Sadržaj kolegija „Mreže računala I“

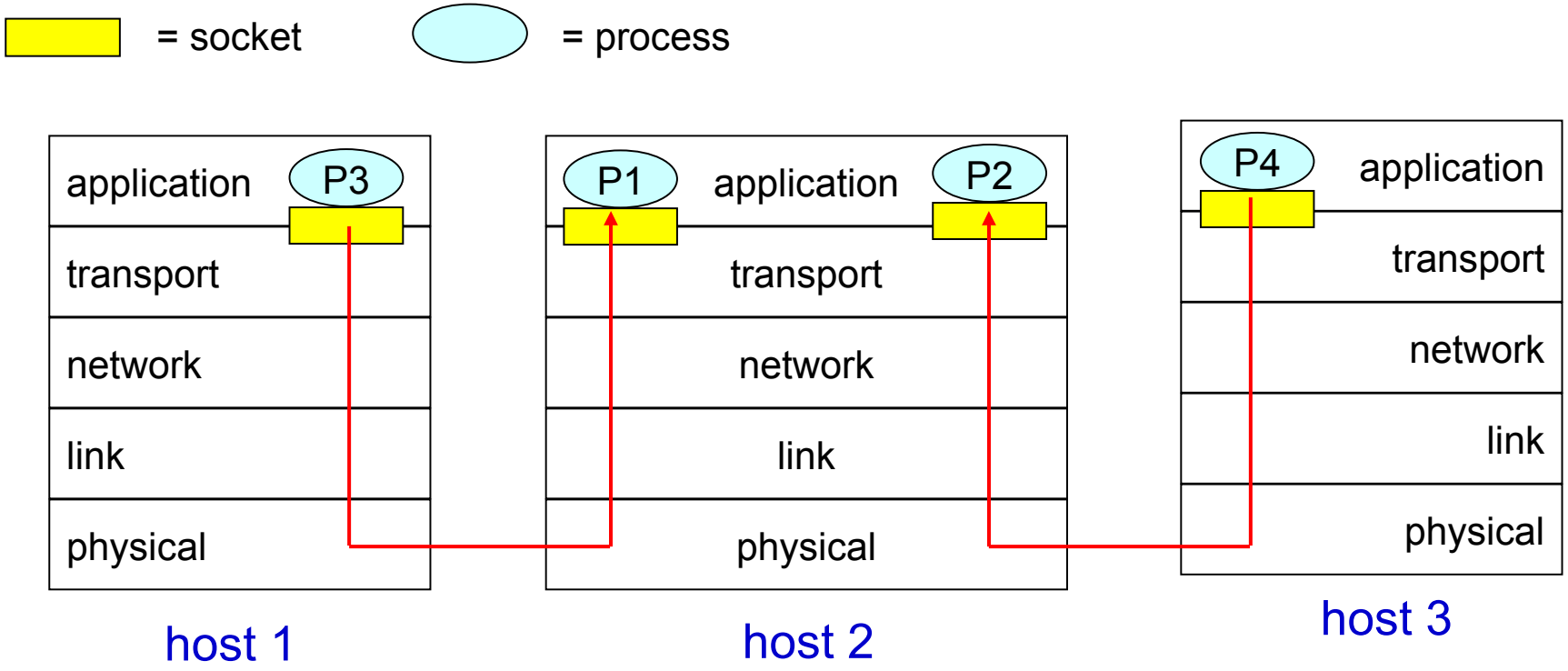
- Uvod
- Aplikacijski sloj
- Transportni sloj
- Sloj mreže
- Sloj veze
- Fizički sloj

