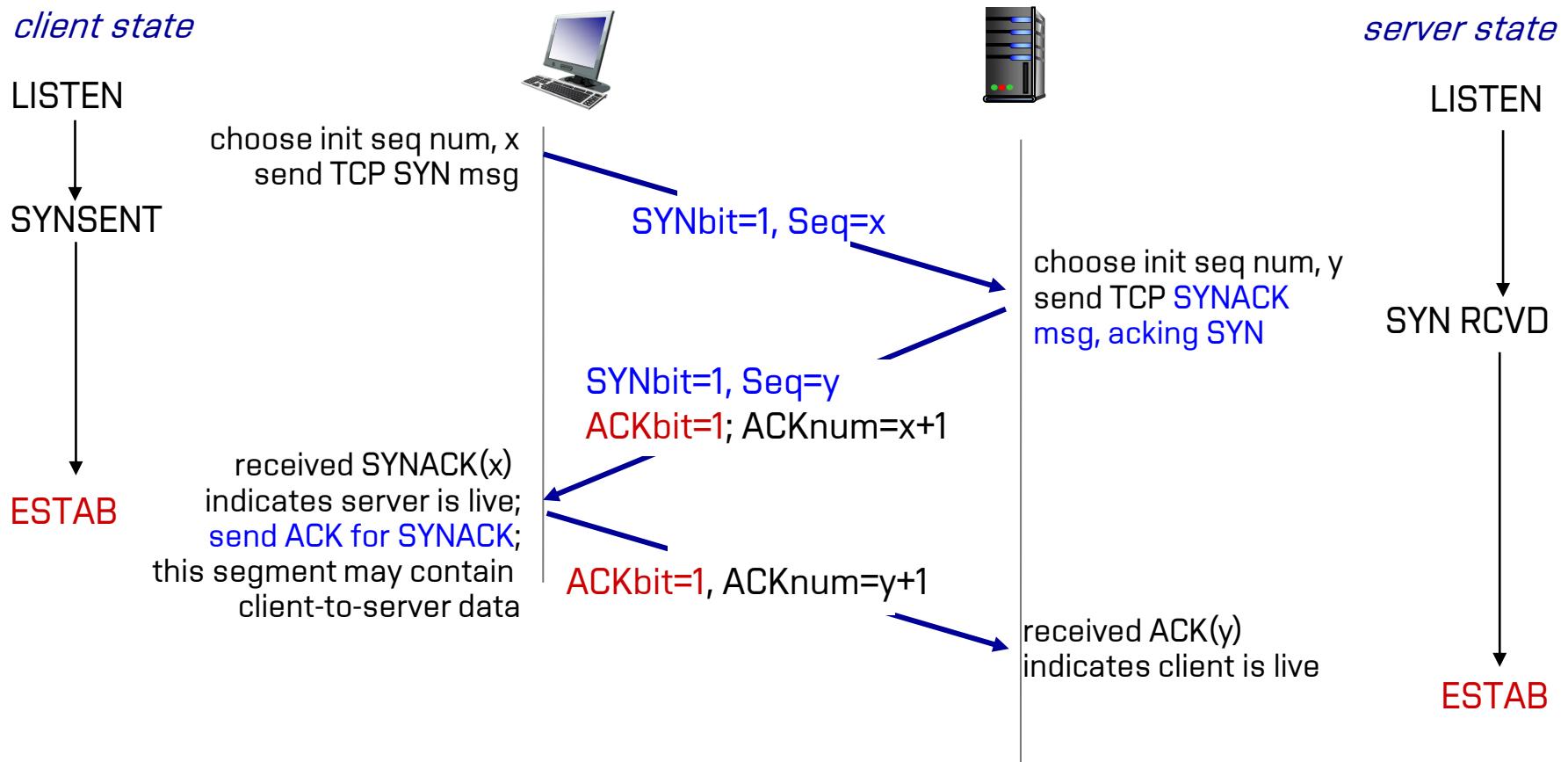
Napůl otevřené spojení!

b) zatoulaná data

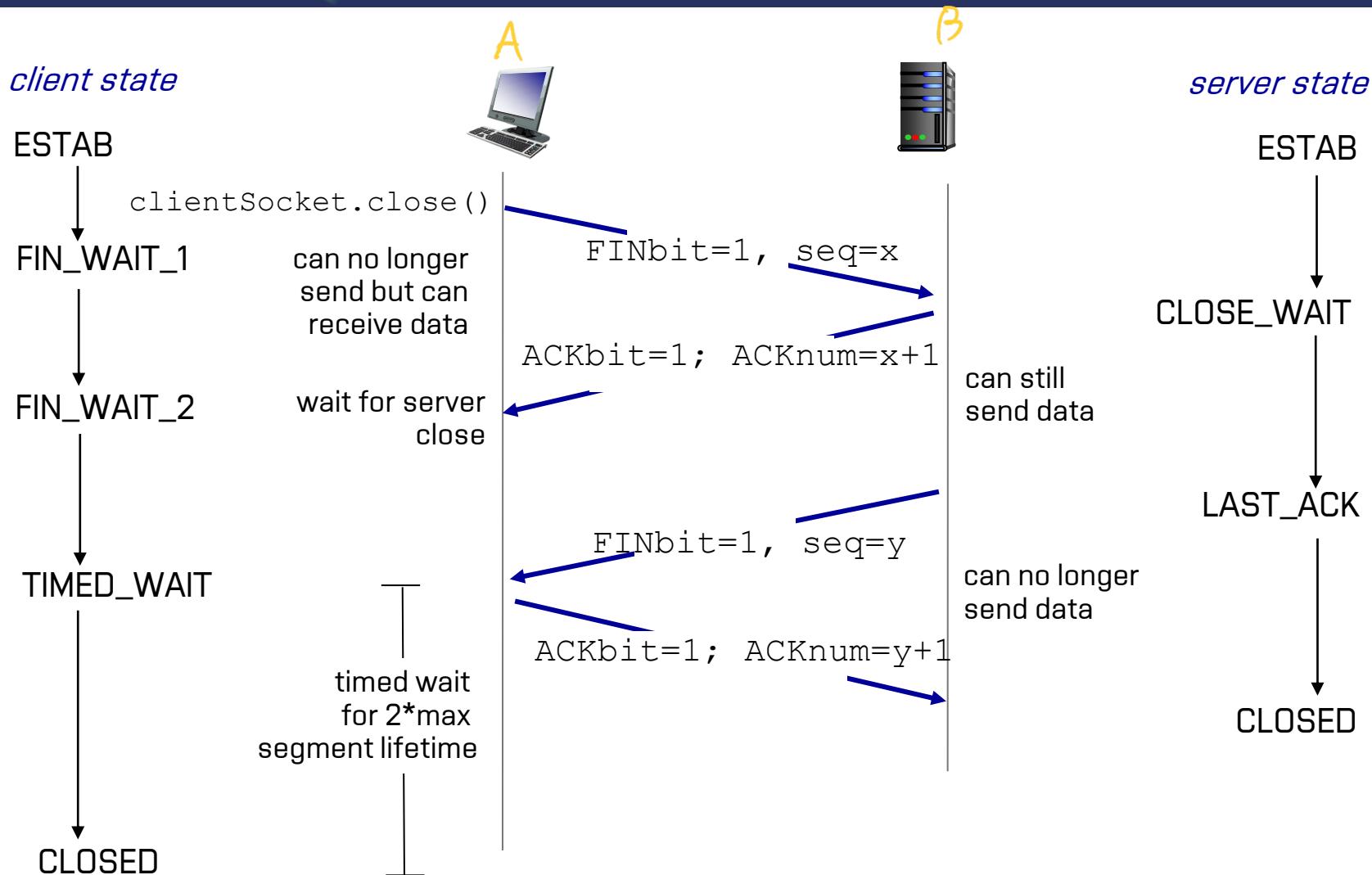


Vložení paketu!

