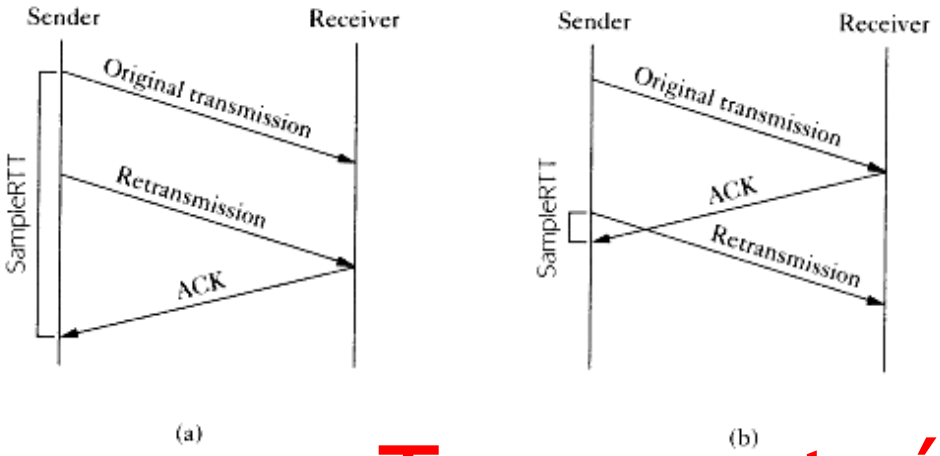
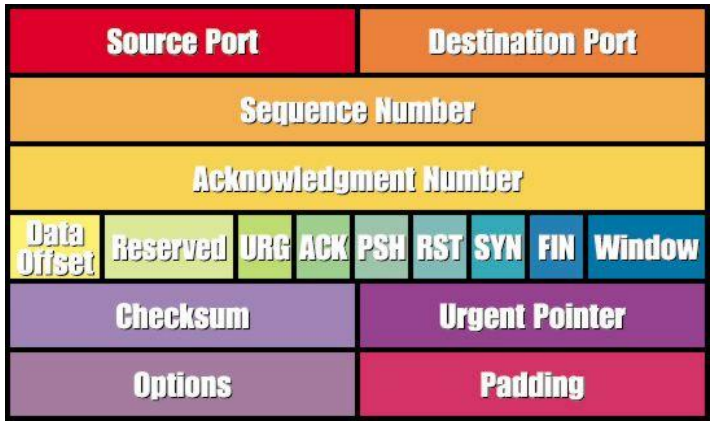
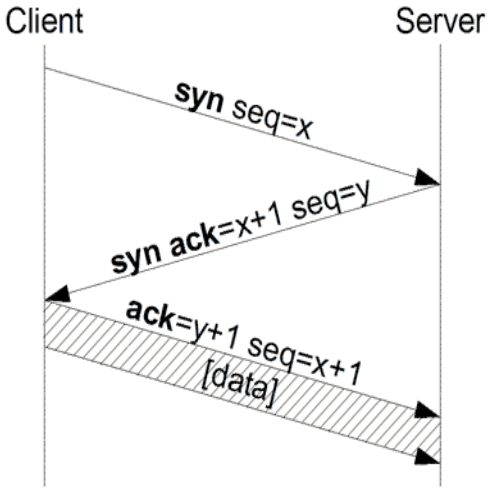
4. prednáška





Transportná vrstva



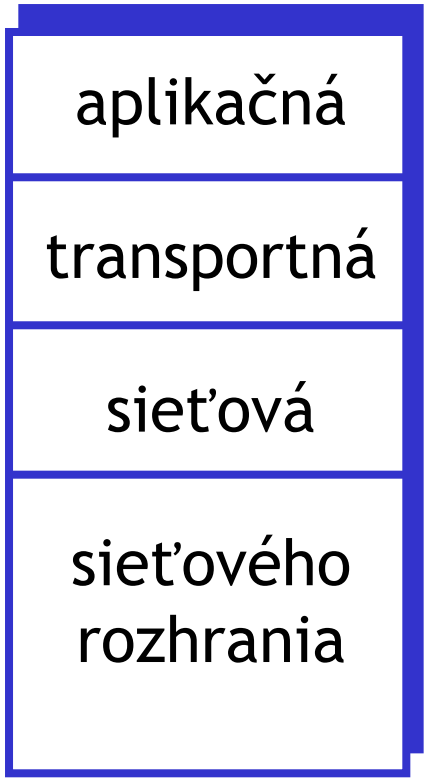
Source Port (16 bits)	Destination Port (16 bits)
Length (16 bits)	Checksum (16 bits)
Data....	

Osnova rozprávania o transportnej vrstve

- 3.1 Služby transportnej vrstvy
- 3.2 Delenie správ a adresácia soketov
- 3.3 UDP: bezstavový transportný protokol
- 3.4 Princípy potvrdzovaného toku dát
- 3.5 TCP: stavový transportný protokol
  - pripájanie a odpájanie
  - štruktúra segmentu
  - potvrdzovaný tok dát
  - kontrola toku dát
- 3.6 Princípy zabezpečenia kontroly zahltenia
- 3.7 Kontrola zahltenia v protokole TCP

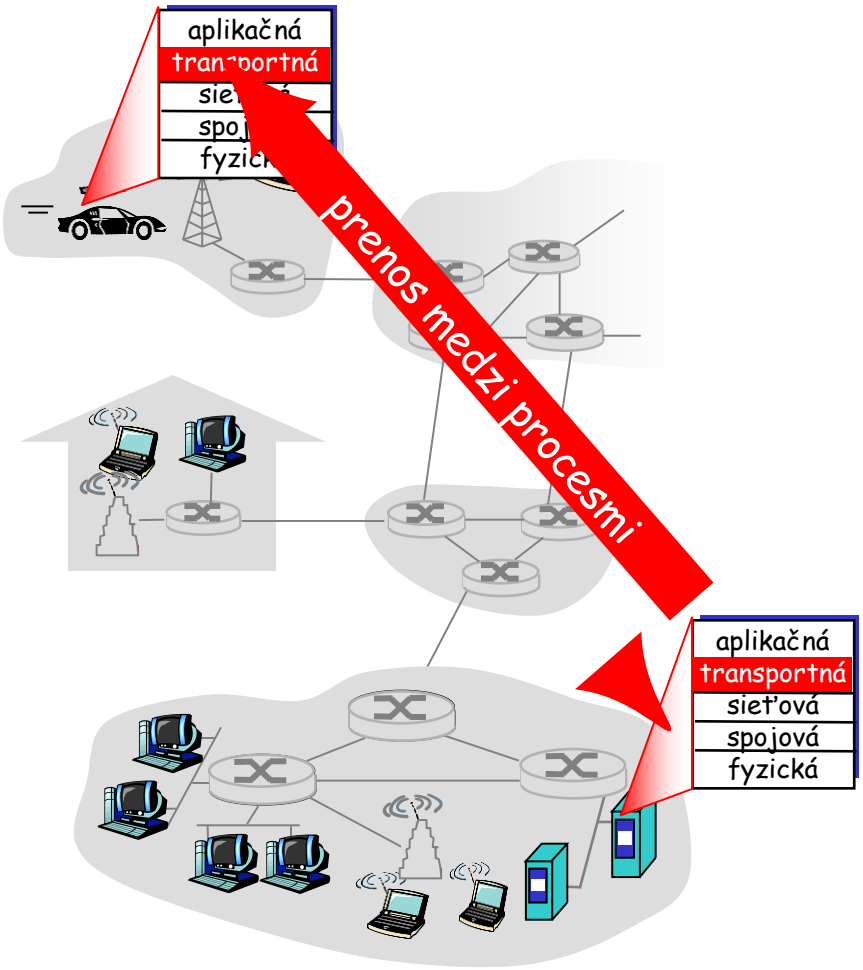
Implementácia internetu TCP/IP

- aplikačná (application): umožňuje fungovanie sieťových aplikácií – definuje tvar a poradie správ
  - HTTP, FTP, SMTP, POP, IMAP, XMPP, SSH, ...
- transportná (transport): prenáša dáta medzi dvoma procesmi na rôznych koncových zariadeniach
  - TCP, UDP
- sieťová (network): smeruje pakety od odosielateľa k príjemcovi
  - IP, smerovacie protokoly
- sieťového rozhrania (network interface): splynutie funkcionality do technológií na prenos dát medzi susednými sieťovými prvkami a spôsobu prenášania binárnych dát
  - PPP, Ethernet



Protokoly a služby transportnej vrstvy

- poskytujú logický prenos dát medzi procesmi programov bežiacich na vzdialených koncových zariadeniach
- transportné protokoly sa realizujú iba na koncových staniciach
- TCP:
  - **odosielajúca strana:** rozdeľuje správy odosielané z procesu a vkladá ich do segmentov a tie posiela na spracovanie sieťovej vrstve
  - **prijímajúca strana:** spája dáta zo segmentov prijaté zo sieťovej vrstvy do správ a posiela ich procesu v aplikačnej vrstve
- UDP:
  - Delenie správ je na aplikačnej vrstve



Transportná verzus sieťová vrstva

- *transportná vrstva:* umožňuje logickú komunikáciu medzi procesmi
  - spolieha sa na služby sieťovej vrstvy
- *sieťová vrstva:* umožňuje logickú komunikáciu medzi stanicami
  - transportná vrstva iba povie, ktorej cieľovej stanici je segment určený

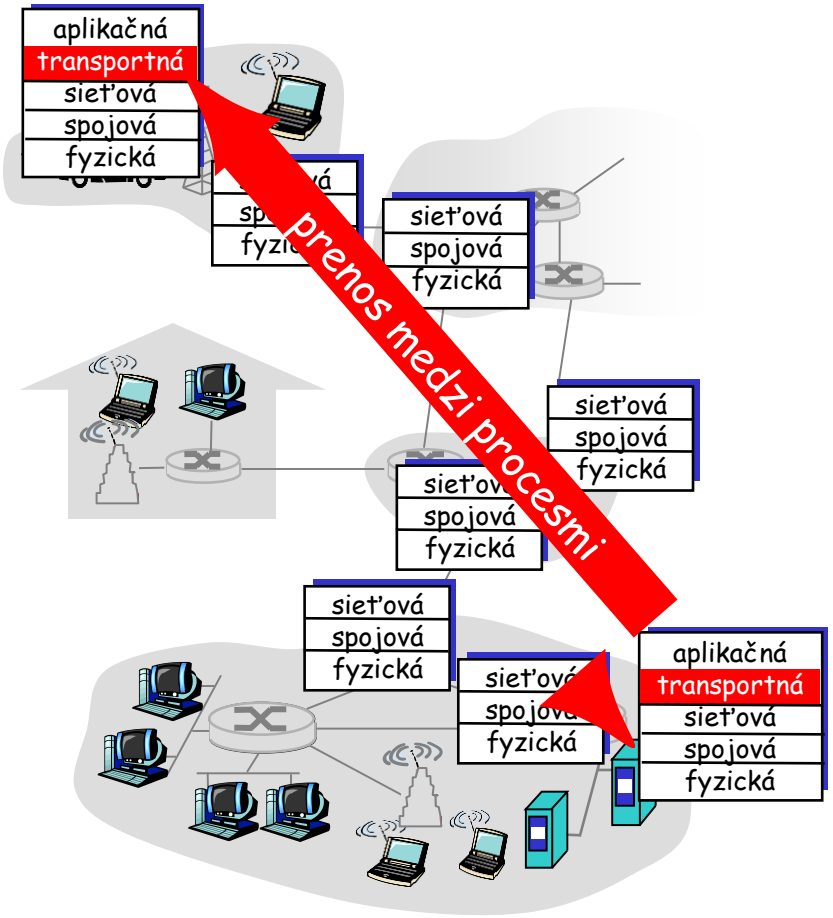
Analógia "Koncert Madonny"

Treba dopraviť pódium a techniku z Londýna do Košíc

- procesy = londýnsky a košický manažér
- správy aplikačnej vrstvy = pódium a technika
- koncové stanice = mestá
- transportný protokol = tímy technikov v koncových mestách a komunikácia medzi nimi
- protokol sieťovej vrstvy= zásielková služba (64 kamiónov, 2 lode a logistika)

Internetové protokoly transportnej vrstvy

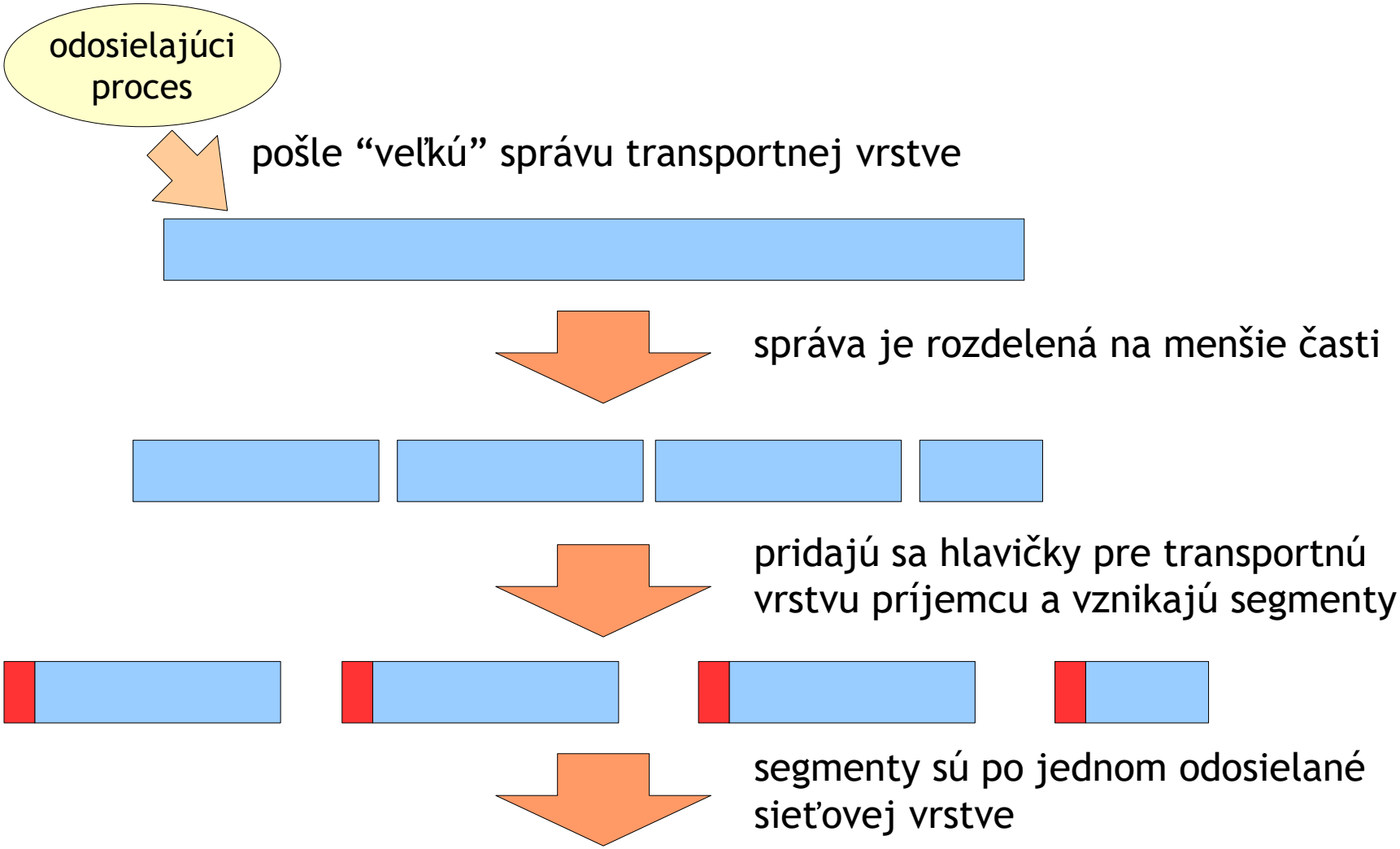
- potvrdzované doručenie v správnom poradí: TCP
  - potvrdzovanie segmentov
  - kontrola zahltenia
  - kontrola toku dát
  - nastavovanie spojenia
- nepotvrdzované doručenie v "náhodnom" poradí: UDP
  - žiadne extra vylepšenie oproti spoľahlivosti IP
- neposkytované služby:
  - zabezpečenie času doručenia
  - zabezpečenie prenosovej rýchlosti