Transportni sloj

- Uvod
- UDP
- Kontrola pogrešaka

Uvod

- Zadatak transportnog sloja: komunikacija između korisničkih procesa



Uvod

- Moguće karakteristike usluge
  - kontrola pogrešaka
  - osiguranje redoslijeda
  - orijentiran na vezu/nije orijentiran na vezu
  - kontrola toka i opterećenja
  - osiguranje kvalitete usluge (npr. brzina prijenosa, kašnjenje, gubitak)
- User Datagram Protocol (UDP)
  - nije orijentiran na vezu; nema kontrolne mehanizme; ne osigurava redoslijed
  - sučelje za jednostavnu komutaciju paketa pomoću IP, odgovornost za kontrolne mehanizme kod aplikacija
- Transmission Control Protocol (TCP)
  - orijentiran na vezu; kontrola pogrešaka, toka i opterećenja; ne osigurava kvalitetu usluge
  - apstrahira tok byte-ova (byte stream)

Transportni sloj

- Uvod
- UDP
- Kontrola pogrešaka

UDP

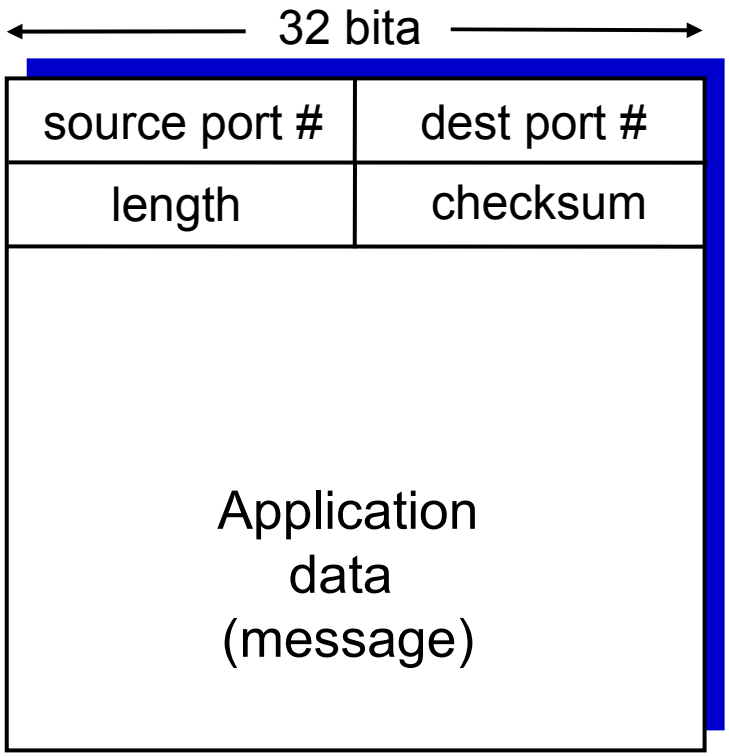
- Segment:

  source port:
  broj izvornog porta (16 bitova)

  dest port:
  broj odredišnog porta (16 bitova)

  length:
  duljina cijelog segmenta
  (16 bitova)

  checksum:
  kontrolni zbroj (16 bitova)
  za kontrolu pogrešaka,
  upotreba je opcionalna,
  0000000000000000_2 znači:
  nije korišten



UDP

- Multipleksiranje i demultipleksiranje

  multipleksiranje: prijenos segmenata različitih korisničkih procesa transportnim slojem na izvornom hostu

  demultipleksiranje: predaja segmenata različitim korisničkim procesima na transportnom sloju odredišnog hosta

  korisnički proces dogovara s transportnim slojem na izvornom hostu broj izvornog porta (izabire ga ili aplikacija ili se od strane operacijskog sustava dodjeljuje neki slobodan port)

  realiziran npr. kroz socket-API:

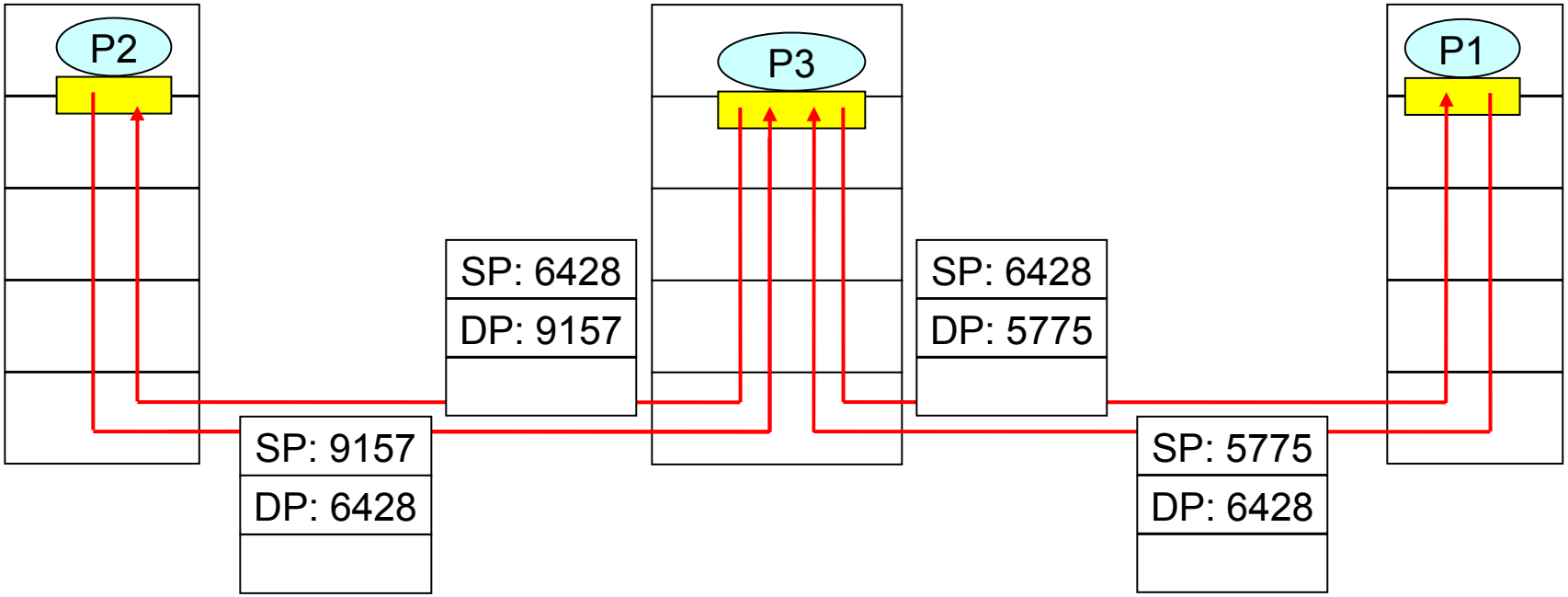
```
DatagramSocket serverSocket =
                    new DatagramSocket(6428);
serverSocket.send(aPacket);
```

  UDP na odredišnom hostu odlučuje prema broju odredišnog porta (i samo prema njemu) kojoj aplikaciji se segment dodjeljuje

  korisnički proces može sadržavati više socket-a

UDP

- Multipleksiranje i demultipleksiranje, primjer:



UDP

- Izračunavanje kontrolnog zbroja

  segment je prikazan kao niz binarnih brojeva duljine 16 bitova

  ovi bitovi se zbrajaju u tzv. aritmetici jednog komplementa (one's complement arithmetic):

  - -x nastaje iz x invertiranjem svih bitova
  - nastane li ostatak (carry), rezultat se inkrementira