# 3-Way Handshake (příznak SYN)

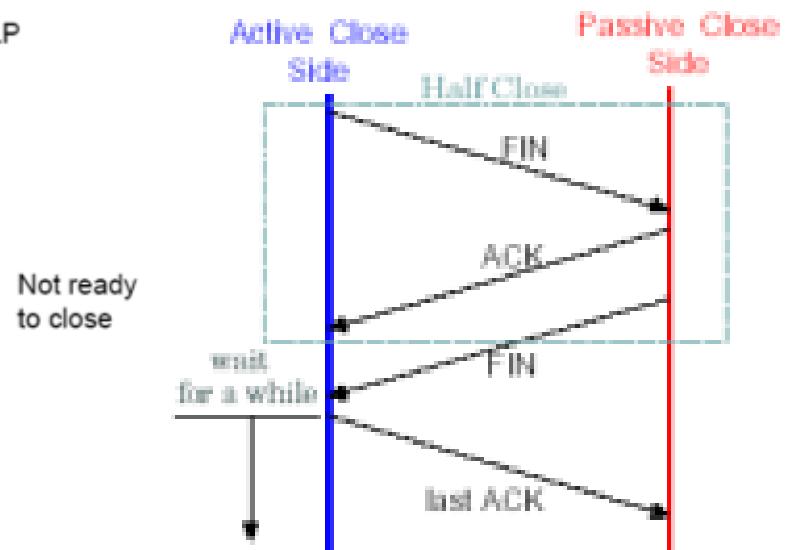
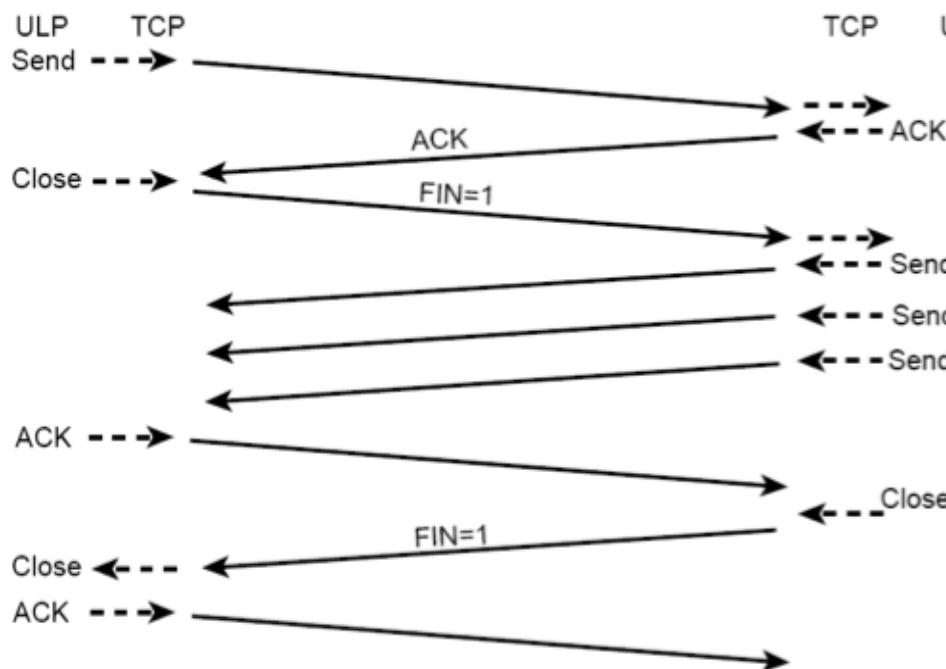


# Ukončení spojení (příznak FIN)

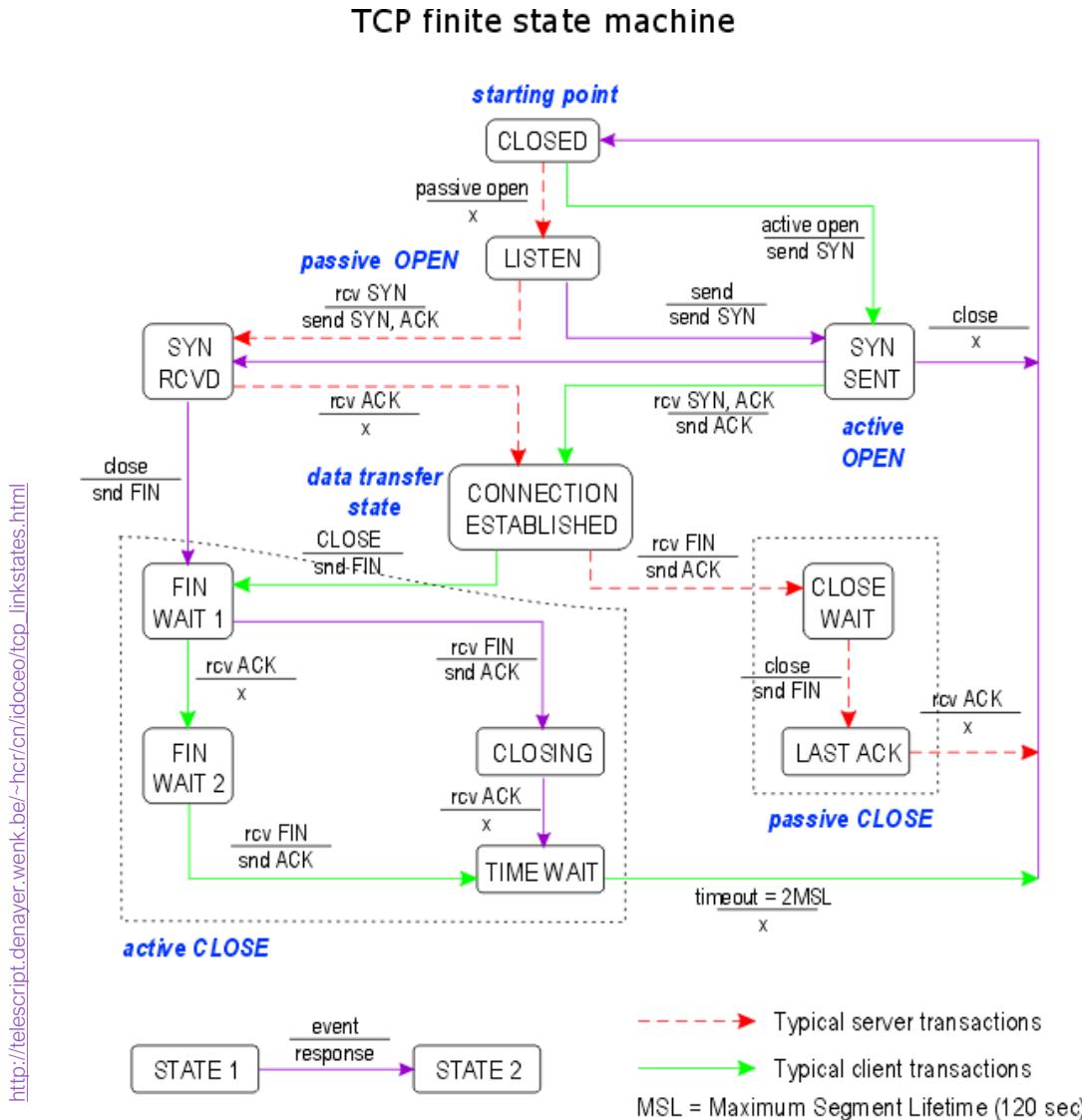


# Druhy ukončení spojení

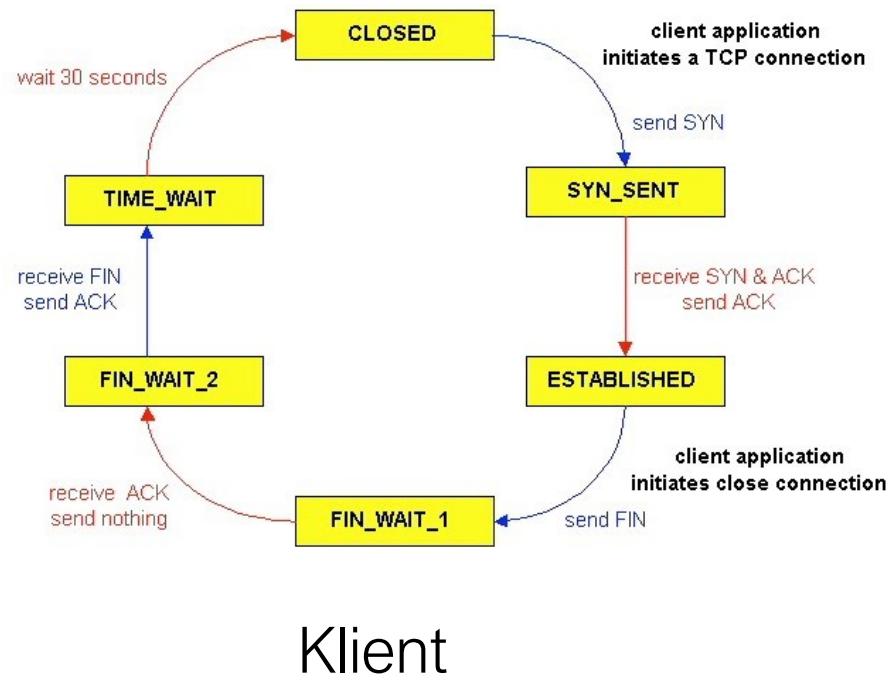
- FIN příznak nastaven -> následné potvrzení
- Aktivní a Pasivní uzavření spojení
- Poloviční uzavření spojení (half close)
  - data posílána pouze jedním směrem