Osnova rozprávania o transportnej vrstve

- 3.1 Služby transportnej vrstvy
- 3.2 Delenie správ a adresácia soketov
- 3.3 UDP: bezstavový transportný protokol
- 3.4 Princípy potvrdzovaného toku dát
- 3.5 TCP: stavový transportný protokol
  - pripájanie a odpájanie
  - štruktúra segmentu
  - potvrdzovaný tok dát
  - kontrola toku dát
- 3.6 Princípy zabezpečenia kontroly zahltenia
- 3.7 Kontrola zahltenia v protokole TCP

Delenie správ do segmentov (TCP)



Delenie a spájanie správ

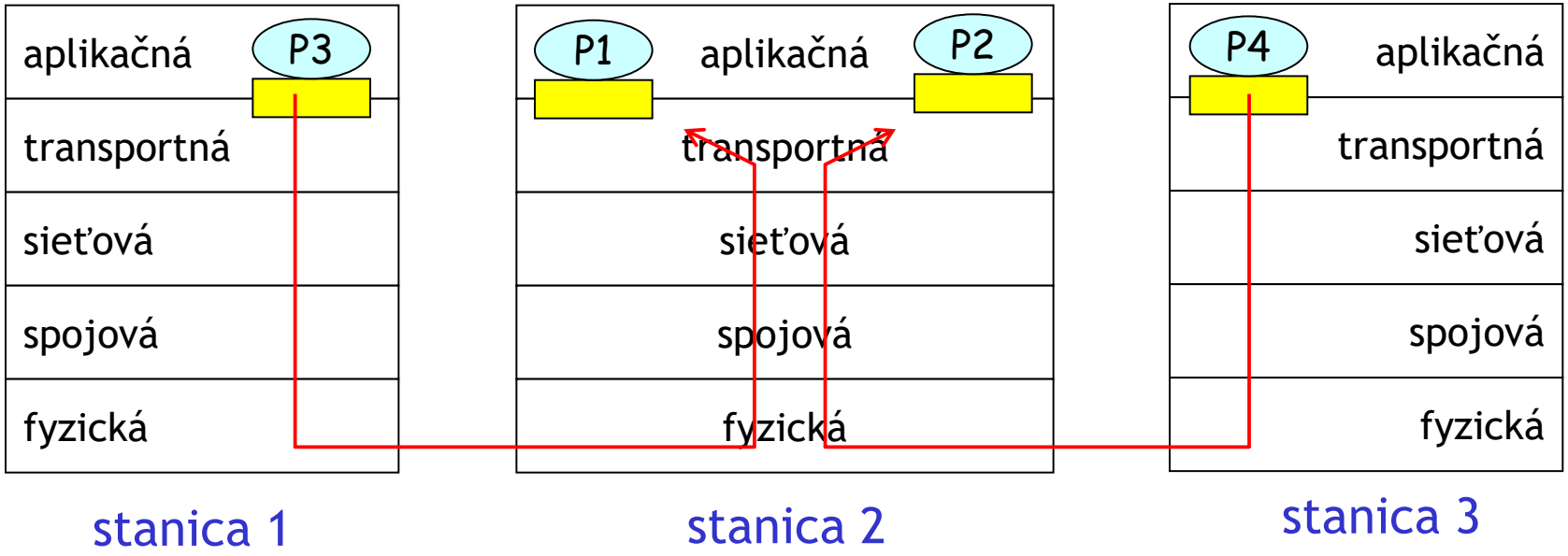
U odosielateľa:

zozbieranie správ zo soketov, vytvorenie segmentov a odoslanie príjemcovi

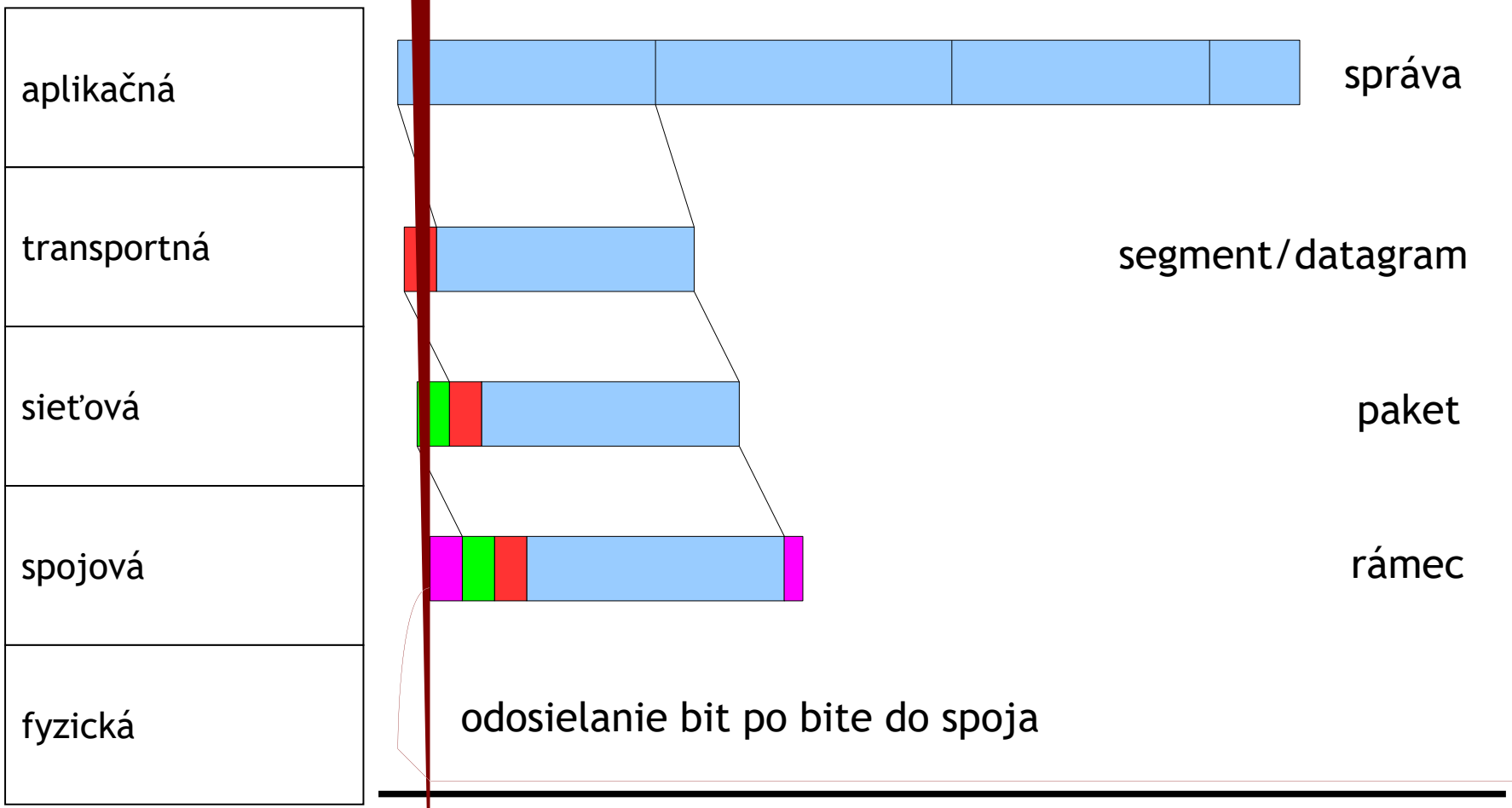
U príjemcu:

vyskladanie správ z prijatých segmentov a ich doručenie k správnym soketom

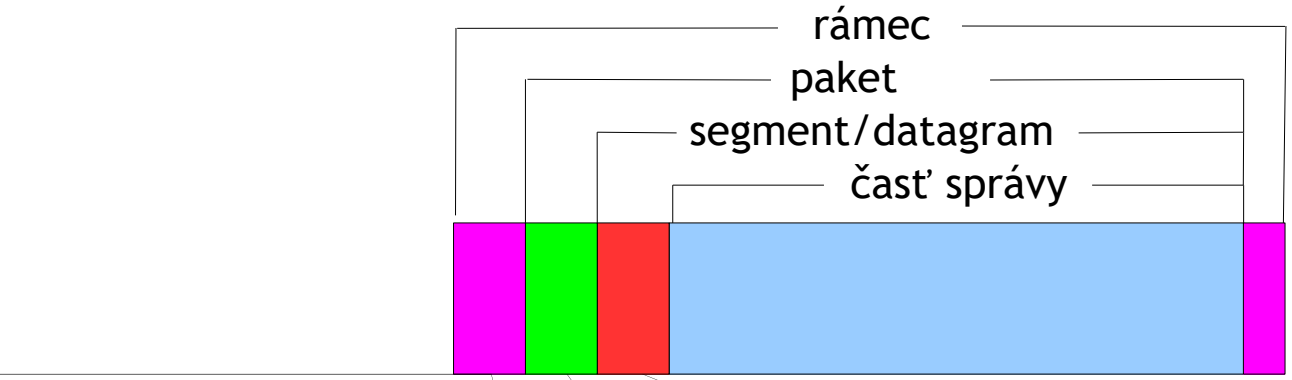
= soket = proces



Tvorba paketu



Adresovacie údaje v pakete



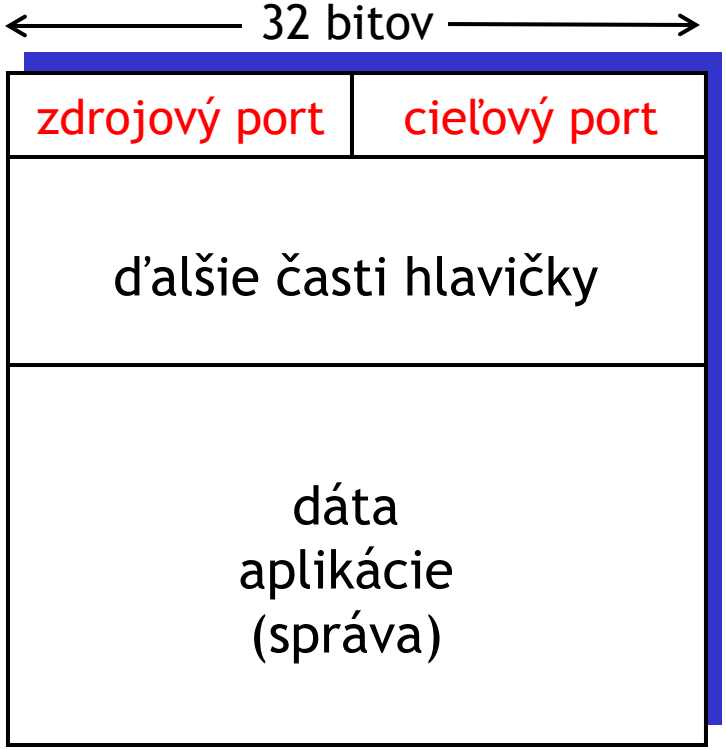
port odosielateľa a
port príjemcu

IP adresa odosielateľa a
IP adresa príjemcu

MAC adresa odosielateľa a
MAC adresa príjemcu

Odosielanie

- stanica dostane IP paket
  - hlavička IP paketu má zdrojovú IP adresu a cieľovú IP adresu
  - sieťová vrstva odovzdá z paketu vyextrahovaný segment/datagram transportnej vrstve
  - hlavička segmentu/datagramu má zdrojový port a cieľový port
- Transportná vrstva používa IP adresy a čísla portov na nasmerovanie segmentu/datagramu k príslušnému soketu



formát segmentu TCP/ datagramu UDP

Adresácia soketov v UDP

- nový soket sa asociuje s číslom portu a prípadne aj s niektorou konkrétnou IP adresou stanice:

```
soket1 = new
  DatagramSocket(12534);
soket2 = new
  DatagramSocket(12535,
              "158.197.31.4");
```

- UDP soket je jednoznačne identifikovaný dvojicou:

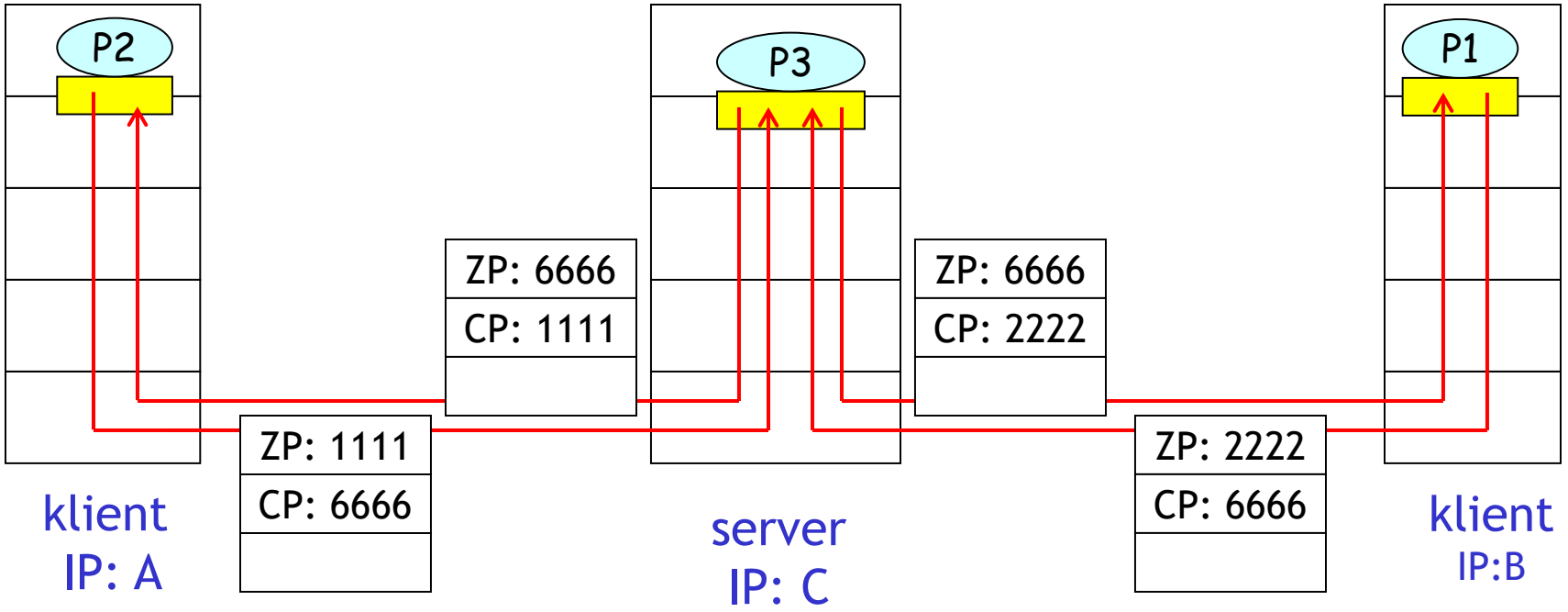
  (cieľová IP adresa, cieľový port)

- Keď stanica prijme UDP datagram:
  - prečíta si číslo cieľového portu a cieľovú IP adresu
  - pošle UDP datagram soketu asociovanému s týmto číslom portu
- pakety s rôznymi zdrojovými IP adresami a/alebo zdrojovými číslami portov môžu byť nasmerované k rovnakému soketu

Adresácia soketov v UDP

= soket = proces

```
DatagramSocket serverSocket = new DatagramSocket(6666);
```



Zdrojový port poskytuje "návratovú adresu"

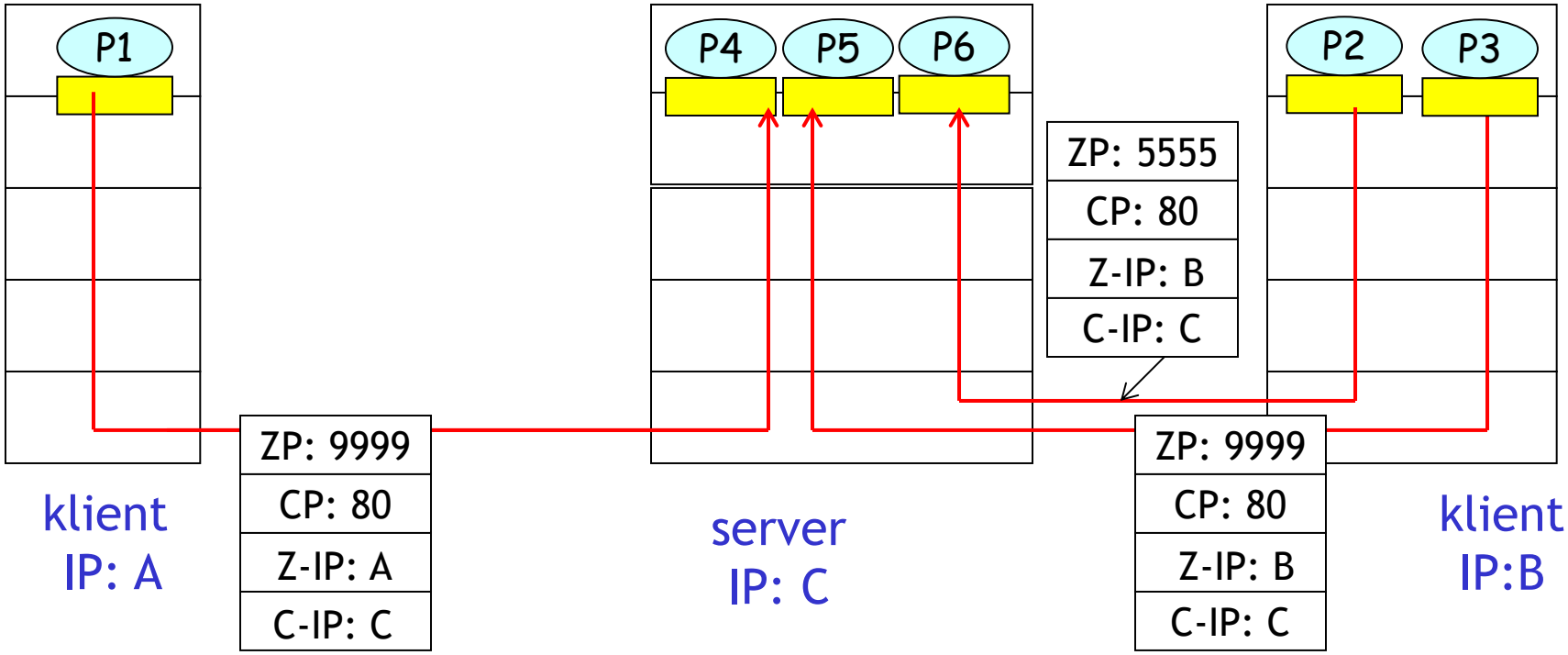
ZP = zdrojový port
CP = cieľový port

Adresácia soketov v TCP

- TCP soket je jednoznačne identifikovaný štvoricou:
  - zdrojová IP adresa
  - zdrojový port
  - cieľová IP adresa
  - cieľový port
- príjemca použije všetky štyri hodnoty na nasmerovanie k správnemu soketu

- proces môže mať súčasne otvorených veľa TCP soketov:
  - každý soket je identifikovaný svojou štvoricou
- napr. webové servery majú pre každého napojeného klienta samostatný soket
  - všetci komunikujú s rovnakým portom

Adresácia soketov v TCP



= soket = proces

ZP = zdrojový port
CP = cieľový port
Z-IP = zdrojová IP adresa
C-IP = cieľová IP adresa

Osnova rozprávania o transportnej vrstve

- 3.1 Služby transportnej vrstvy
- 3.2 Delenie správ a adresácia soketov
- 3.3 UDP: bezstavový transportný protokol
- 3.4 Princípy potvrdzovaného toku dát
- 3.5 TCP: stavový transportný protokol
  - pripájanie a odpájanie
  - štruktúra segmentu
  - potvrdzovaný tok dát
  - kontrola toku dát
- 3.6 Princípy zabezpečenia kontroly zahltenia
- 3.7 Kontrola zahltenia v protokole TCP

UDP: User Datagram Protocol [RFC 768]

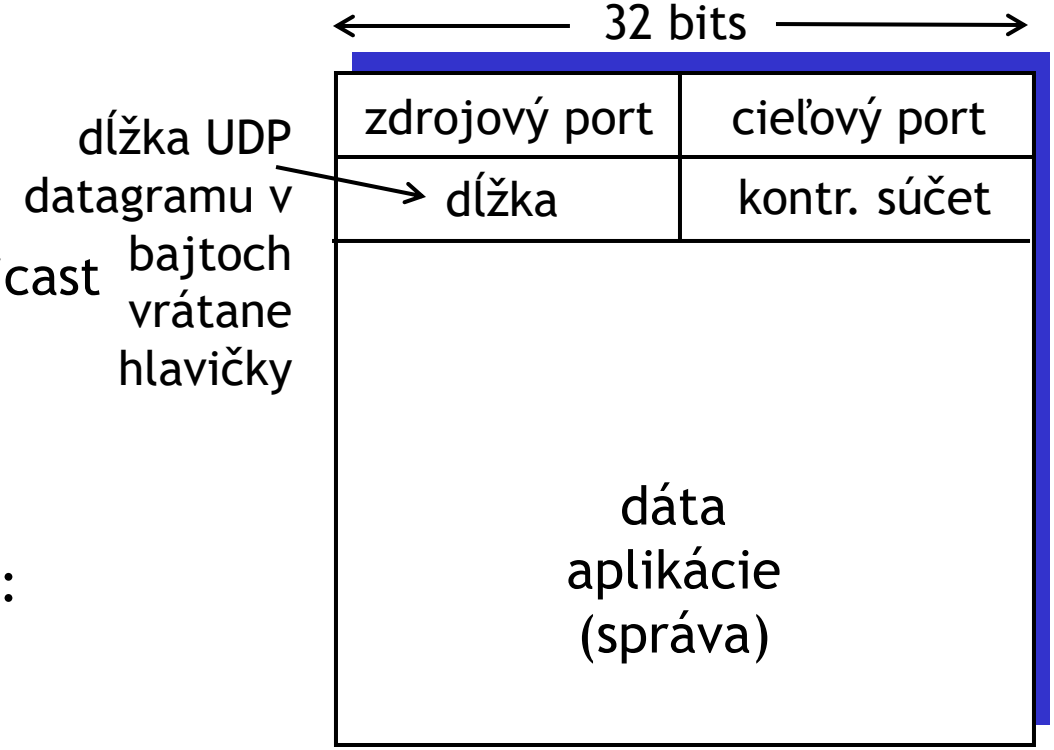
- UDP datagramy sa môžu:
  - stratiť
  - byť doručené v inom poradí, ako boli odoslané
- odosielanie „najväčším úsilím“ (best effort)
- *bezspojový protokol:*
  - žiadne nadväzovanie spojenia (handshaking) medzi odosielateľom a príjemcom
  - každý UDP datagram je obslúžený nezávisle na ostatných

Načo je dobré UDP?

- žiadne inicializácie spojenia, ktoré môžu spôsobiť oneskorenie
- jednoduchý: nie je potrebné žiadne uchovávanie stavu odosielania/prijímania
- real-time prenos
- žiadna kontrola zahltenia: UDP odosiela dáta hneď, keď ich dostane z aplikačnej vrstvy
- odosielanie datagramu viacerým cieľovým staniciam (broadcast, multicast)

UDP: použitie

- Streamovanie multimédií
  - tolerujú stratu častí dát
  - závislé na rýchlosti odosielania
  - viacerým cieľom cez multicast
- DNS
- SNMP – simple network management protocol
- spoľahlivý prenos cez UDP: spoľahlivosť je riešená v aplikačnej vrstve
  - špecifické zotavovanie z chýb pre danú aplikáciu



formát UDP datagramu

Kontrolný súčet UDP datagramu

Cieľ: odhalenie "chýb" (napr. zmenené bity) v dôjdenom datagrame

Odosielateľ:

- prechádza obsah datagramu ako množinu 16-bitových čísiel
- kontrolný súčet: sčítanie týchto čísiel a na záver vyrobenie inverzného čísla z výsledku sčítania (nuly vymení za jednotky a jednotky za nuly)
- odosielateľ pridá kontrolný súčet do UDP hlavičky

Príjemca:

- vypočíta súčet obsahu datagramu ako množiny 16-bitových čísiel
- nakoniec sčíta výsledok s kontrolným súčtom z UDP hlavičky.
  - Ak výsledkom nie je číslo so samými jednotkami, našli sme chybu a segment zahodíme
  - Ak áno, nenašli sme chybu
  - (ale chyba mohla aj tak nastať)

Príklad kontrolného súčtu

- Obsah datagramu tvoria červené čísla

Odosielateľ:

	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	1	0	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
	1	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
súčet	1	1	0	0	1	0	1	0	1	1	0	0	1	0	1	0
kontrolný súčet	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1

Príjemca – spraví rovnaký súčet obsahu datagramu:

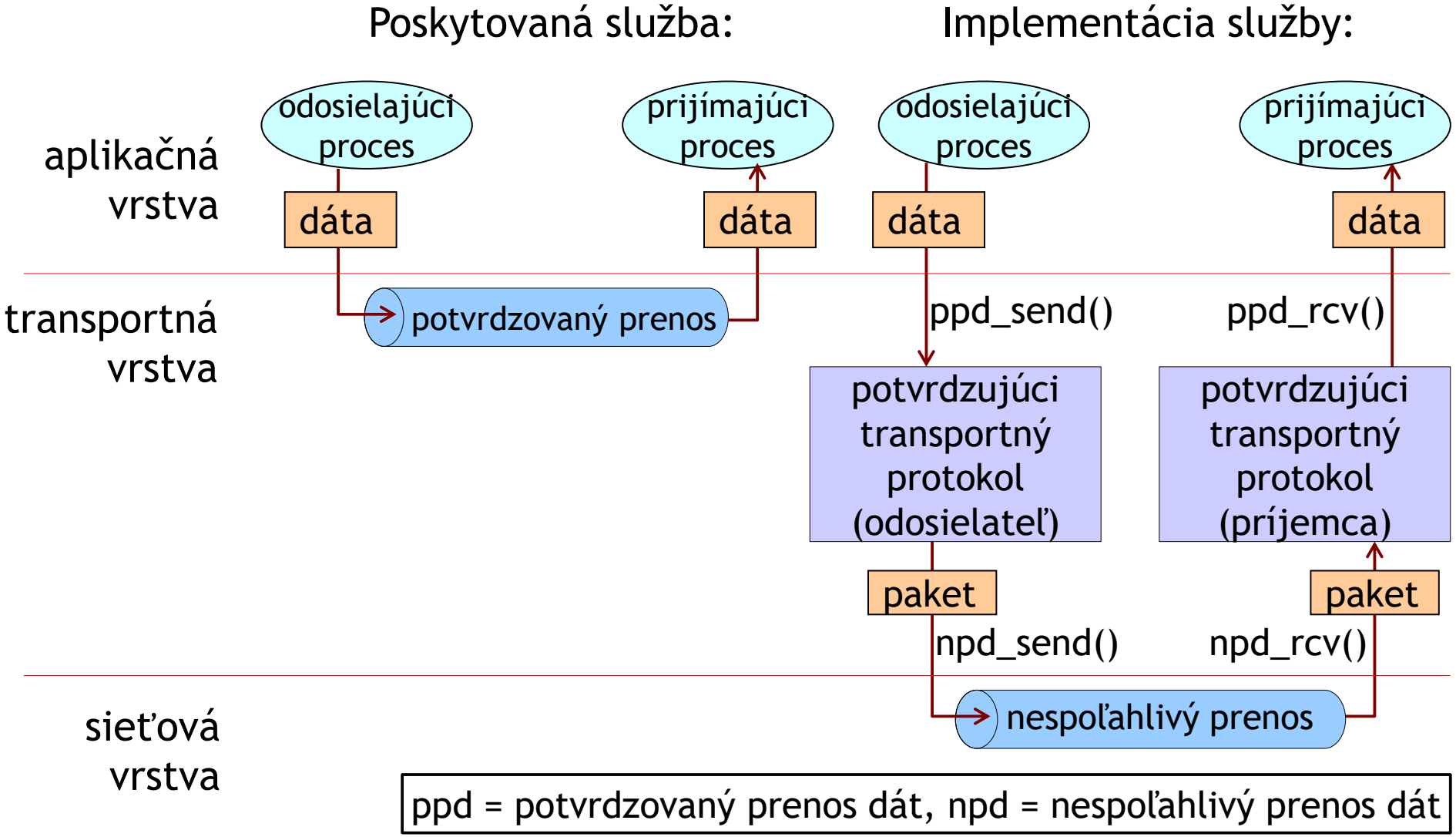
súčet	1	1	0	0	1	0	1	0	1	1	0	0	1	0	1	0
	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1
kontrolný súčet z hlavičky	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

samé 1 - OK

Osnova rozprávania o transportnej vrstve

- 3.1 Služby transportnej vrstvy
- 3.2 Delenie správ a adresácia soketov
- 3.3 UDP: bezstavový transportný protokol
- 3.4 Princípy potvrdzovaného toku dát
- 3.5 TCP: stavový transportný protokol
  - pripájanie a odpájanie
  - štruktúra segmentu
  - potvrdzovaný tok dát
  - kontrola toku dát
- 3.6 Princípy zabezpečenia kontroly zahltenia
- 3.7 Kontrola zahltenia v protokole TCP