  rezultat se invertira i to je kontrolni zbroj

  pošiljatelj računa kontrolni zbroj i upisuje ga u segment

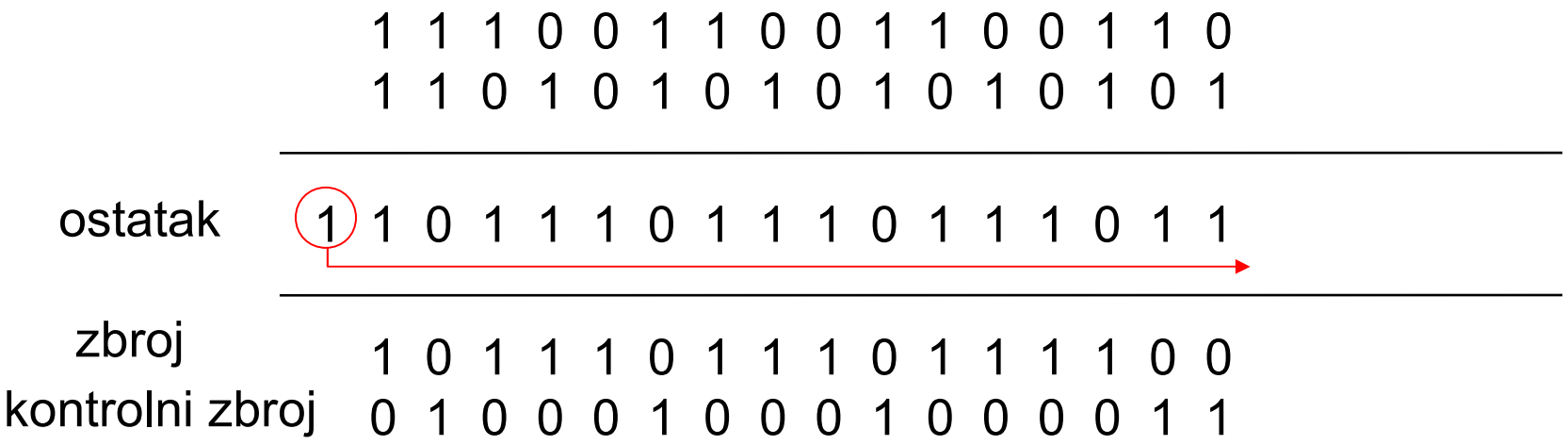
  na isti način primatelj računa kontrolni zbroj i dodaje (u aritmetici jednog komplementa) kontrolni zbroj pročitan iz segmenta

  ako ne postoji pogreška u bitu, onda kao rezultat nastaje 1111111111111111_2 (prikaz 0 u jednom komplementu)

  pojedine pogreške bita se mogu prepoznati, ali ne i dvostruke

UDP

- Izračunavanje kontrolnog zbroja, primjer:



UDP

- Pseudo-zaglavlje

  Pseudo-zaglavlje sadrži izvornu i odredišnu IP adresu, broj protokola (17 za UDP) i duljinu segmenta

  UDP pošiljatelja najprije upisuje 0 u checksum polje, generira pseudo-zaglavlje i računa kontrolni zbroj zajedno za UDP segment i pseudo-zaglavlje

  ovaj kontrolni zbroj upisuje se u checksum polje

  zatim se segment i pseudo-zaglavlje prosljeđuju na IP

  UDP primatelja dobiva (od IP) UDP segment i pseudo-zaglavlje, piše 0 u checksum polje i računa kontrolni zbroj za segment i pseudo-zaglavlje