# Zjednodušený FSM od TCP

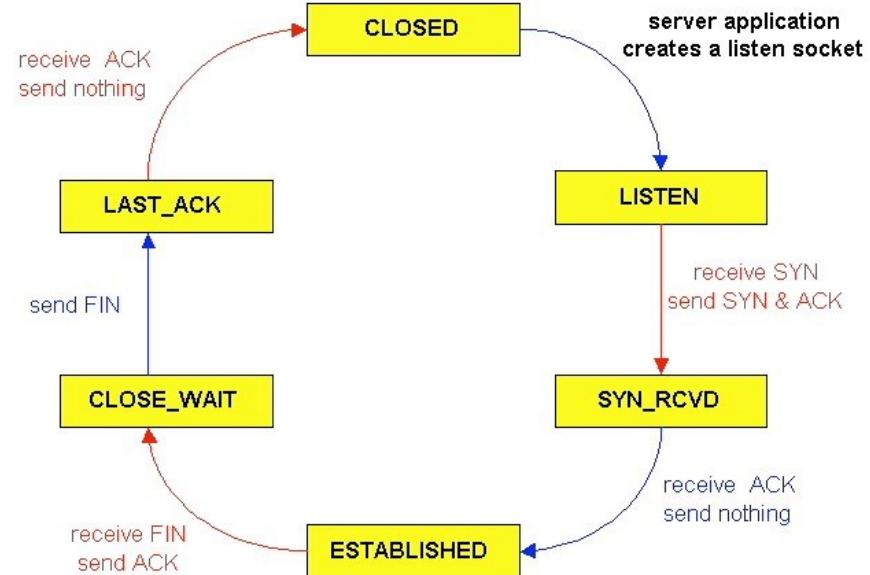


# Überzjednodušený FSM od TCP



Klient

Server



# 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 (rexmit timer) znovuzasílání
- K znovuzasílání (retransmission) dojde:
  - když vyprší časovač
  - když se obdrží duplicitní ACK

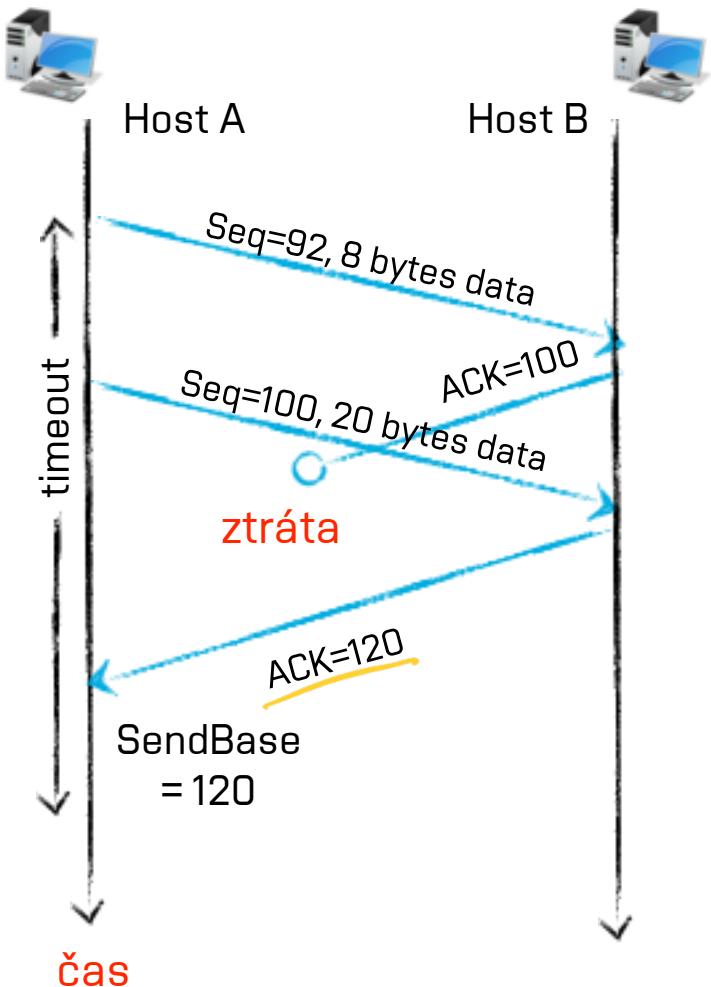
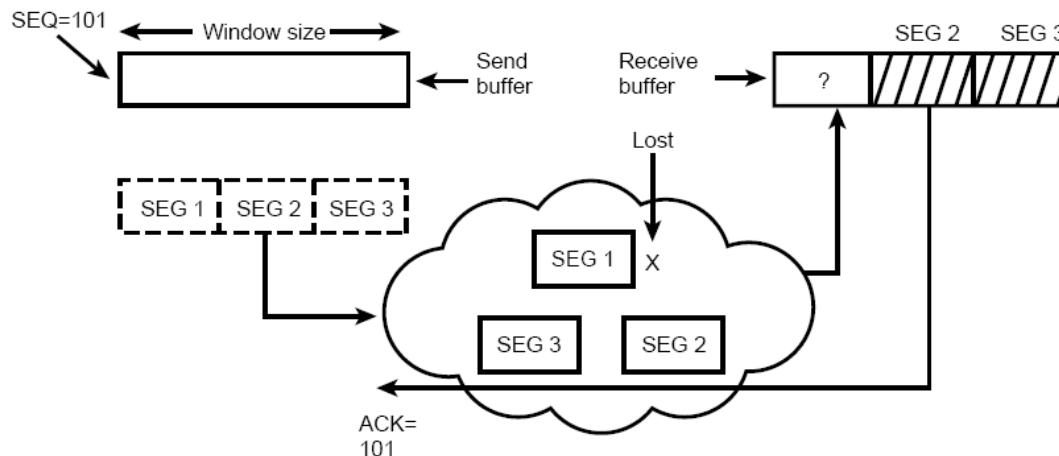
```
NextSeqNum = InitialSeqNum
SendBase = InitialSeqNum
loop (forever) {
    switch(event):
        event: data received from application above
            create TCP segment with
            sequence numberNextSeqNum
            if (timer currently not running)
                start timer
                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 znova zasílat
- Co s out-of-order daty?
  - TCP neřeší, závisí na konkrétní implementaci

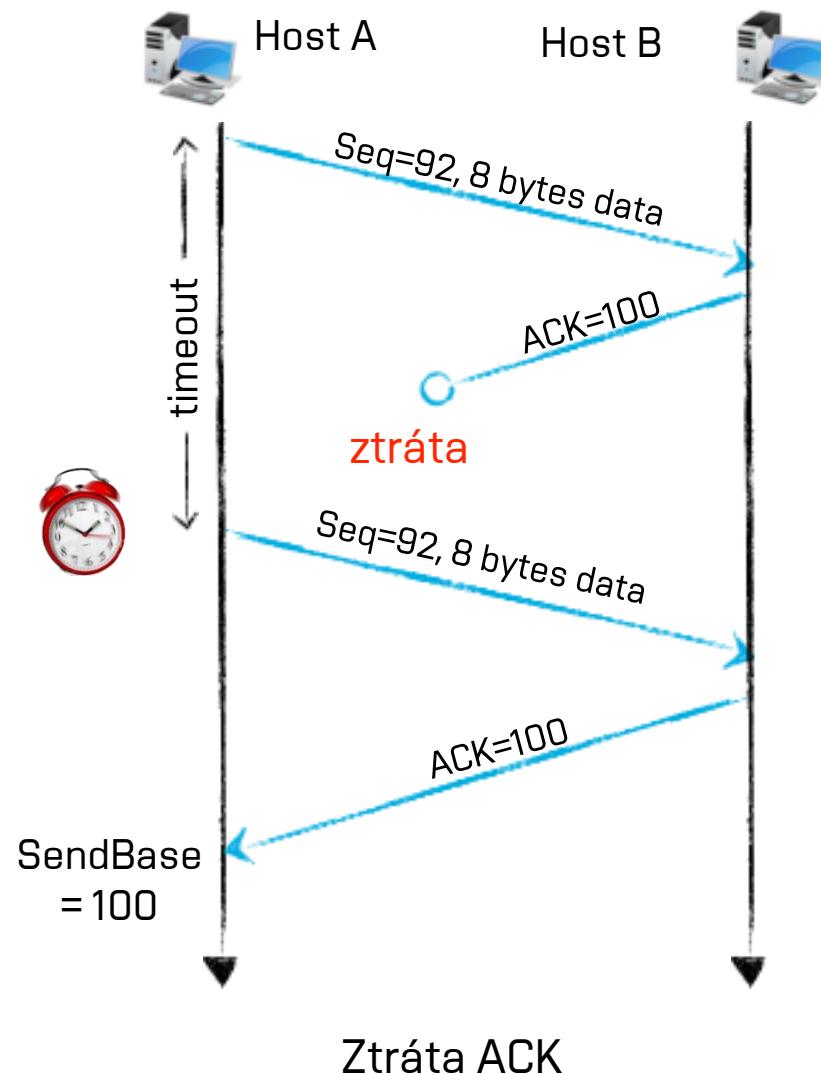


Kumulativní ACK

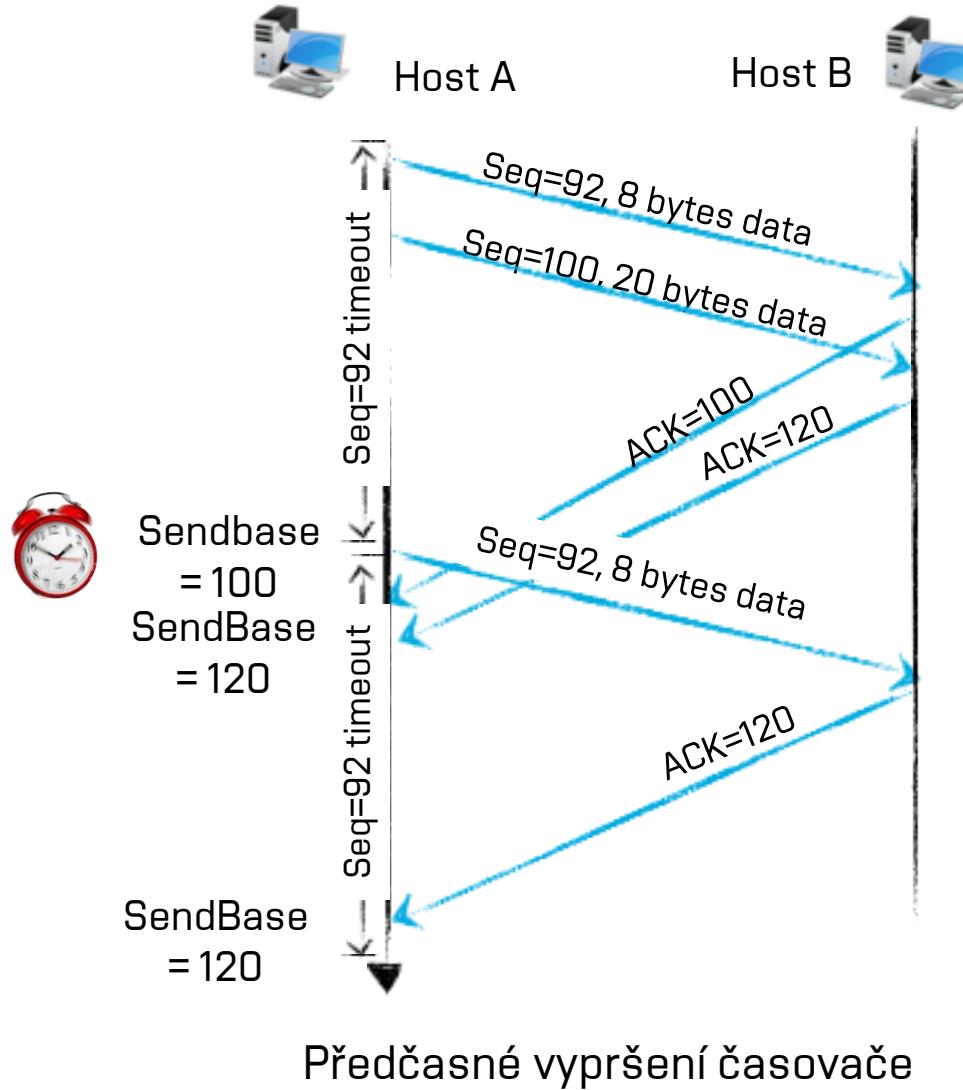
# Znovuzasílání

- *Musí pořešít 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

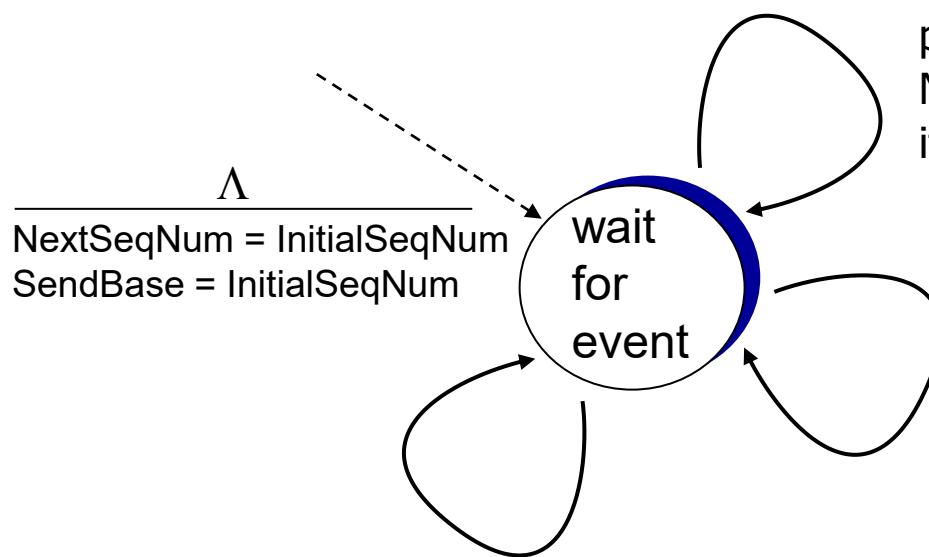
# TCP Retransmission (ztráta ACK)



# TCP Retransmission (zpožděné ACK)



# FSM odesilatele



**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
}
```