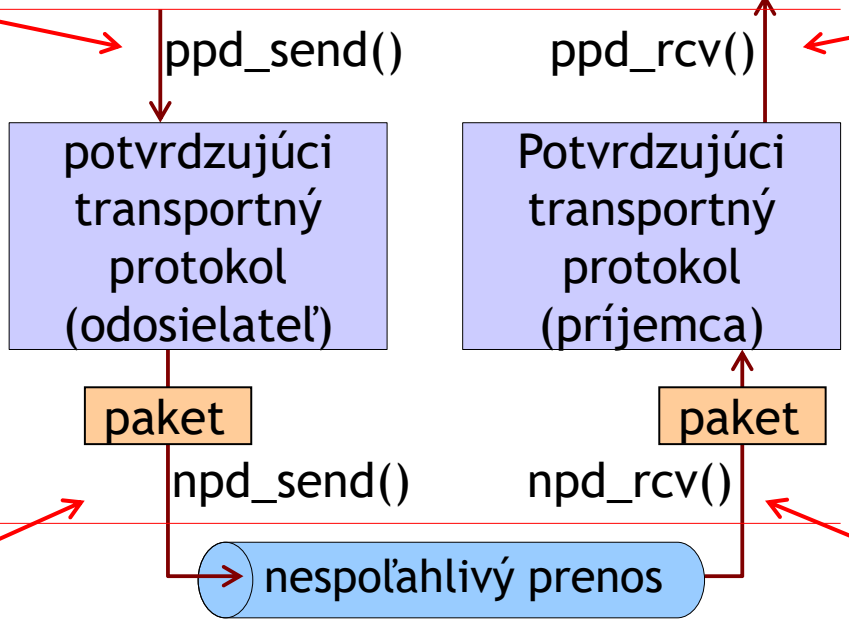
Princípy potvrdzovaného prenosu



Princípy potvrdzovaného prenosu

ppd_send() : volané z aplikačnej vrstvy, odovzdá dáta určené pre aplikačnú vrstvu príjemcu

ppd_rcv() : zavolané potvrdzujúcim protokolom, aby odovzdal dáta aplikačnej vrstve



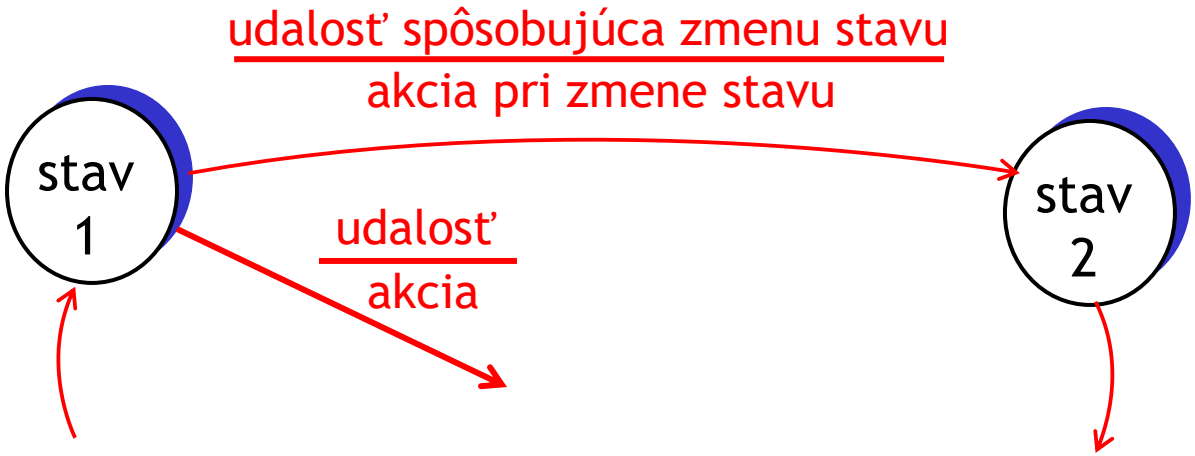
npd_send() : volané potvrdzujúcim protokolom na odoslanie paketu príjemcovi cez nespoľahlivý kanál

npd_rcv() : zavolané, keď paket dôjde cez nespoľahlivý kanál

Cesta k potvrdzovanému prenosu dát

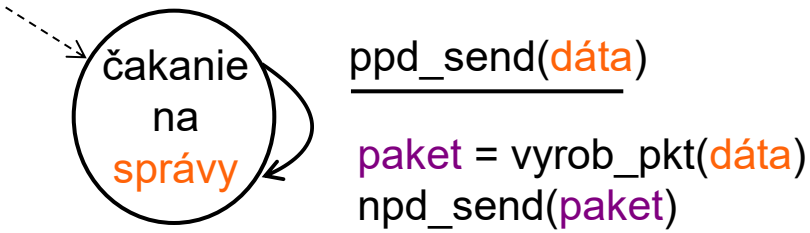
Náš postup:

- Postupne zlepšujeme odosielateľa a príjemcu potvrdzujúceho transportného protokolu
- Uvažujeme zatiaľ iba jednosmerný prenos dát
  - iba kontrolné dáta môžu putovať oboma smermi
- Na popis odosielateľa a príjemcu použijeme konečnostavové automaty

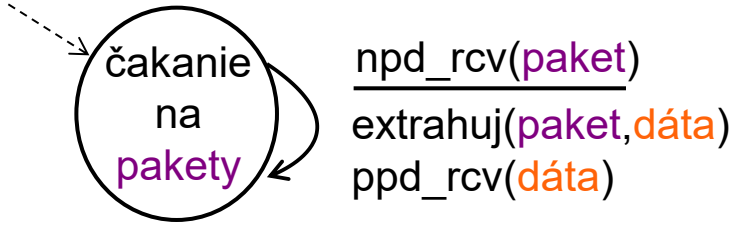


ppd 1.0: potvrdzovaný prenos dát cez spoľahlivý kanál

- používaný kanál je dokonale spoľahlivý
  - žiadne bitové chyby
  - žiadna strata paketov
- odosielateľ aj príjemca majú jednostavový automat:
  - odosielateľ posiela dáta do kanála
  - príjemca prijíma dáta z kanála



odosielateľ

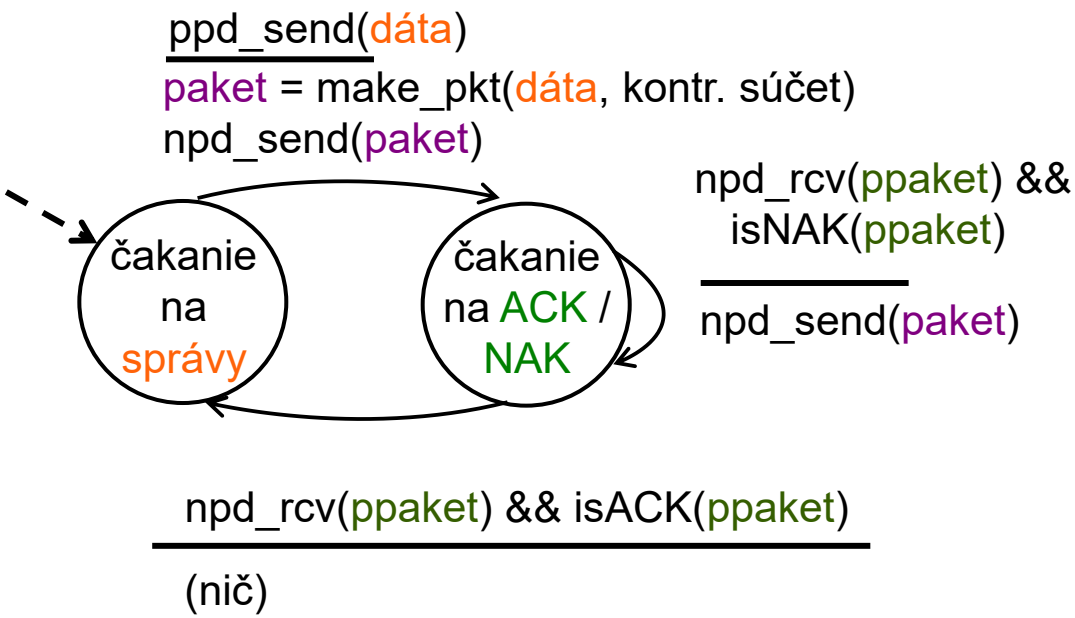


príjemca

ppd 2.0: kanál s bitovými chybami

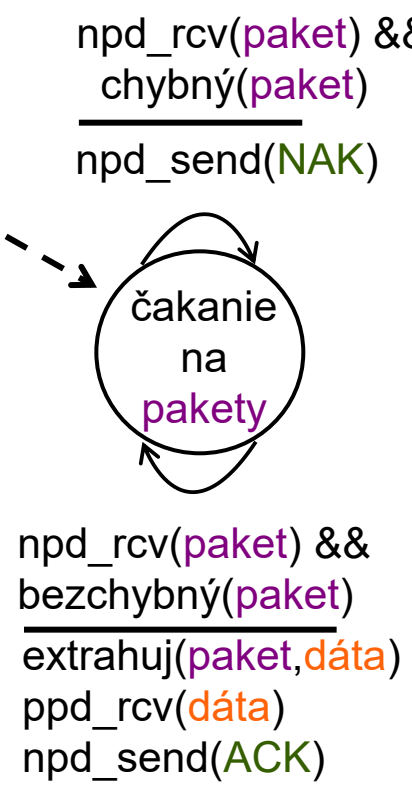
- používaný kanál môže prehodiť hodnotu niektorých bitov
  - použijeme kontrolný súčet na odhalenie chýb
- *otázka*: ako sa zotaviť z takýchto chýb:
  - *potvrdenia (ACKs):* príjemca explicitne povie odosielateľovi, že paket prijal v poriadku
  - *negatívne potvrdenia (NAKs):* príjemca explicitne povie odosielateľovi, že paket prijal s chybami
  - odosielateľ pošle paket ešte raz vždy, keď prijme NAK
- vylepšenie ppd 2.0 oproti ppd 1.0:
  - odhaľovanie chýb
  - spätná väzba od príjemcu: kontrolné správy (ACK,NAK)

ppd 2.0: automaty



odosielateľ

príjemca



ppd 2.0 má fatálny nedostatok!

Čo ak sú ACK/NAK poškodené?

- odosielateľ nevie čo sa stalo na strane príjemcu!
- nemôže jednoducho preposlať: možná duplicita

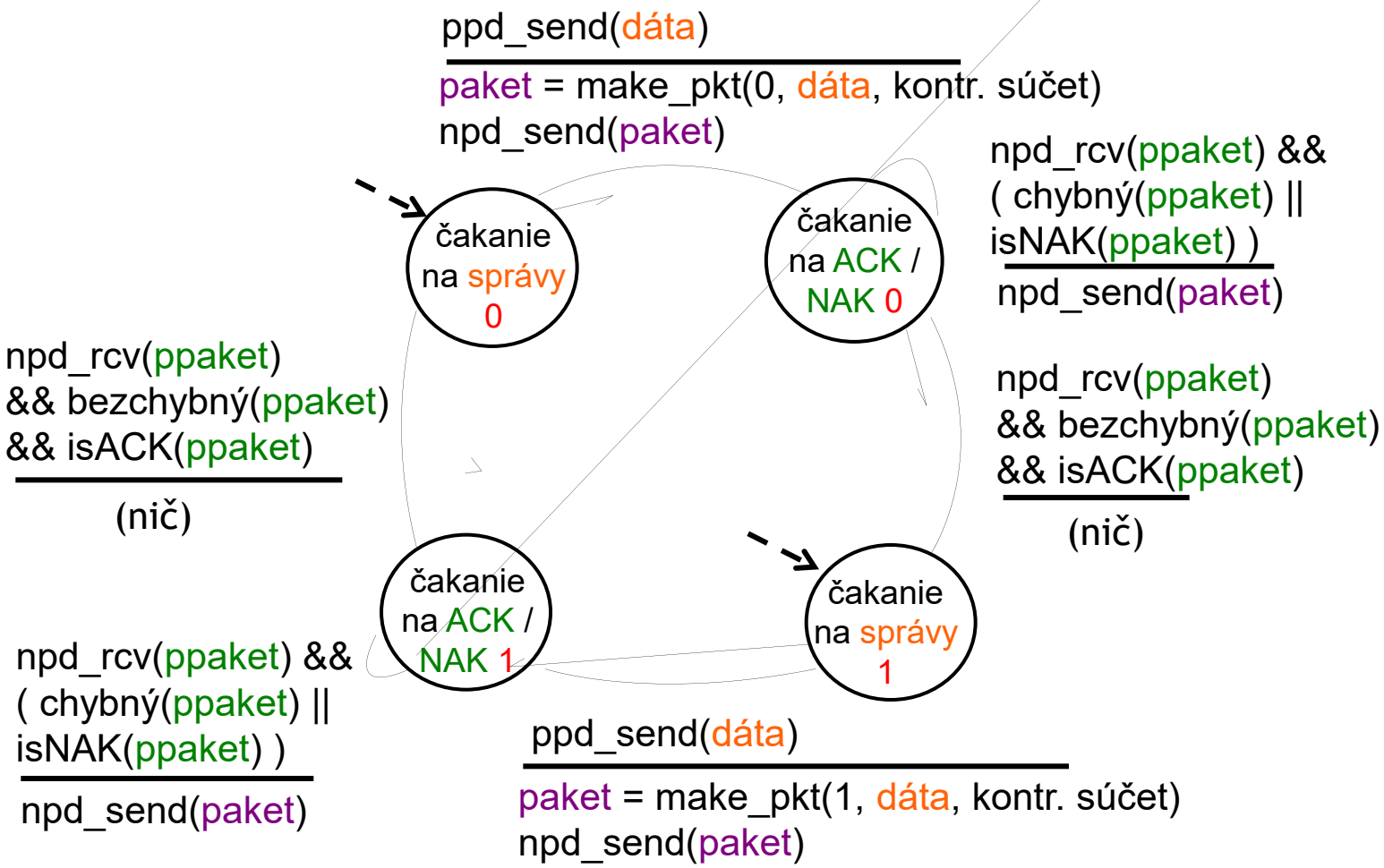
Riešenie duplicít:

- odosielateľ znova pošle aktuálny paket, ak je ACK/NAK poškodené
- odosielateľ pridá *sekvenčné číslo* každému paketu
- príjemca duplicity zahadzuje

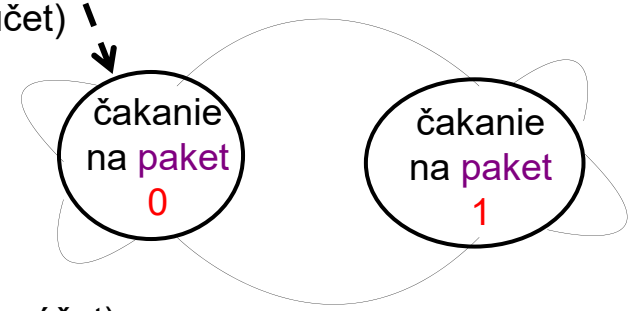
stop and wait

Odosielateľ čaká po každom odoslanom pakete na odpoveď príjemcu

ppd 2.1: odosielateľ kt. zvládne poškodené ACK/NAK



ppd 2.1: príjemca



npd_rcv(paket) && bezchybný(paket)
&& has_seq0(paket)

extrahuj(paket,dáta)
ppd_rcv(dáta)
ppaket = make_pkt(ACK, kontr. súčet)
npd_send(ppaket)

npt_rcv(paket) && chybný(paket)

ppaket = make_pkt(NAK, kontr. súčet)
npd_send(ppaket)

npd_rcv(paket) && chybný(paket)

ppaket = make_pkt(NAK, kontr. súčet)
npd_send(ppaket)

npd_rcv(paket) &&
bezchybný(paket) &&
has_seq1(paket)

ppaket = make_pkt(ACK, kontr. súčet)
npd_send(ppaket)

npd_rcv(paket) &&
bezchybný(paket) &&
has_seq0(paket)

ppaket = make_pkt(ACK, kontr. súčet)
npd_send(ppaket)

npd_rcv(paket) && bezchybný(paket)
&& has_seq1(paket)

extrahuj(paket,dáta)
ppd_rcv(dáta)
ppaket = make_pkt(ACK, kontr. súčet)
npd_send(ppaket)

ppd 2.1: diskusia

Odosielateľ:

- pridanie sekvenčného čísla do paketu
- dve sekvenčné čísla (0,1) stačia. Prečo?
- musí zisťovať chybovosť prijatých ACK/NAK paketov
- dvojnásobok stavov
  - extra stavy si "pamätajú", či odoslaný paket mal číslo 0 alebo 1

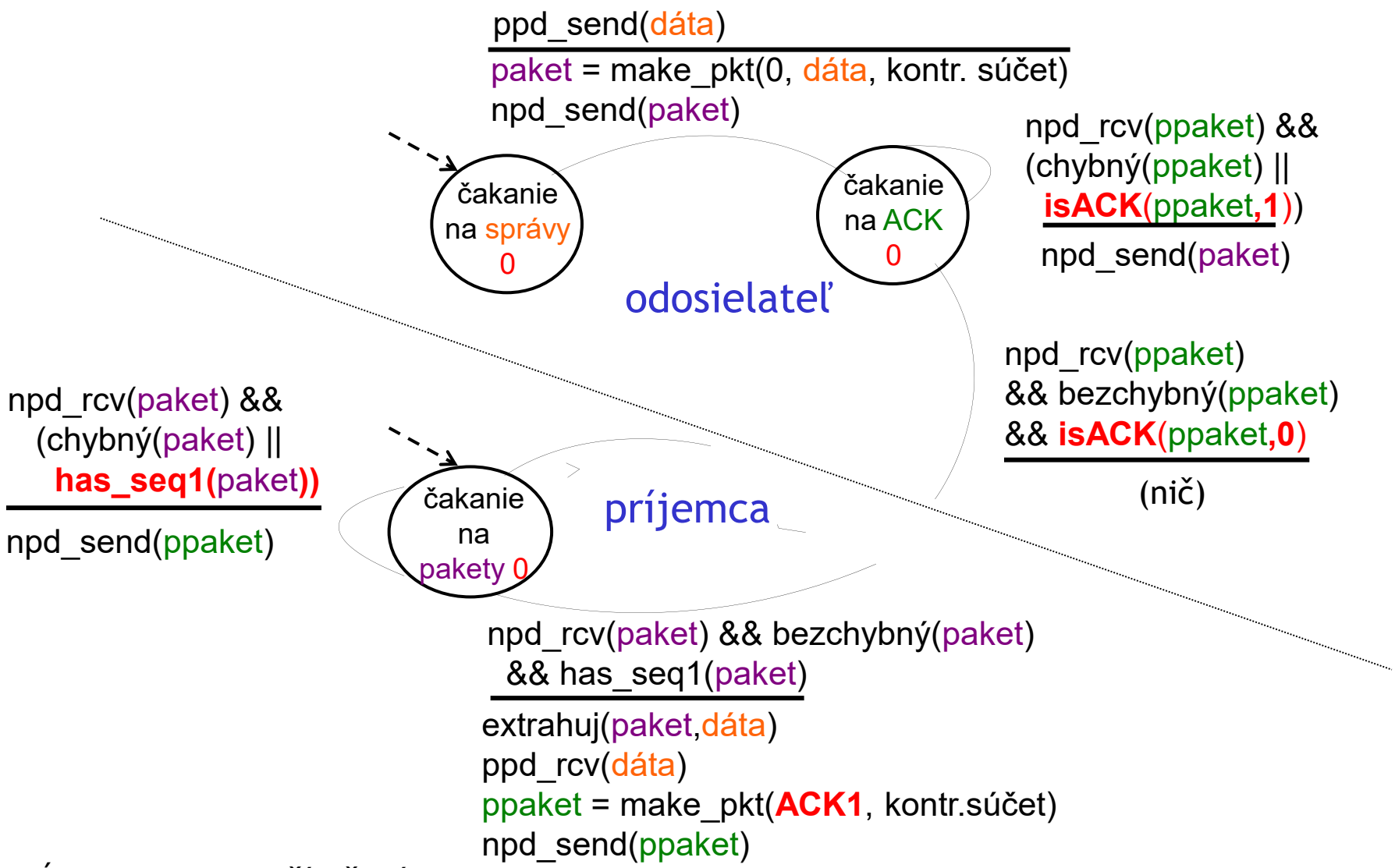
Príjemca:

- musí zisťovať duplicitu paketov
  - stav identifikuje, či očakávame paket so sekvenčným číslom 0 alebo 1
- poznámka: príjemca nevie, či ACK/NAK bol odosielateľovi doručený bez chyby

ppd 2.2: protokol bez NAK správ

- rovnaká funkcionalita ako ppd 2.1, ale používame už iba ACK potvrdenia
- namiesto správy NAK, príjemca odošle ACK so sekvenčným číslom posledného bezchybného paketu
  - potvrdzovací ACK paket musí obsahovať číslo paketu, ktorý potvrdzujeme
- dva ACK pakety s rovnakým číslom majú rovnaký dôsledok ako NAK: *pošli aktuálny paket znova*

ppd 2.2: polovice automatov odosielateľa a príjemcu



ppd 3.0: kanál s bitovými chybami a stratami paketov

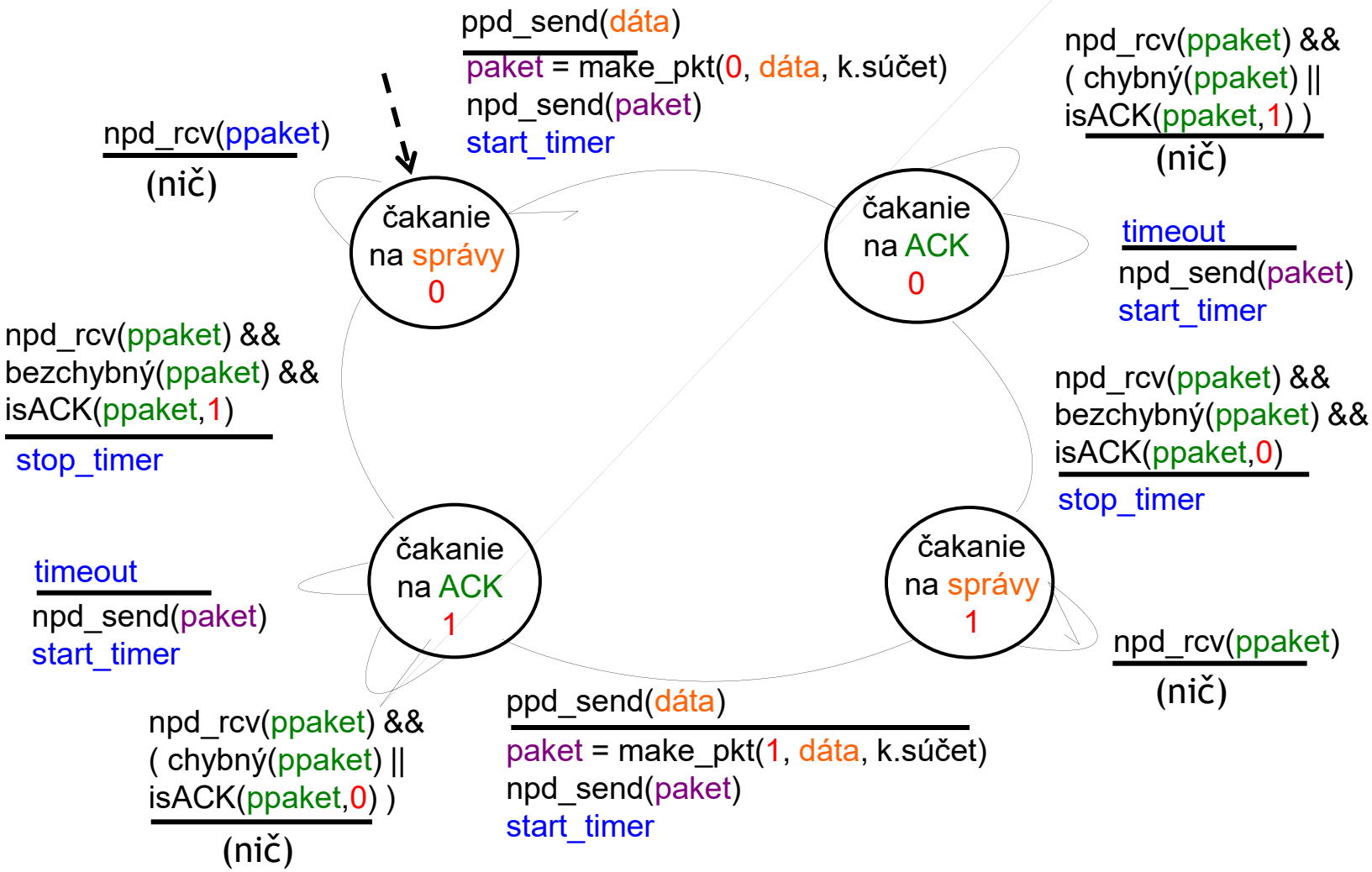
Nový predpoklad: použitý kanál môže aj stratiť pakety (dáta alebo potvrdenia)

- kontrolný súčet, sekvenčné číslo a potvrdenia ACK využijeme, ale to nestačí

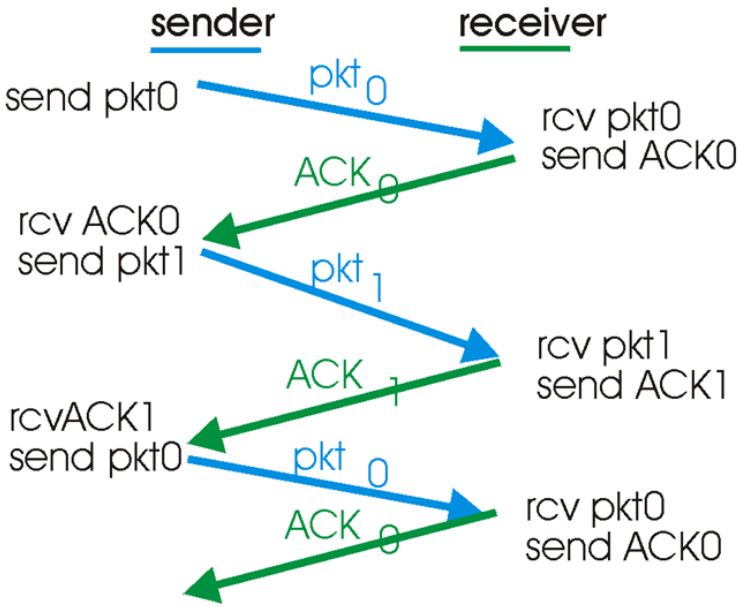
Prístup: odosielateľ čaká "rozumný" čas na ACK

- ak v tomto čase žiadne ACK neprišlo, pošle paket znova
- ak sa paket (alebo ACK) iba oneskorí (nestratí):
  - posielanie bude spôsobovať duplicity, ale to vieme vyriešiť cez sekvenčné číslo
  - príjemca musí špecifikovať sekvenčné číslo v ACK
- vyžaduje "časovač": meranie času od posledného odoslania paketu