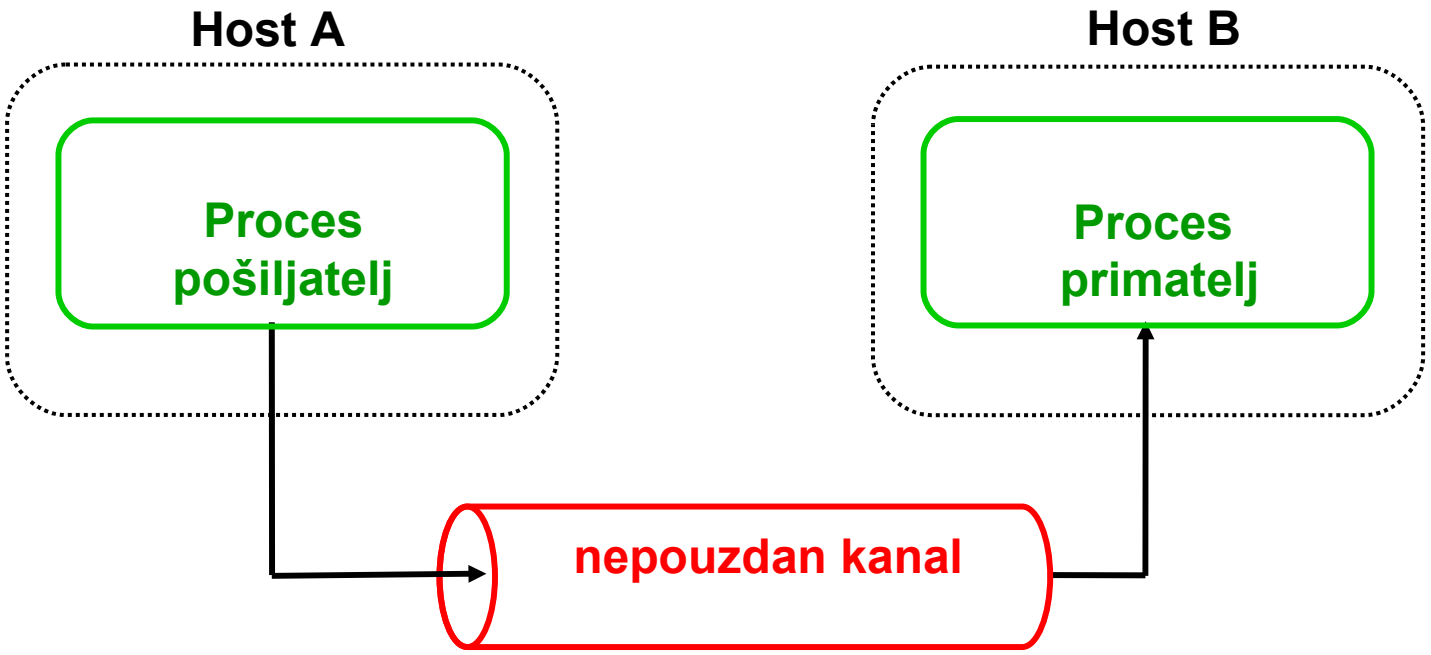
  prednost: provjera kontrolnog zbroja prepoznaje i pogreške u IP adresama, npr. krivo proslijeđene segmente