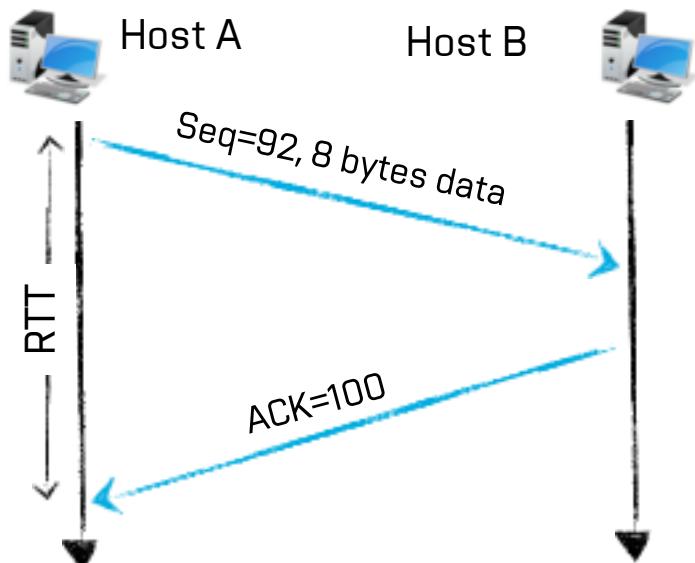
# FSM příjemce (jak s ACK)

- 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 mezery.	Okamžitě odešli ACK, který označuje další očekávaný oktet mezery.