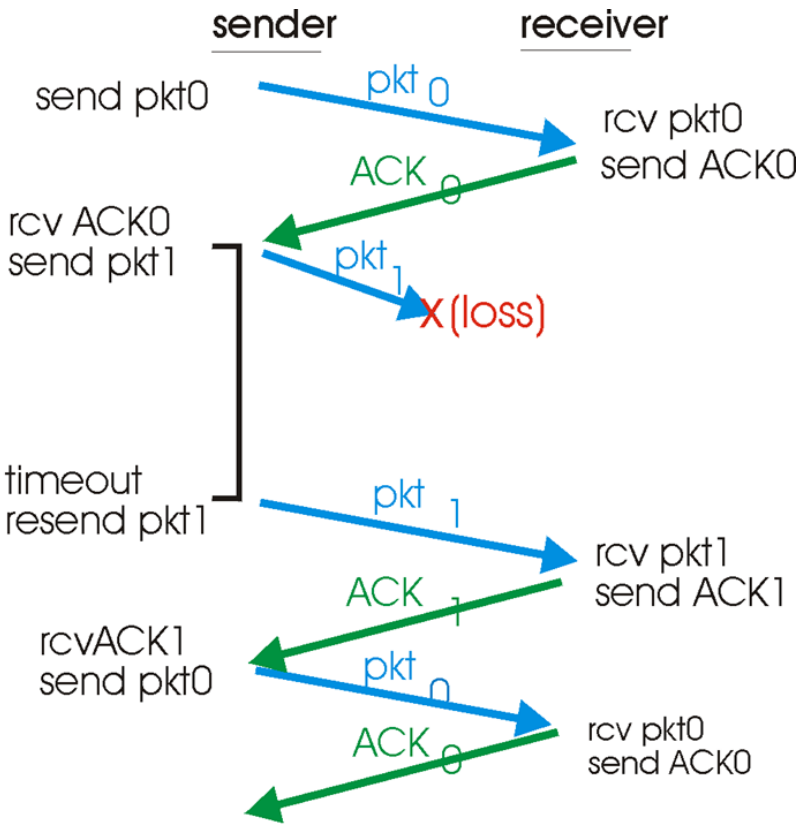
ppd 3.0 odosielateľ



ppd 3.0 v čase

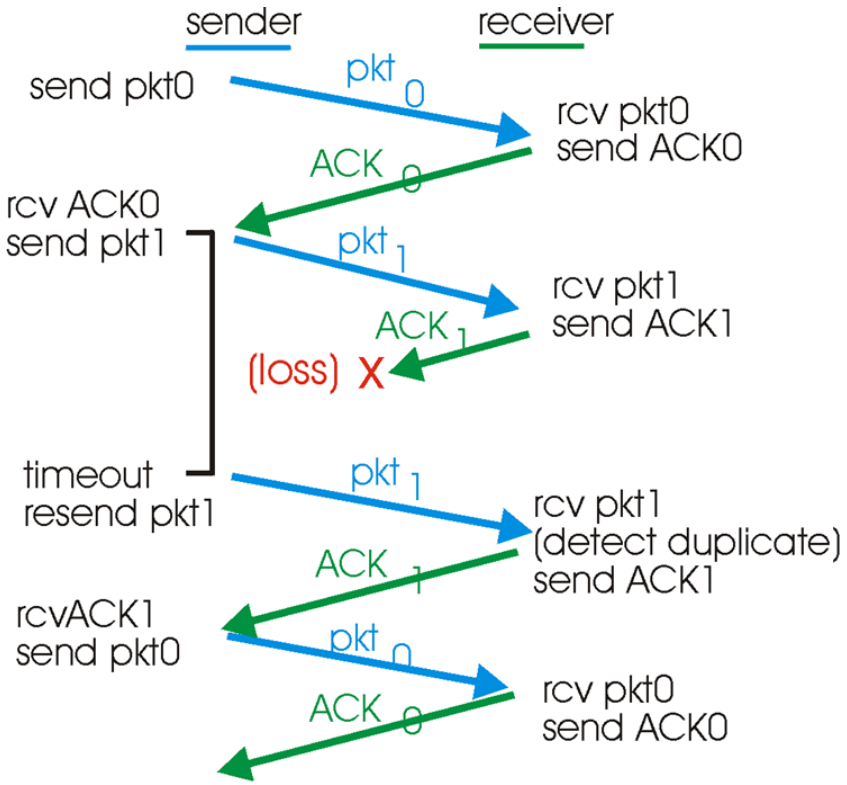


(a) operation with no loss

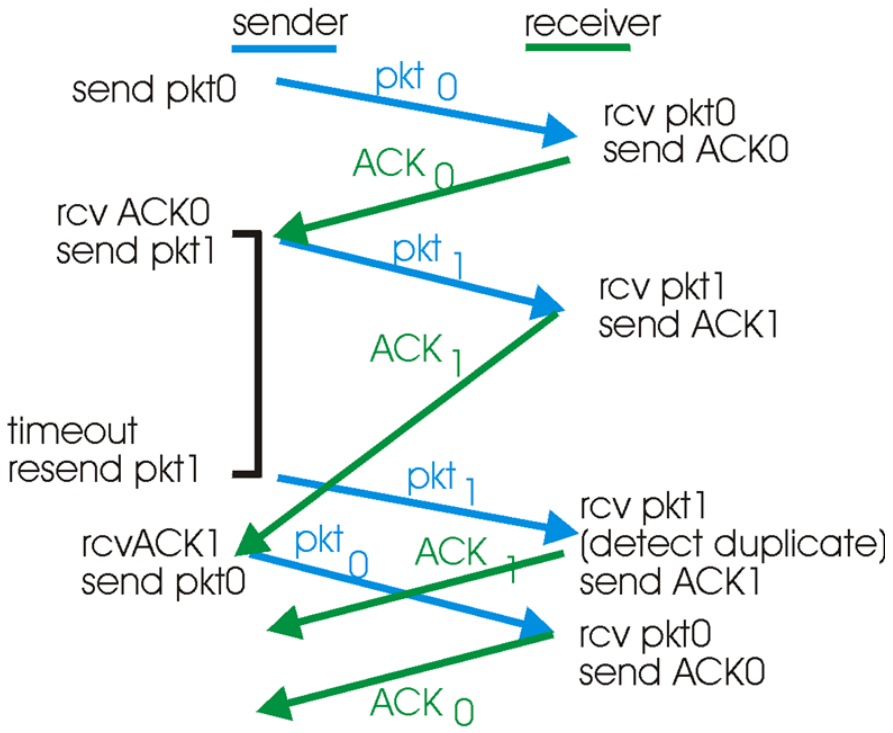


(b) lost packet

ppd 3.0 v čase



(c) lost ACK



(d) premature timeout

rýchlosť ppd 3.0

- ppd 3.0 funguje (použitie je napr. vo WiFi), ale výkon je biedny
- príklad: 1 Gb/s spoj, zdržanie prenosom 15 ms, pakety veľkosti 1kB:

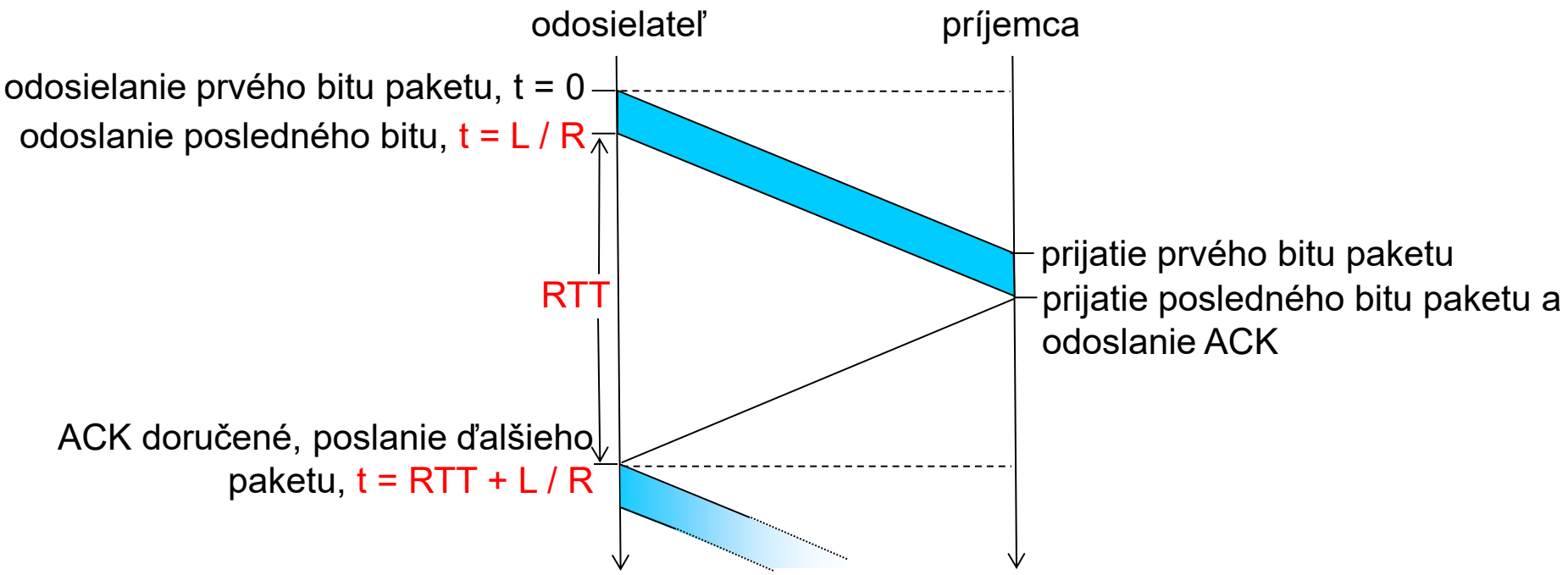
$$T_{odoslanie} = \frac{L\ (\text{počet bitov paketu})}{R\ (\text{šírka pásma v b/s})} = \frac{8kb}{10^9\ b/s} = 8\ \text{mikrosekúnd}$$

- E: efektivita – zlomok času v ktorom sa odosiela

$$E = \frac{L\ /\ R}{RTT + L\ /\ R} = \frac{0{,}008\ ms}{30{,}008\ ms} = 0.00027$$

- 1kB paket za 30 ms -> 33kB/s prenos cez 1 Gb/s spoj
- protokol obmedzuje fyzické možnosti spoja!

ppd3.0: stop-and-wait

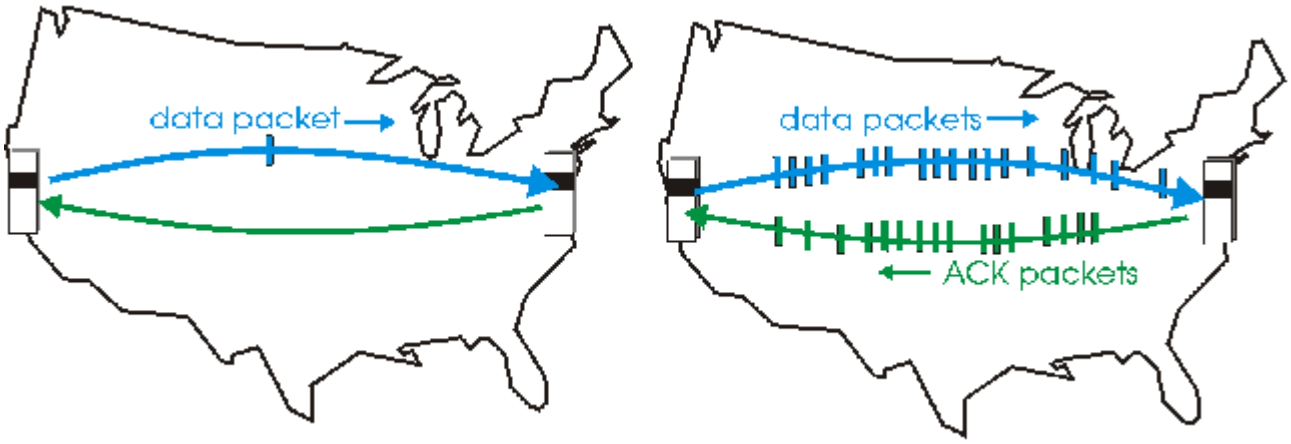


$$E = \frac{L / R}{RTT + L / R} = \frac{0{,}008\ ms}{30{,}008\ ms} = 0.00027$$

Protokoly s pipeliningom

Pipelining: odosielateľ odosiela viac paketov, pokiaľ mu prídu ACK pakety

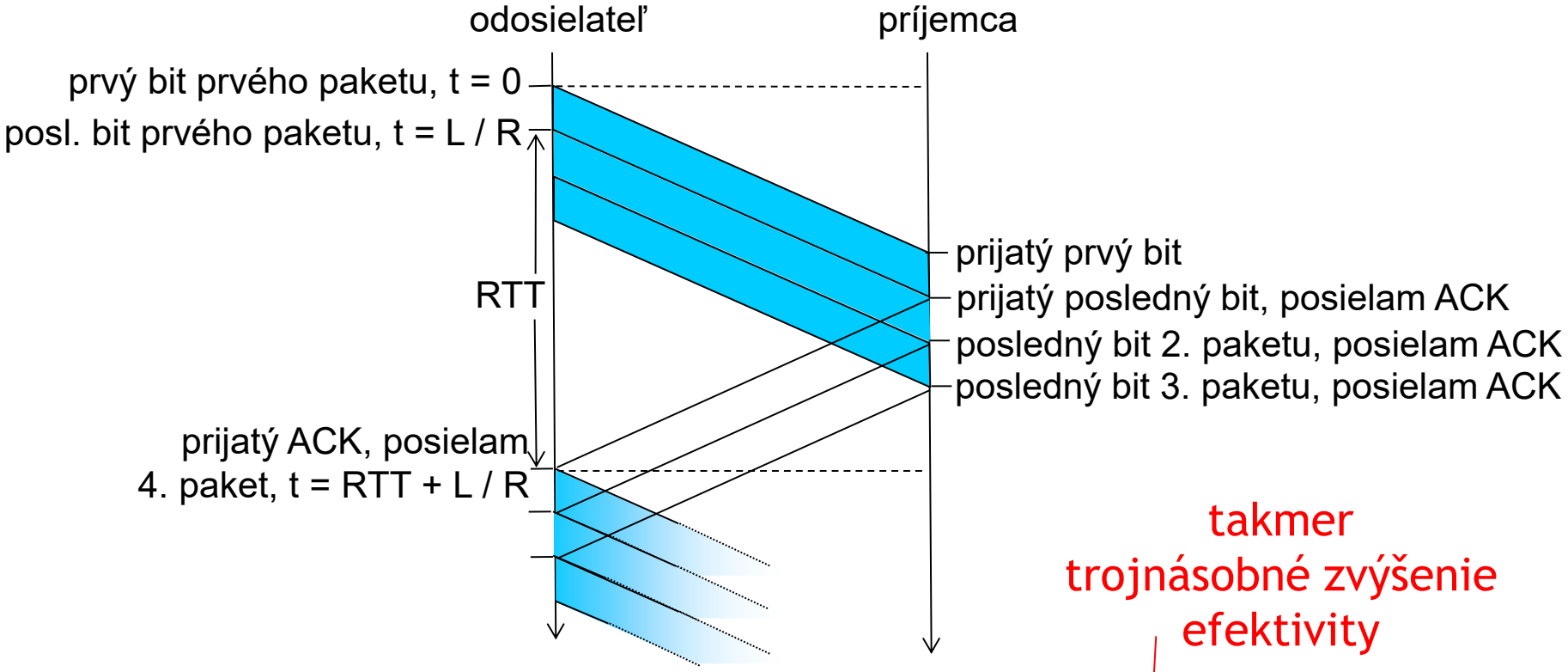
- rozsah sekvenčných čísiel sa musí zväčšiť
- odosielateľ aj príjemca musia udržiavať buffer paketov



stop-and-wait

pipelining

Pipelining: zvýšenie efektivity



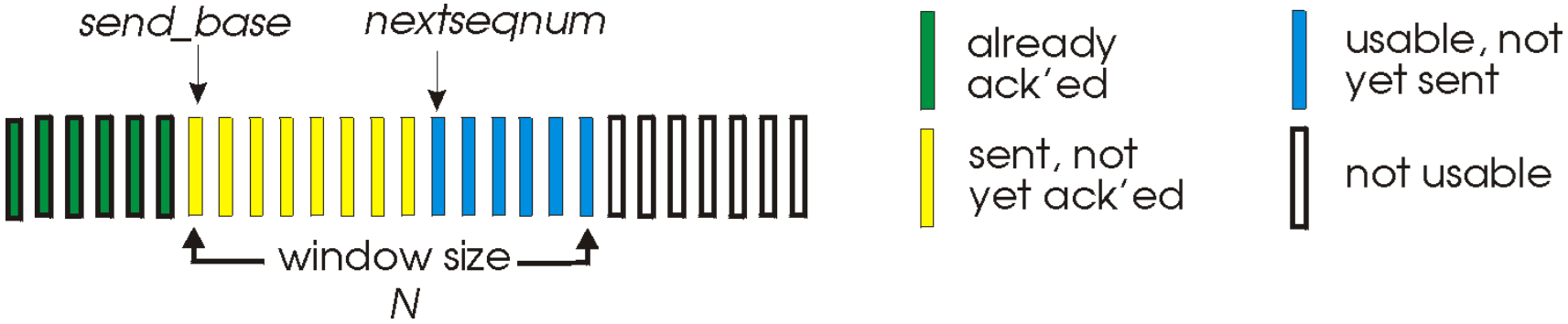
takmer trojnásobné zvýšenie efektivity

$$E = \frac{3 * L / R}{RTT + 3 * L / R} = \frac{0{,}024 \text{ ms}}{30{,}024 \text{ ms}} = 0.0008$$