  nedostatak: povreda principa uslojavanja

Transportni sloj

- Uvod
- UDP
- Kontrola pogrešaka

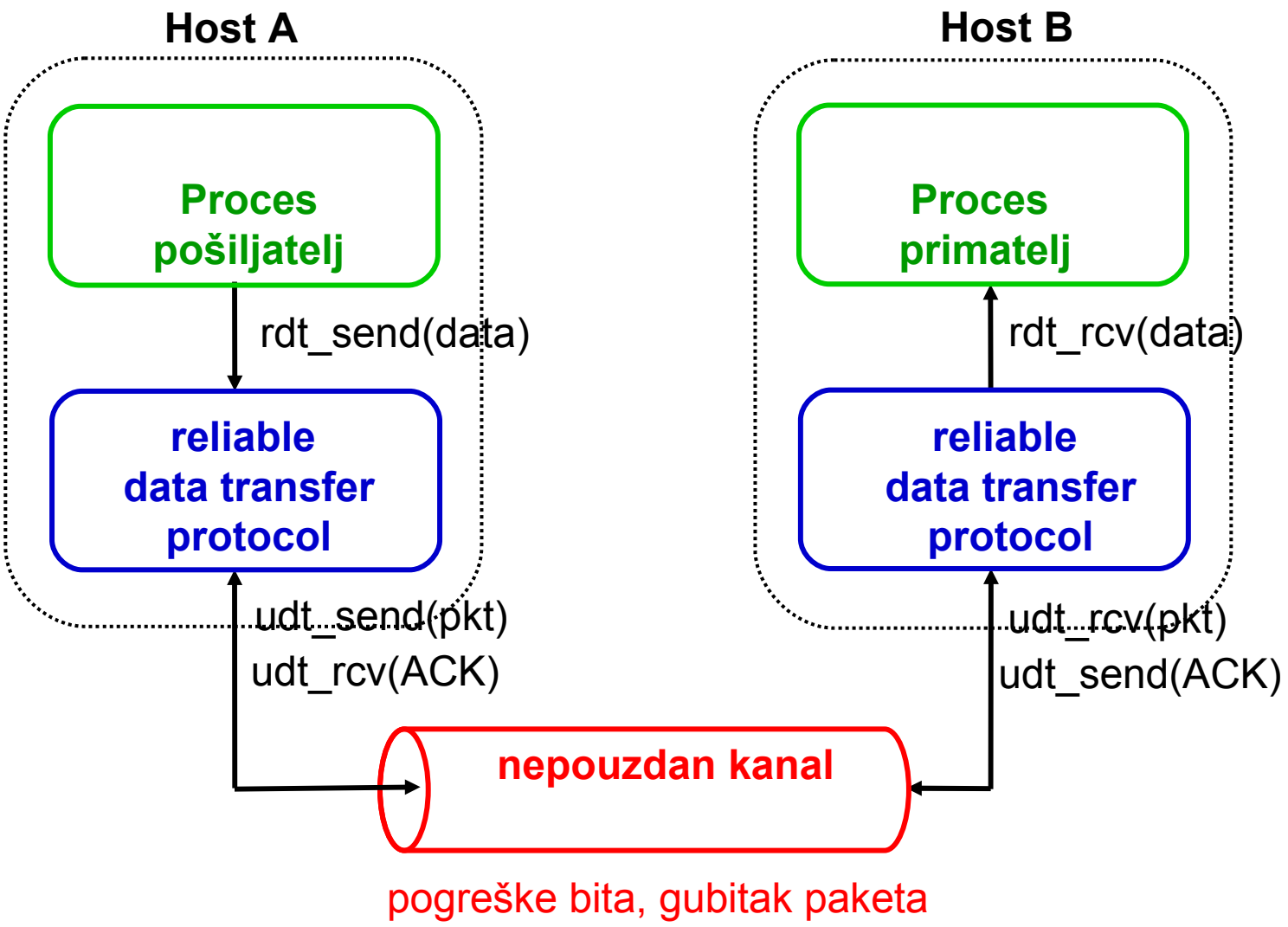
Kontrola pogrešaka



šum, *buffer overflow*, ispadi komponenti uzrokuju pogreške bita i gubitak paketa

rješava se protokolima s prepoznavanjem pogrešaka, potvrdama i ponavljanjem slanja

Kontrola pogrešaka



Kontrola pogrešaka

- 3 osnovna protokola za pouzdan transport:

  Stop-and-Wait

  - pošiljatelj dodaje – u svrhu prepoznavanja pogreške – kontrolni zbroj ili Cyclic Redundancy Check (CRC)
  - primatelj šalje potvrdu (acknowledgment, ACK)
  - nakon timeout-a (= potvrda nije stigla) paket se ponovo šalje
  - za prepoznavanje mogućih duplikata potrebni su redni brojevi (SQN – sequence number)

  Protokoli kliznog prozora (sliding window protocols)

  - šalje se više paketa odjednom kako bi se “popunio” kanal
  - Go-Back-N i Selective Repeat
  - razlikuju se s obzirom na *timeout*, potvrde, ponovno slanje

Transportni sloj

- Uvod
- UDP
- Kontrola pogrešaka

Stop-and-Wait

- neformalan opis:

Ponašanje pošiljatelja

1. šalji paket s aktualnim SQN i uključi *timer*
2. ako se ACK vrati bez pogreške bita i s aktualnim SQN prije isteka timeout-a, inkrementiraj SQN i vrati se na 1. korak
3. ako je timeout istekao, ponovo šalji paket, također ponovo uključi *timer* i vrati se na 2. korak

Ponašanje primatelja

- ako je paket primljen bez pogreške bita i s aktualnim SQN, šalji ACK s aktualnim SQN i inkrementiraj SQN; inače ponovo šalji posljednji ACK