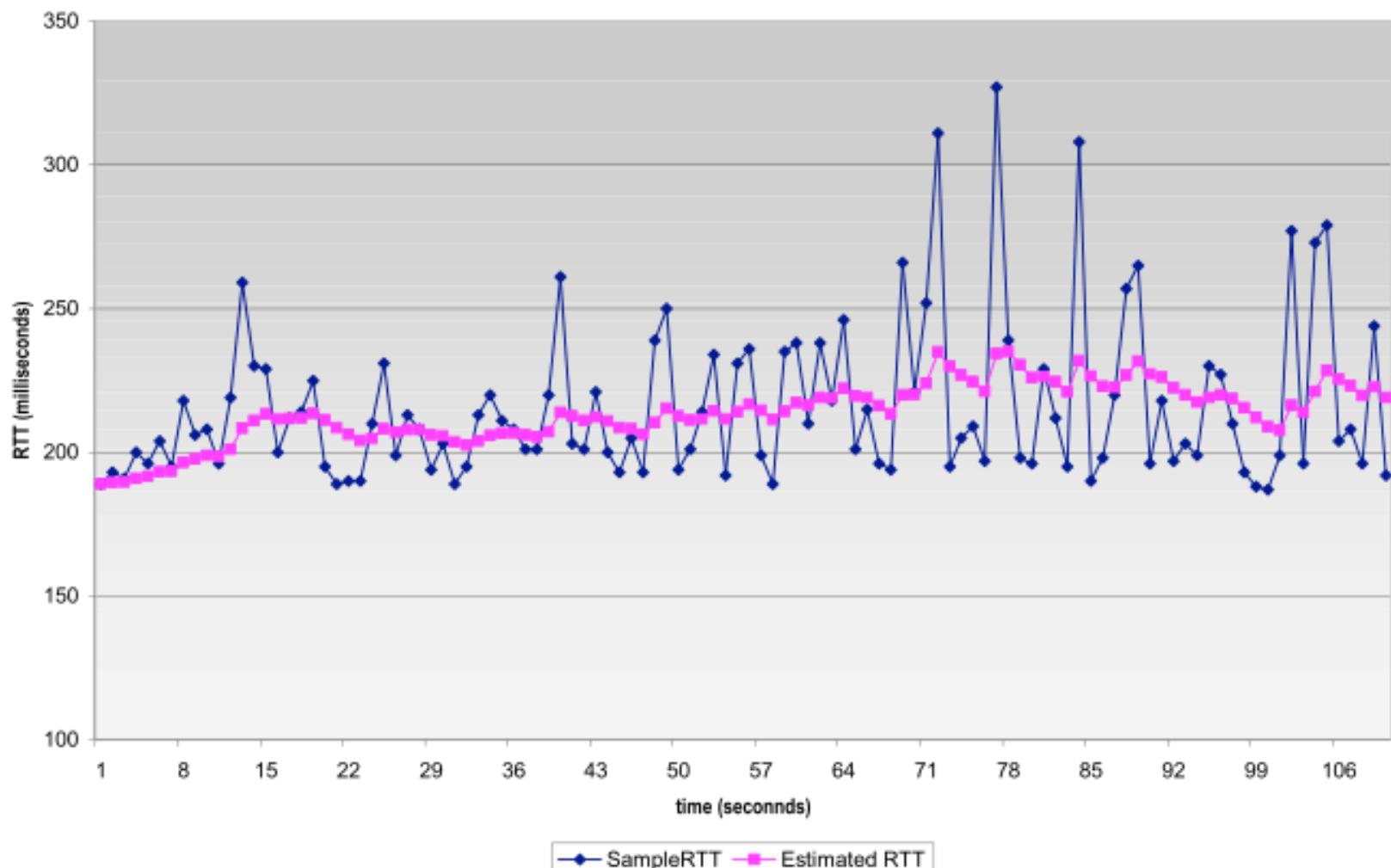
# Vypršení timeoutu

- 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
- $RTT$  je často relativně velký
  - což vede ke zbytečně dlouhé čekání na znovuvyslání ztraceného paketu
- Ztracené segmenty lze poznat pomocí duplikovaných ACKs
  - Odesilatel 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
  - = znova pošli segment než vyprší  $RTT$  na druhé straně