Go-Back-N

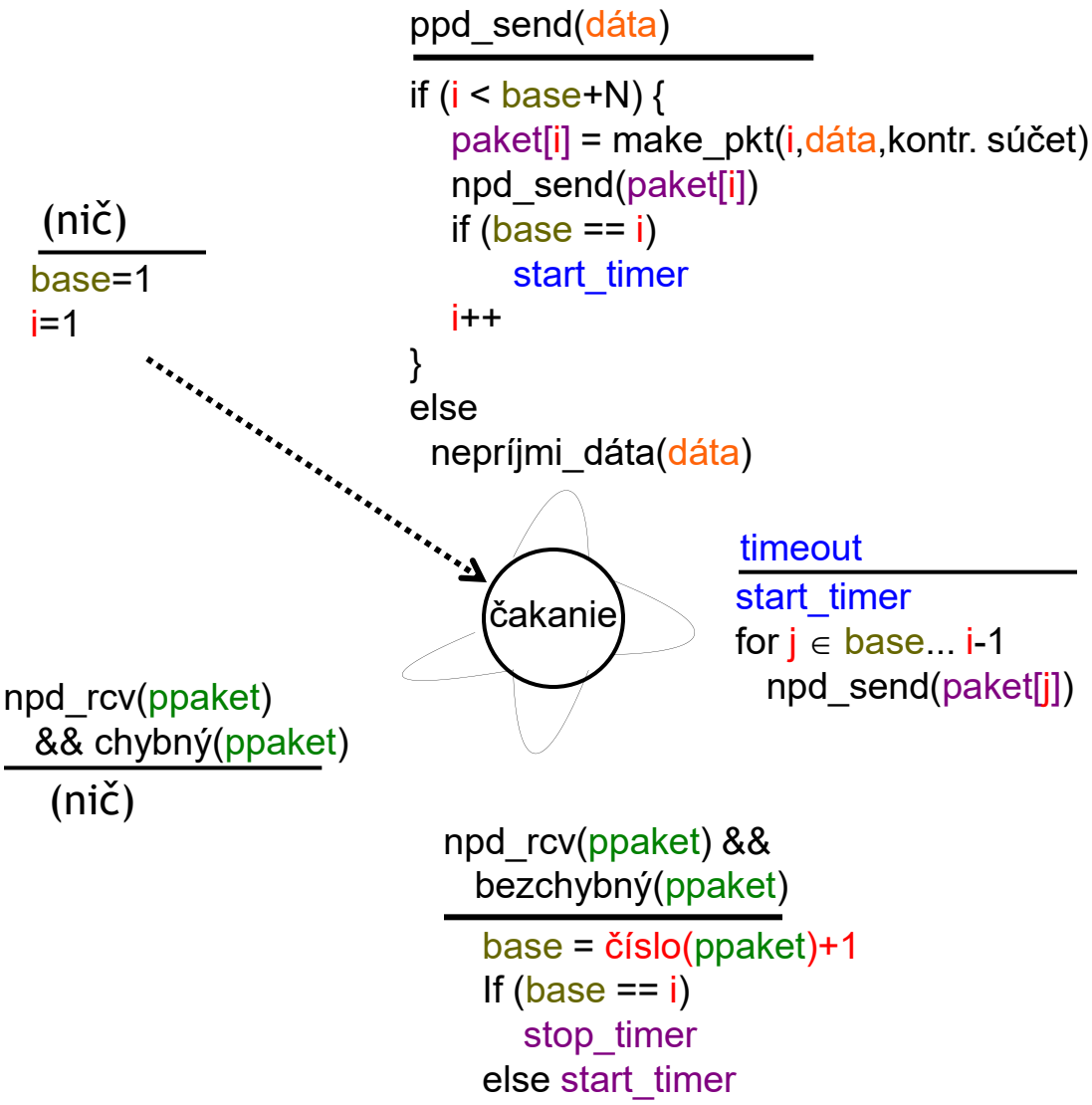
Odosielateľ:

- k-bitové sekvenčné číslo v hlavičke paketu
- "okno" veľkosti N určujúce, koľko odoslaných paketov nemusí byť potvrdených

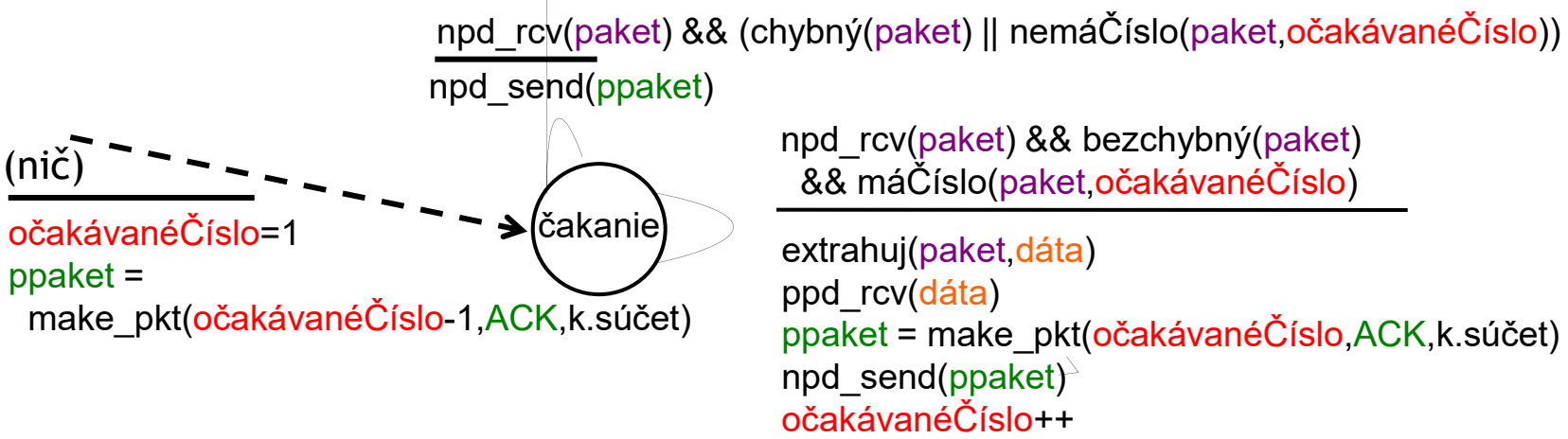


- ACK(n): kumulatívne potvrdenie za prijatie súvislej časti paketov až po sekvenčné číslo n vrátane - "kumulatívny ACK"
  - aj tu sa môže stať, že prídu rovnaké ACK pakety
- časovač pre okno paketov (štart časovača pri novom *send_base*)
- timeout: vykonáme opätovné preposlanie paketu na *send_base* pozícii a aj všetkých odoslaných paketov s vyšším číslom

Go-Back-N: odosielateľ

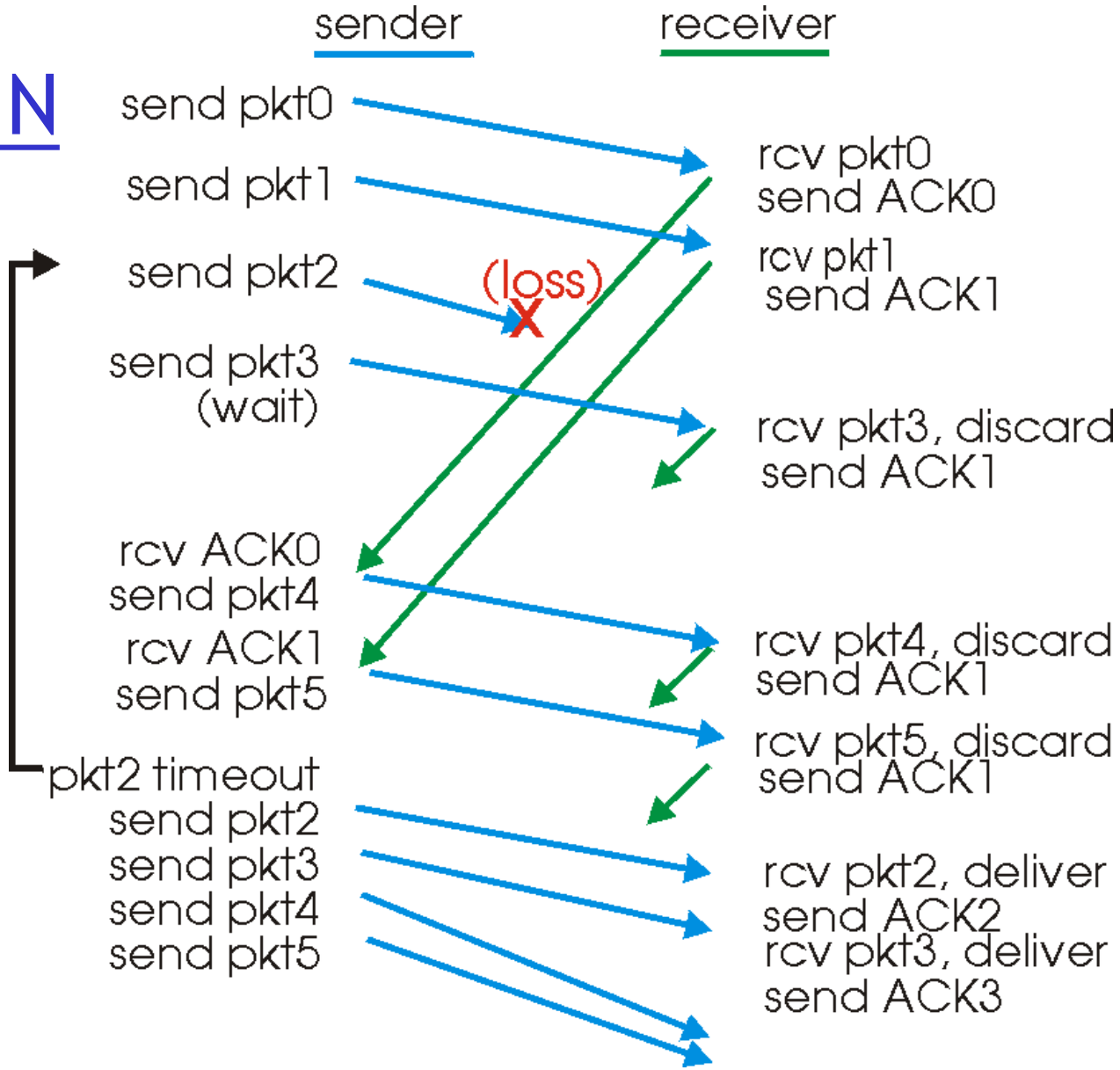


Go-Back-N: príjemca



- vždy odošle ACK pre bezchybne doručený paket s najväčším číslom v poradí bezchybne doručených
  - môžeme posielať rovnaké ACK viac krát
  - potrebujeme si pamätať iba očakávané číslo
- pakety mimo poradia doručených paketov:
  - zahodíme (nejdú do buffra) -> buffer príjemca nepotrebuje!
  - opäť pošleme ACK s posledným číslom v poradí správne doručených