Stop-and-Wait

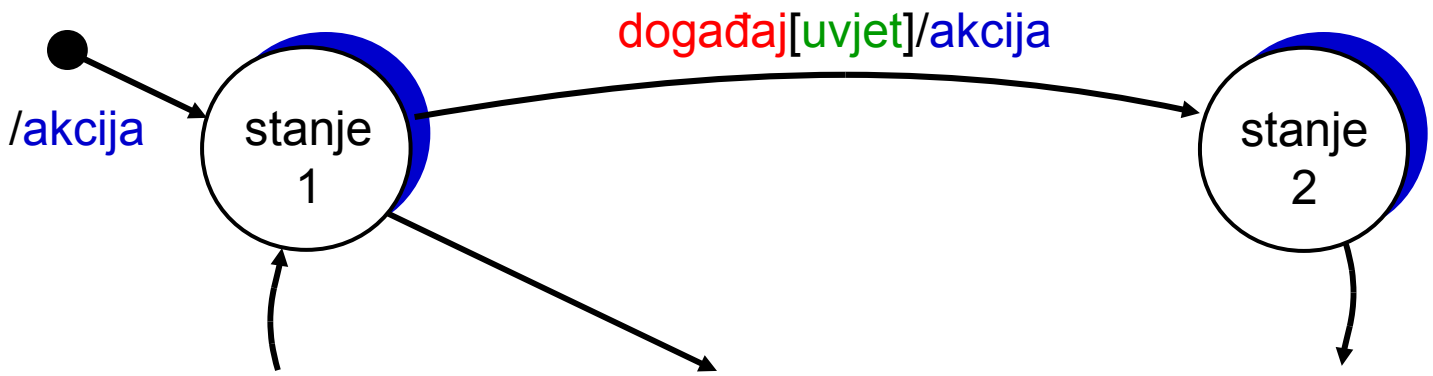
- Opis pomoću UML *Statecharts*

  Statechart se uvijek nalazi u nekom stanju (*state*), crno označena točka predstavlja početno stanje (*initial state*)

  Prijelaz između stanja (*state transition*) se ostvaruje nekim događajem (*event*) i ispunjavanjem nekog uvjeta (*guard*)

  Nakon prijelaza u novo stanje izvodi se neka akcija (*action*)

  Iz praktičnih razloga moguće je uvesti i varijable



Stop-and-Wait

- Napomene vezane uz *Statecharts*

  Statecharts predstavljaju varijantu konačnih automata

  Događaji, uvjeti i akcije se često opisuju kroz pseudokôd (time dobivamo tzv. “poluformalan” opis)