# Fast Retransmit

## Příklad

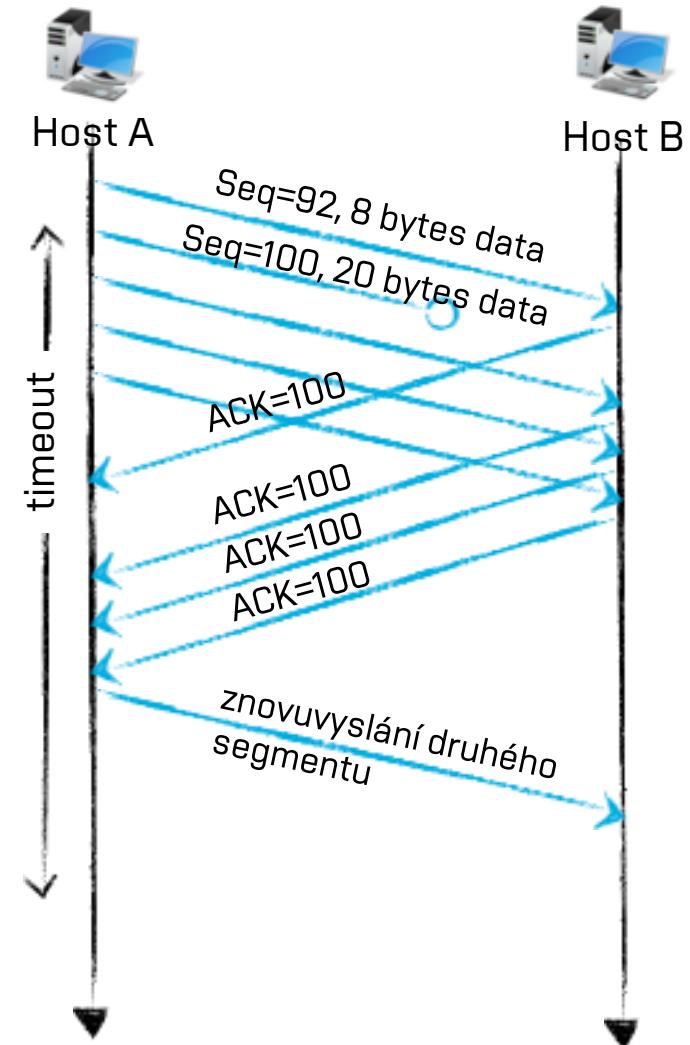
event: ACK received, with ACK field value of  $y$

```
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

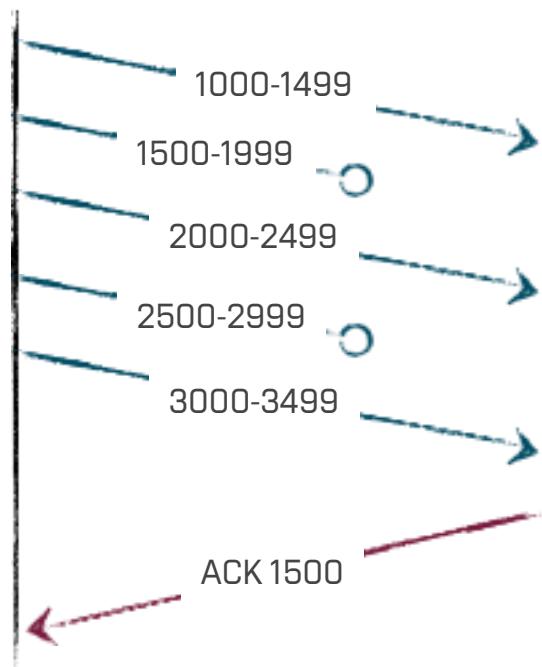
Fast retransmit



# 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

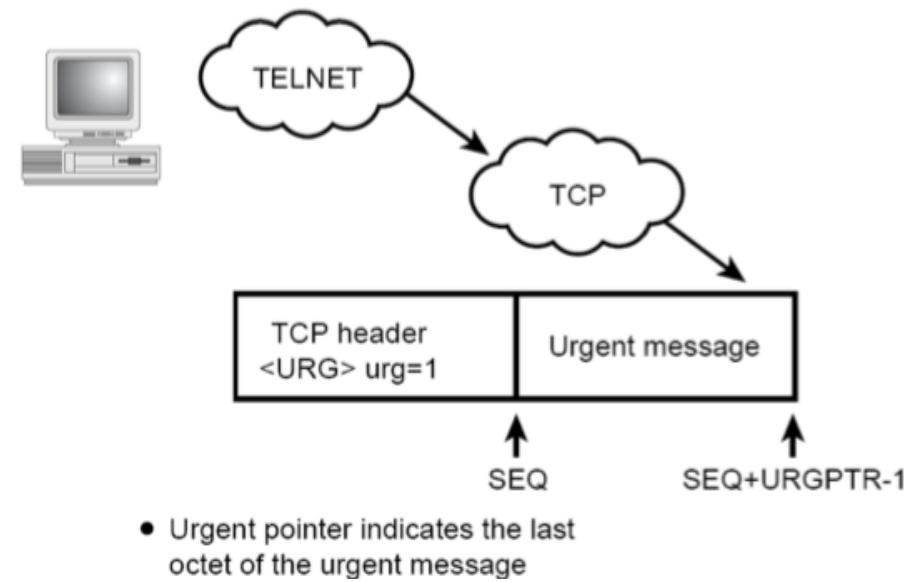


Kind=5	Length
Left Edge of 1st Block	
Right Edge of 1st Block	
.	
Left Edge of nth Block	
Right Edge of nth Block	

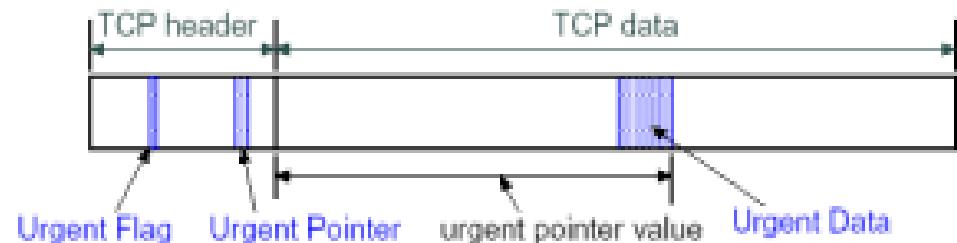
LeftEdge	RightEdge
2000	2500
3000	3500

# 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 urychlit o tom informovat druhou stranu.
  - Interaktivní aplikace využívají této vlastnosti: rlogin, telnet, ...

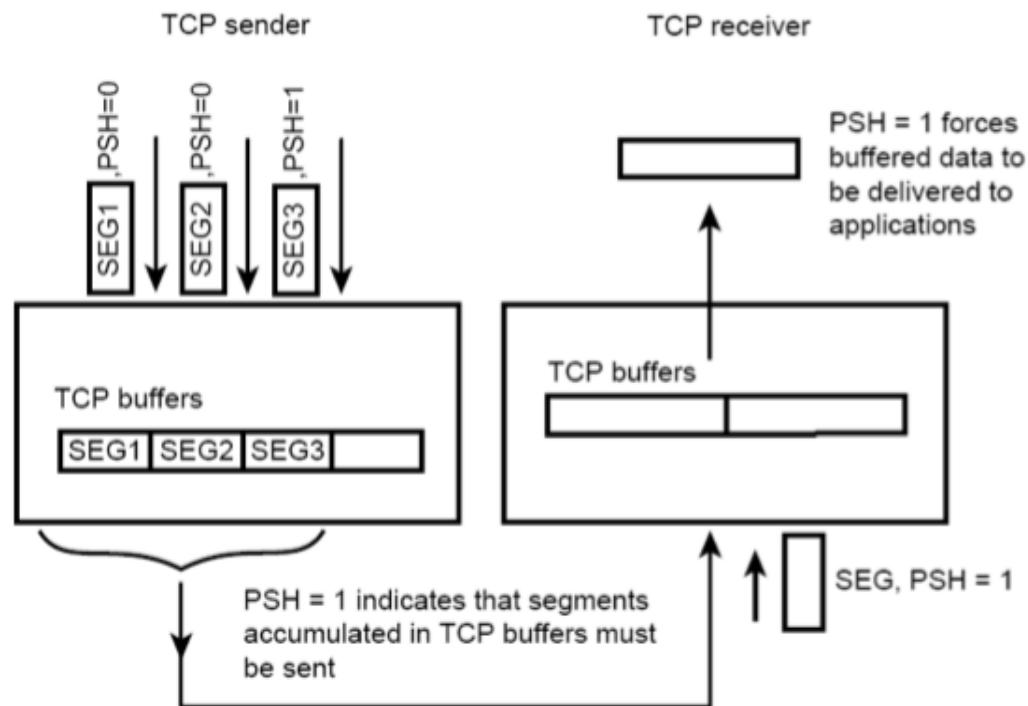


TCP packet with urgent data



# 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 vyrovnavací pamětí
  - Pokud směrovač nemůže vyslat paket v danou dobou, uloží jej do vyrovnavací 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 zahlcení

## End-end



*Tímto se budeme zabývat (Internet)*