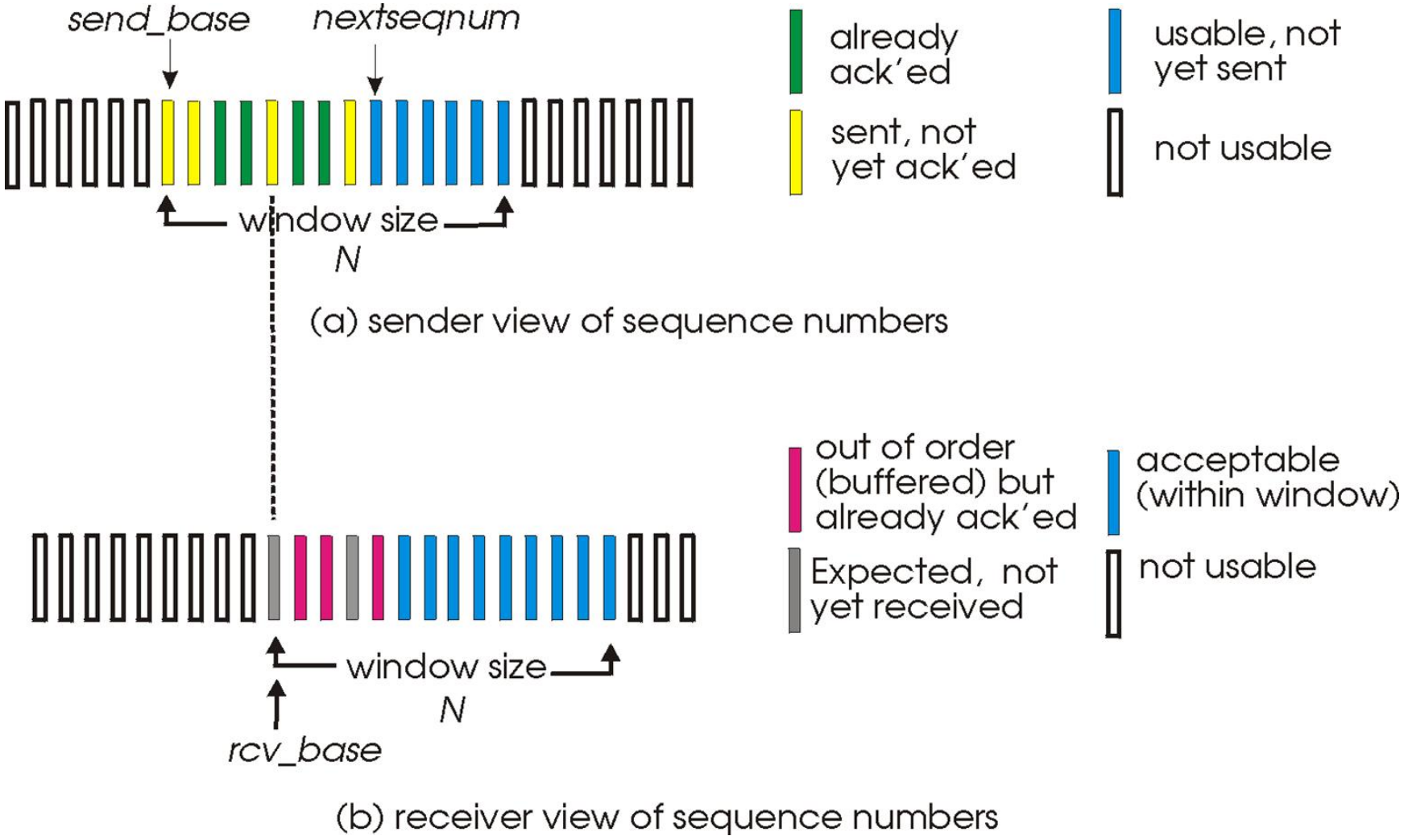
Go-Back-N


sender
receiver
send pkt0
send pkt1
send pkt2
(loss)
send pkt3
(wait)
rcv pkt0
send ACK0
rcv pkt1
send ACK1
rcv pkt3, discard
send ACK1
rcv ACK0
send pkt4
rcv ACK1
send pkt5
rcv pkt4, discard
send ACK1
rcv pkt5, discard
send ACK1
pkt2 timeout
send pkt2
send pkt3
send pkt4
send pkt5
rcv pkt2, deliver
send ACK2
rcv pkt3, deliver
send ACK3

Selektívne opakovanie

- príjemca potvrdzuje každý bezchybný paket zvlášť
  - buffer pre pakety, ktoré majú vyššie čísla ako bezchybne doručený paket s najväčším číslom v poradí bezchybne doručených
- odosielateľ preposiela znova iba tie pakety, pre ktoré neprijal potvrdzovacie ACK pakety
  - časovač pre každý nepotvrdený paket
  - okno odoslaných paketov a povolených na odoslanie

Selektívne opakovanie: okná odosielateľa aj príjemcu



Selektívne opakovanie

odosielateľ

dáta z aplikačnej vrstvy:

- ak je voľné sekvenčné číslo v okne, pošle paket

timeout(n):

- pošle znova paket s číslom n a reštartuje jeho časovač

ACK(n) v okne odosielateľa:

- označí paket n ako doručený
- ak n bol paket s najmenším číslom v okne, posuň okno tak, aby začínalo na prvom nepotvrdenom pakete

príjemca

paket s číslom n z intervalu [rcv_base, rcv_base+N-1]

- pošli ACK(n)
- mimo poradia: daj do buffra
- v poradí: pošli paket aplik. vrstve spolu so všetkými z buffra v poradí za ním, a posuň okno k ďalšiemu nedoručenému paketu

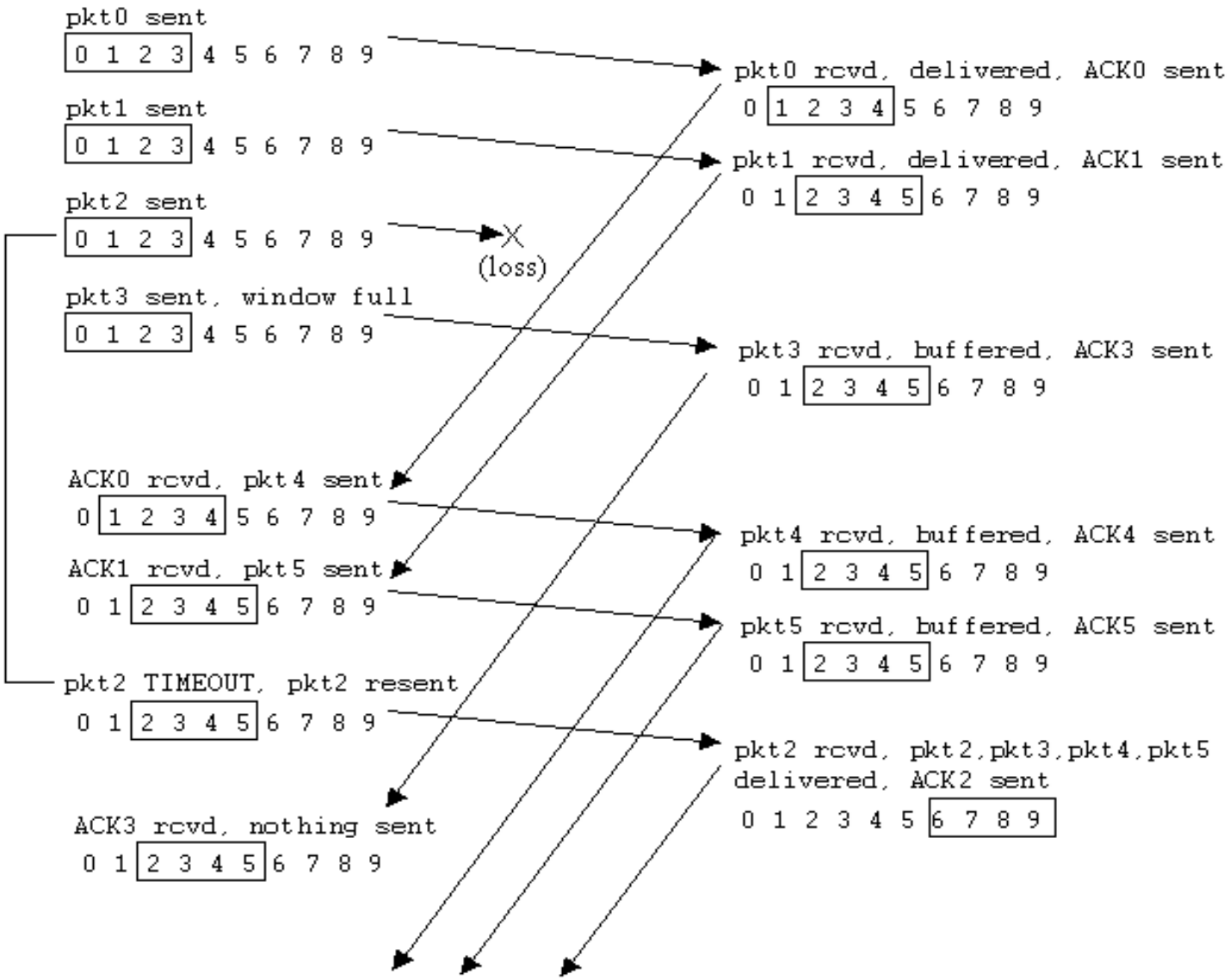
paket s číslom n z intervalu [rcvbase-N,rcvbase-1]

- pošli ACK(n)

inak:

- ignoruj

Selektívne opakovanie v čase



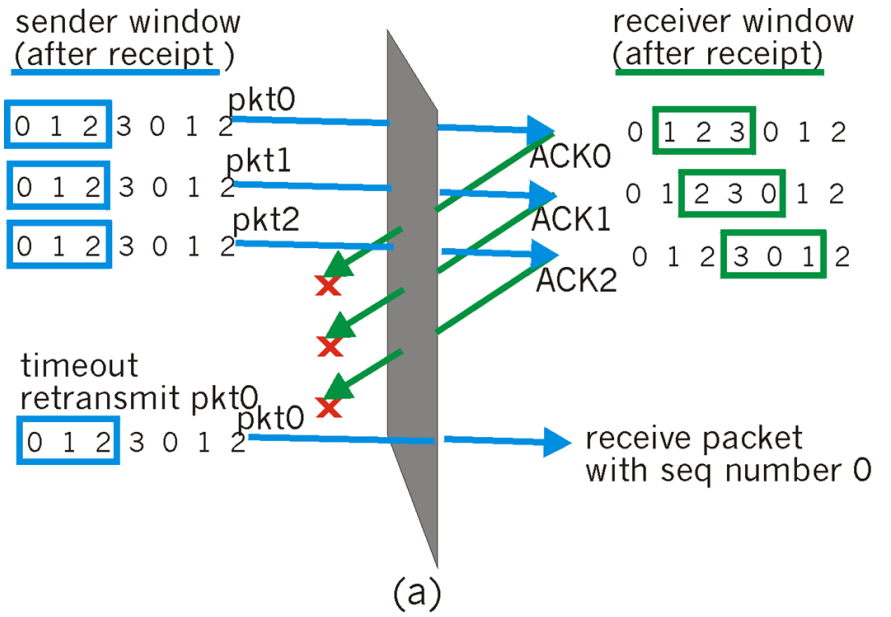
Selektívne opakovanie

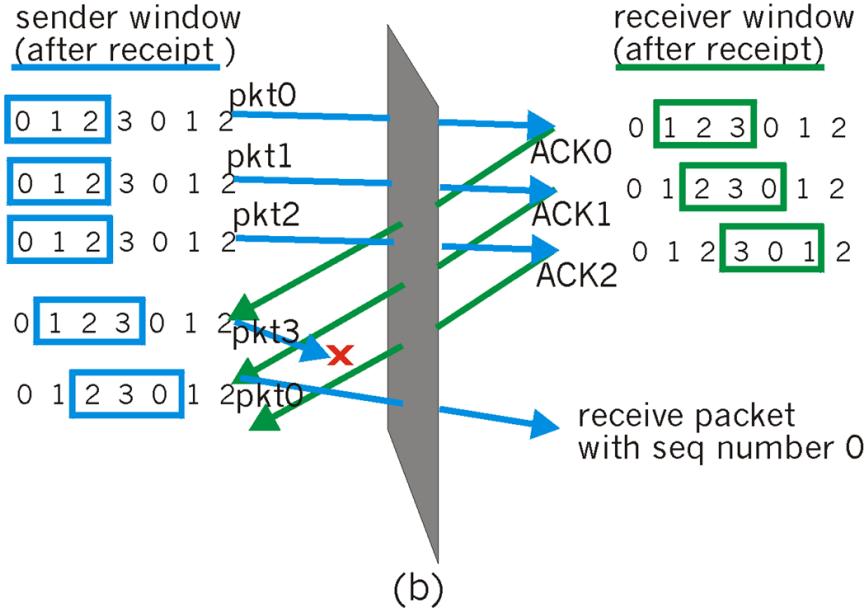
Príklad:

- sekv. čísla: 0, 1, 2, 3
- veľkosť okna=3

- príjemca vidí v oboch prípadoch to isté!
- nesprávne opätovné posielanie, ktoré sa javí ako nové

Otázka: aký je vzťah medzi počtom sekvenčných čísiel a veľkosťou okna?





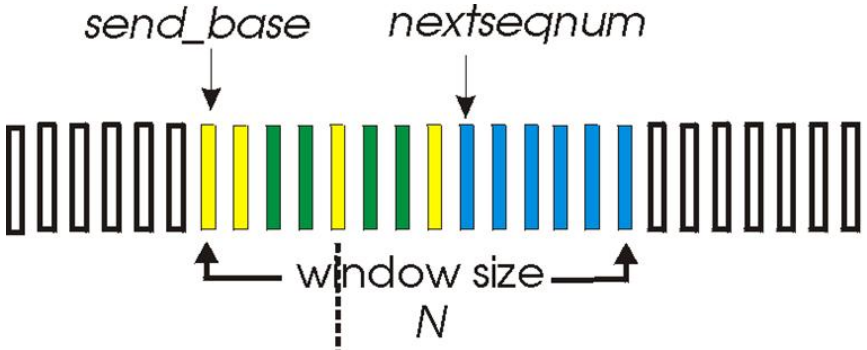
Osnova rozprávania o transportnej vrstve

- 3.1 Služby transportnej vrstvy
- 3.2 Delenie správ a adresácia soketov
- 3.3 UDP: bezstavový transportný protokol
- 3.4 Princípy potvrdzovaného toku dát
- 3.5 TCP: stavový transportný protokol
  - pripájanie a odpájanie
  - štruktúra segmentu
  - potvrdzovaný tok dát
  - kontrola toku dát
- 3.6 Princípy zabezpečenia kontroly zahltenia
- 3.7 Kontrola zahltenia v protokole TCP

Potvrdzovaný prenos dát cez TCP

- TCP využíva nespoľahlivý (best effort) prenos dát protokolom IP
- pipelining segmentov
  - okno príjemcu aj odosielateľa
- kumulatívne potvrdenia
- jediný časovač na preposielania
- opätovné posielanie je spôsobené:
  - timeout-om
  - viacnásobným potvrdením

Okná v TCP: takmer ako pri selektívnom opakovaní



(a) sender view of sequence numbers

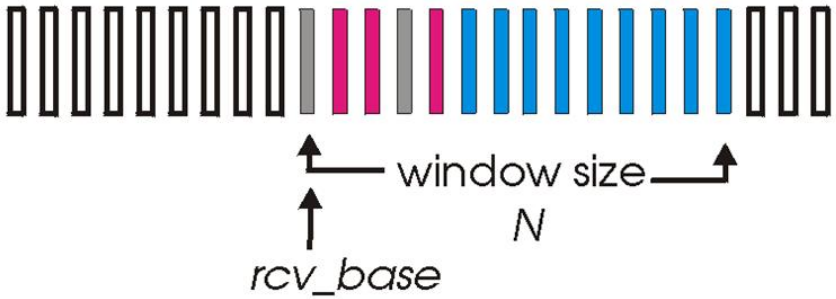
already ack'ed

sent, not yet ack'ed

usable, not yet sent

not usable

takéto okamžite spôsobia posunutie okna (kumulatívne potvrdenie)



(b) receiver view of sequence numbers

out of order (buffered) but already ack'ed

Expected, not yet received

acceptable (within window)

not usable

potvrdzujeme vždy číslom rcv_base

Ďakujem za pozornosť

Modifikované slajdy z knihy:

Computer Networking: A Top Down Approach ,
4th edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2007.