- síťová vrstva neposkytuje žádné informace a podporu pro řízení zahlcení
- koncové stanice sledují charakteristiku sítě a approximují 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

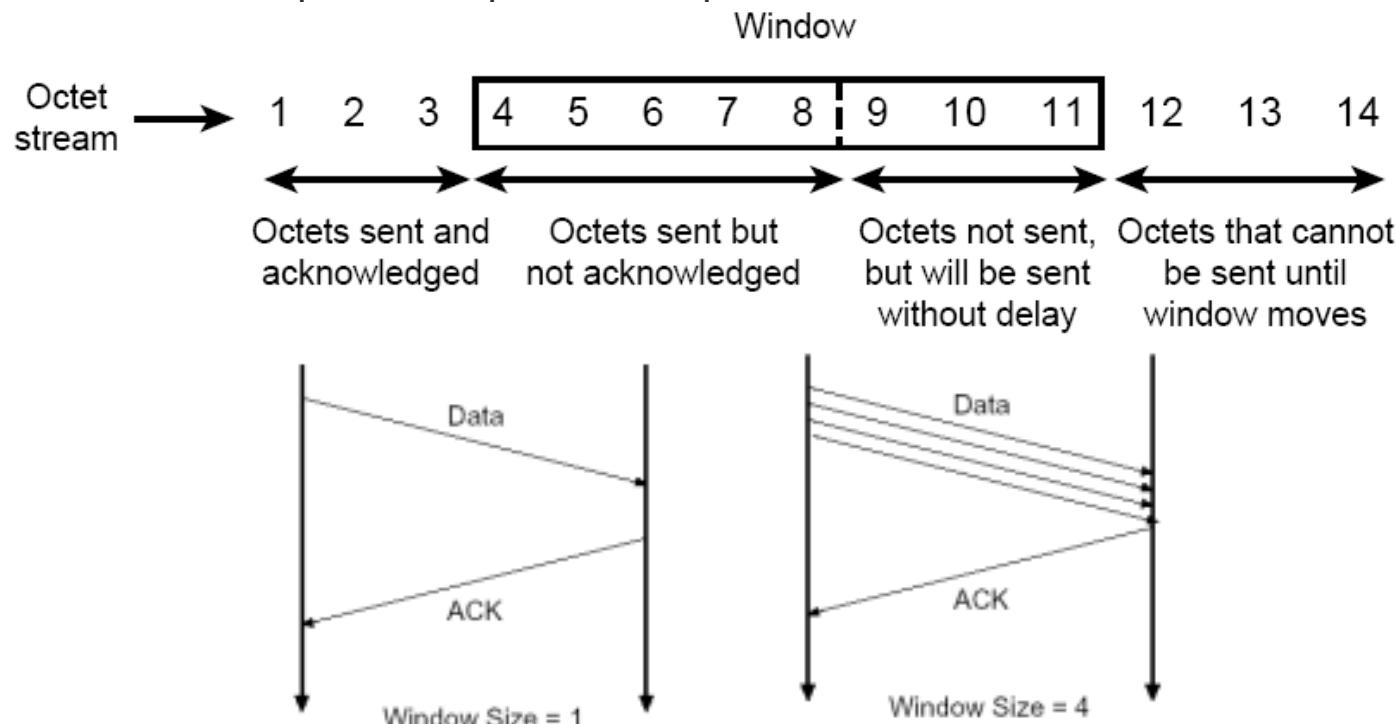
# Základní princip

- 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 (odesilatel zvyšuje rychlosť přenosu)
    - Jestliže se pakety ztrácejí ...
      - TCP předpokládá že síť je zahlcena (odesilatel snižuje rychlosť přenosu)
  - TCP zvyšuje rychlosť 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 → odesilatel by měl přestat vysílat
- Cílem mechanismu je maximálně využít přenosové linky  
a minimalizovat zpoždění způsobená potvrzováním

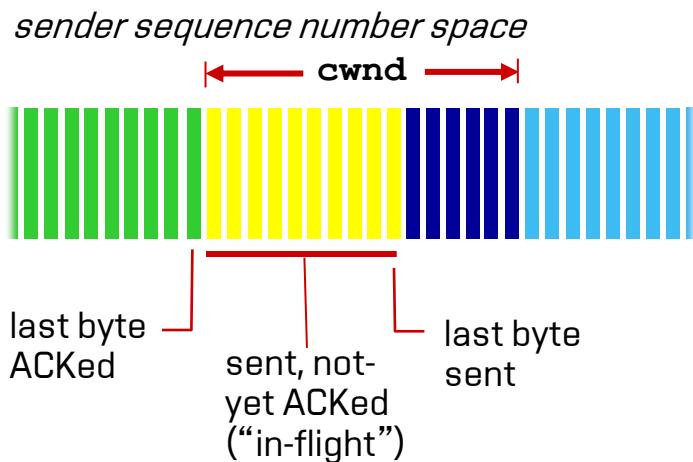


# Detaily

Odesilatel limitován dle:

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

cwnd je dynamické, hodnota funkce předvídá zahlcení sítě



TCP rychlosť odesílaní:

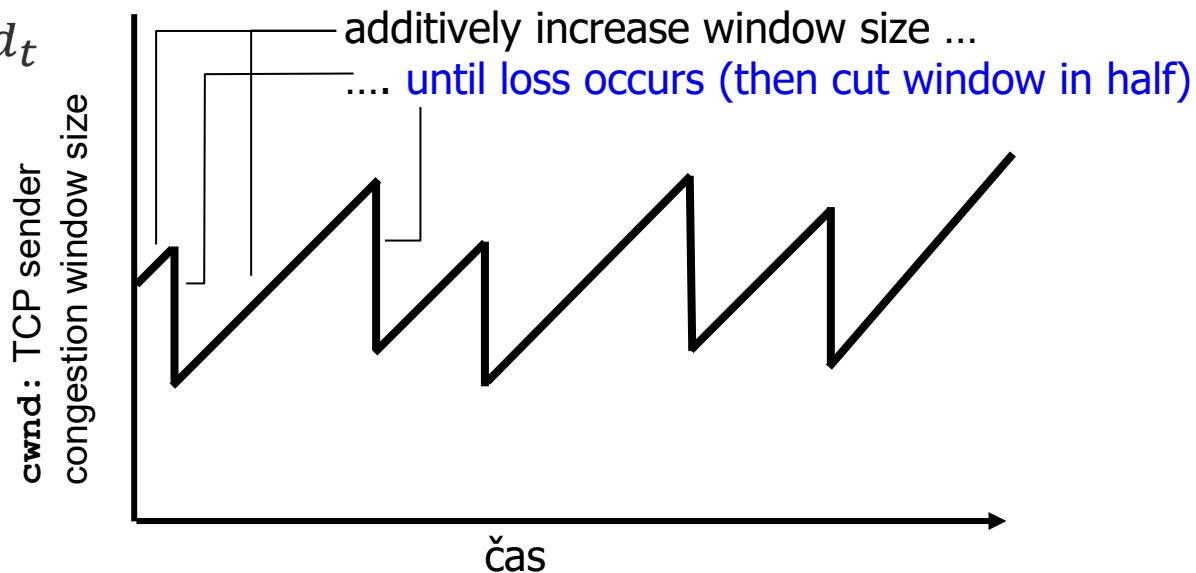
- 1) odešly  $cwnd$  B
- 2) počkej  $RTT$  na ACKs
- 3) pak pošli více B

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

- Additive Increase, Multiplicative Decrease

- RFC 2581
- Odesílatel zvedá rychlosť odesílaní lineárne
  - $cwnd_{t+1} = cwnd_t + 1$
  - V prípade ztráty sníží rychlosť na polovinu

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

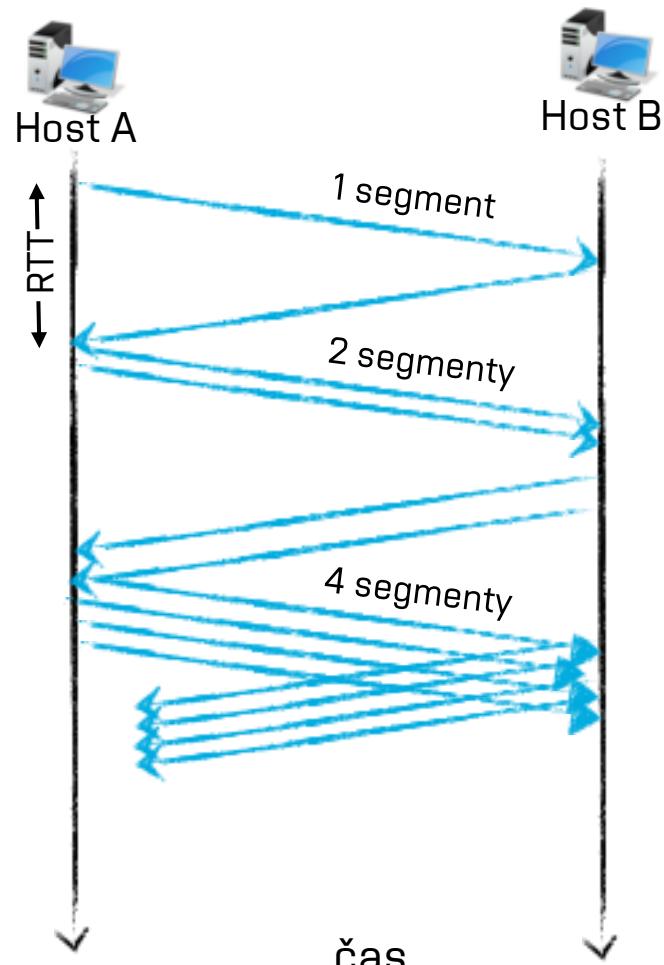


# Dějepisné okénko

- 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 přenosou rychlosť: 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

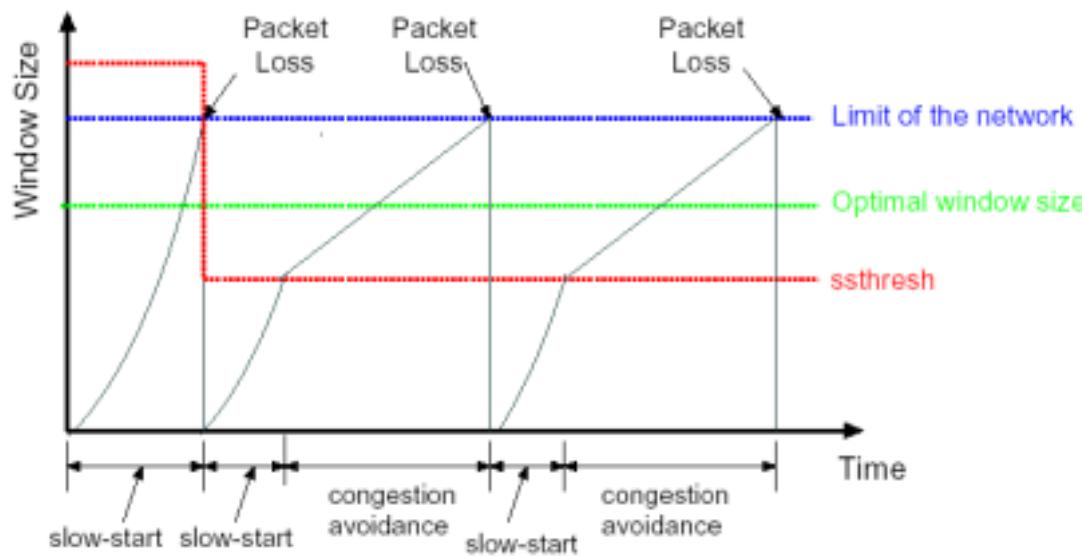


# Historie řízení zahlcení v TCP

- Tři hlavní verze řízení zahlcení TCP – později už jen dílčí úpravy
  - Řízení zahlcení 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

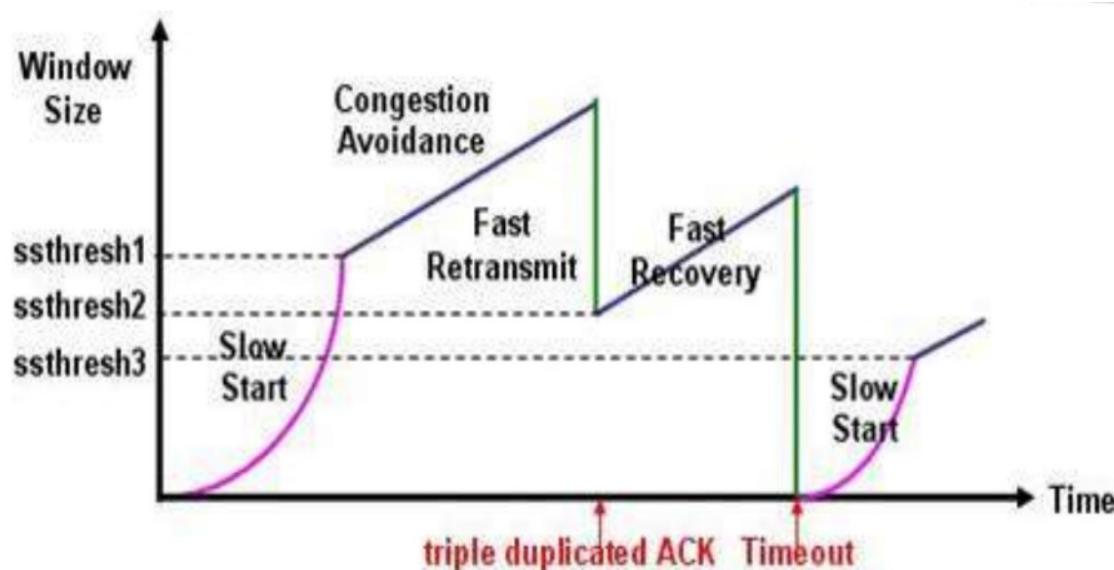
# Tahoe (SS+CA)

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

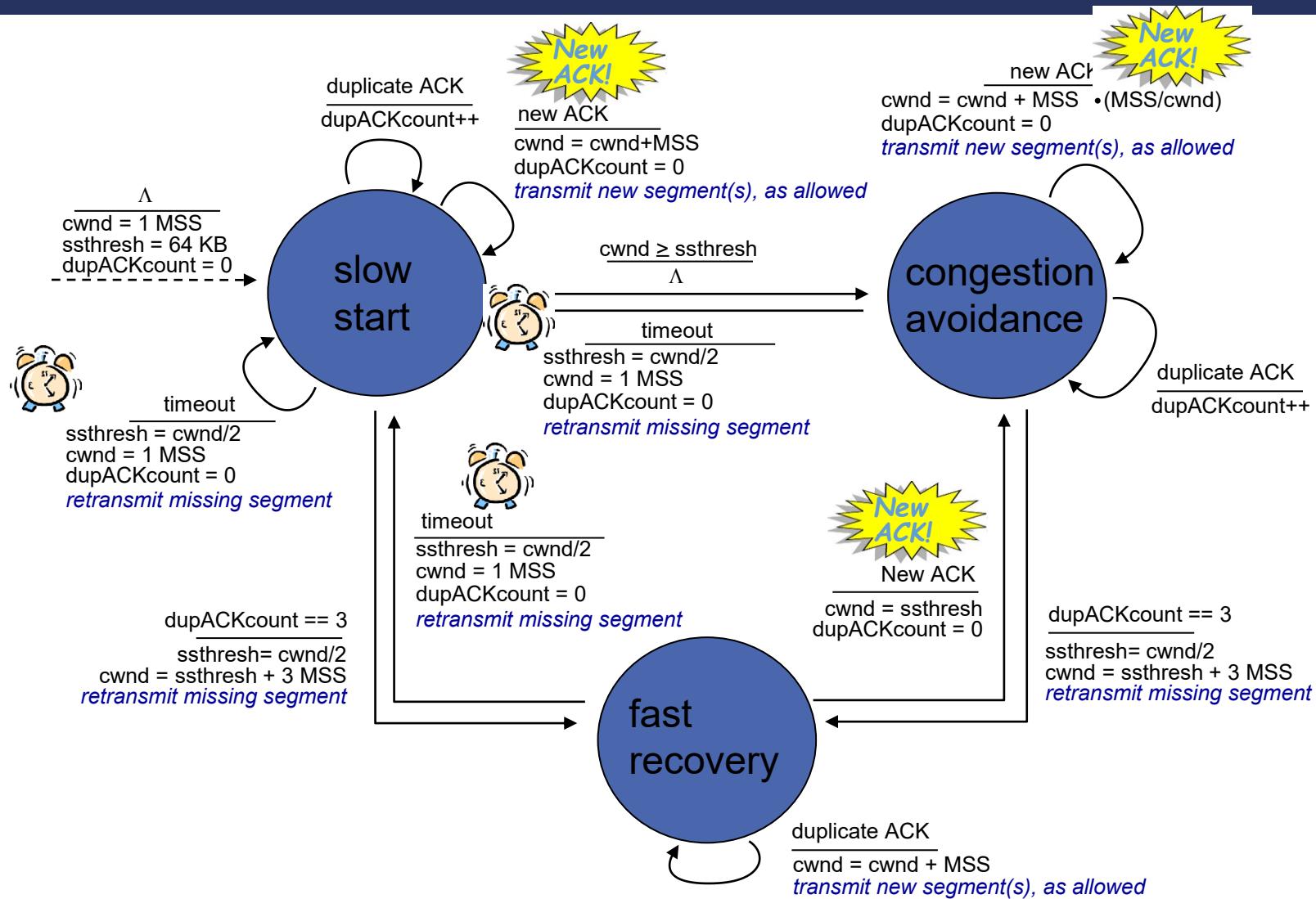


# Reno (SS+CA+FR)

- 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

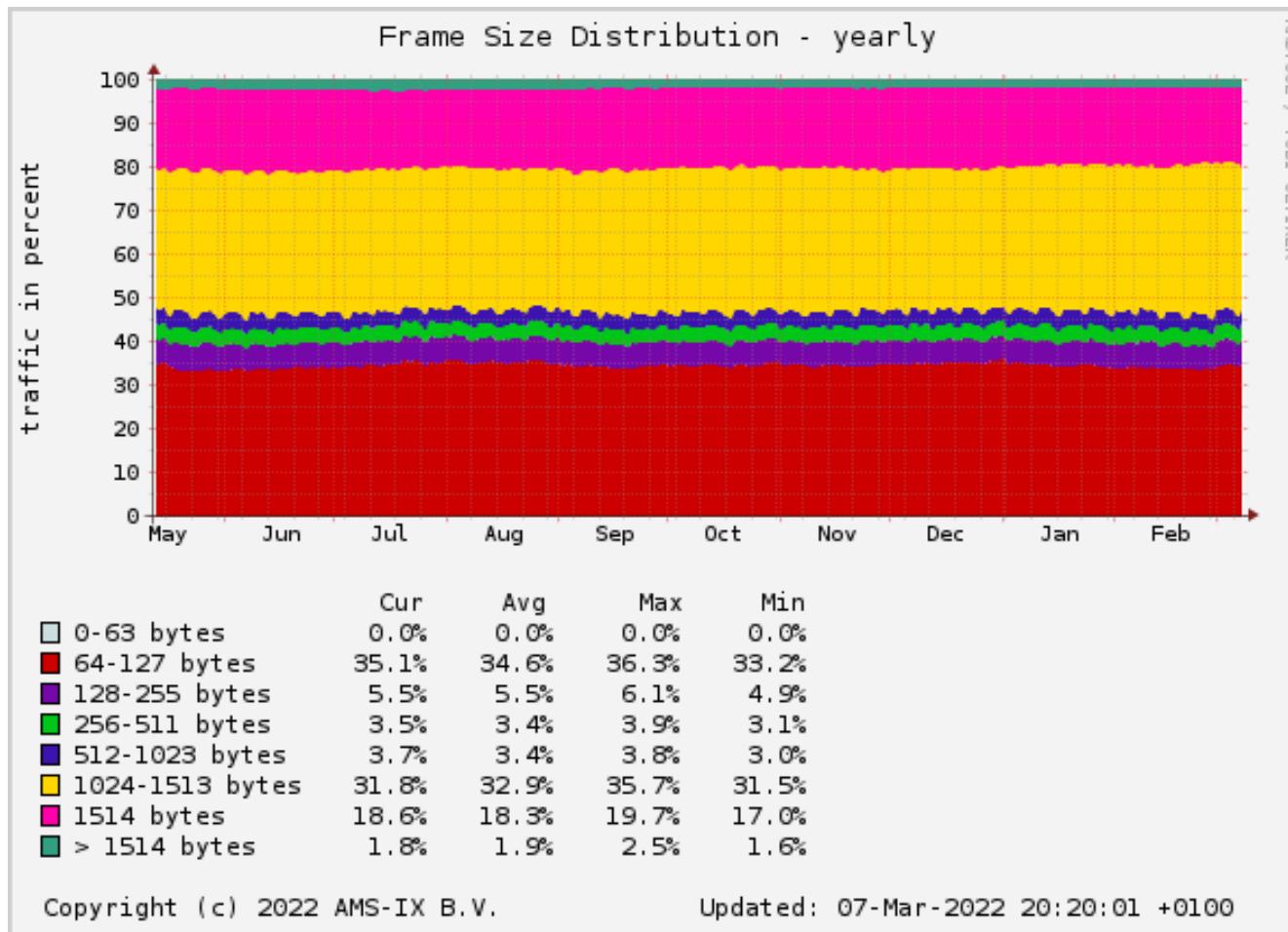


# Reno FSM



# 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