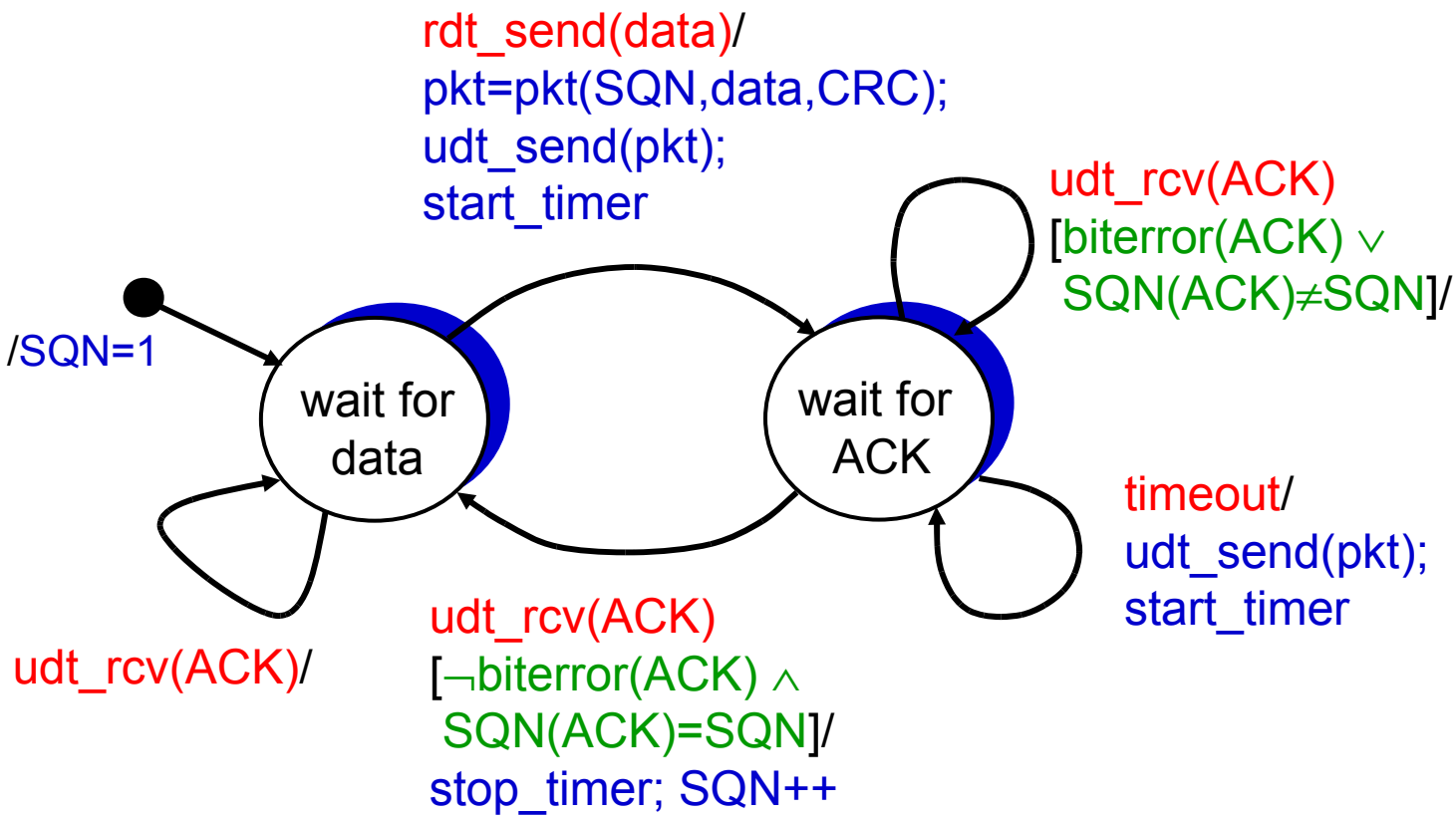
  Ponašanje protokola često se modelira ovakvim (ili sličnim) automatima

  Postoje programski alati koji takvo modeliranje podržavaju: protokoli se mogu specificirati kao automati iz čega se može generirati kôd; na osnovu toga moguće je izvoditi različite analize, simulacije i testiranja

  Ovdje se koriste *statecharts* za precizan opis protokola *Stop-and-Wait* (kasnije i ostalih protokola)

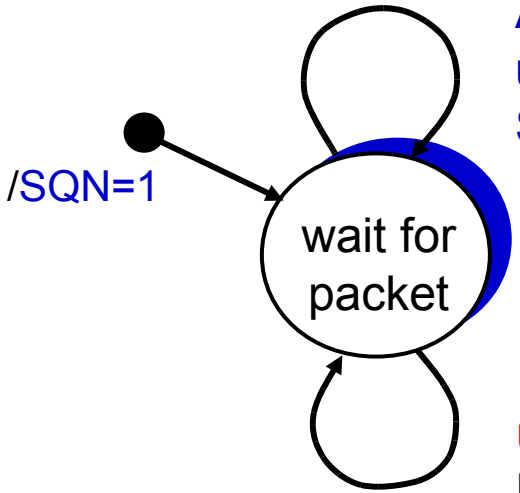
Stop-and-Wait

- Pošiljatelj:



Stop-and-Wait

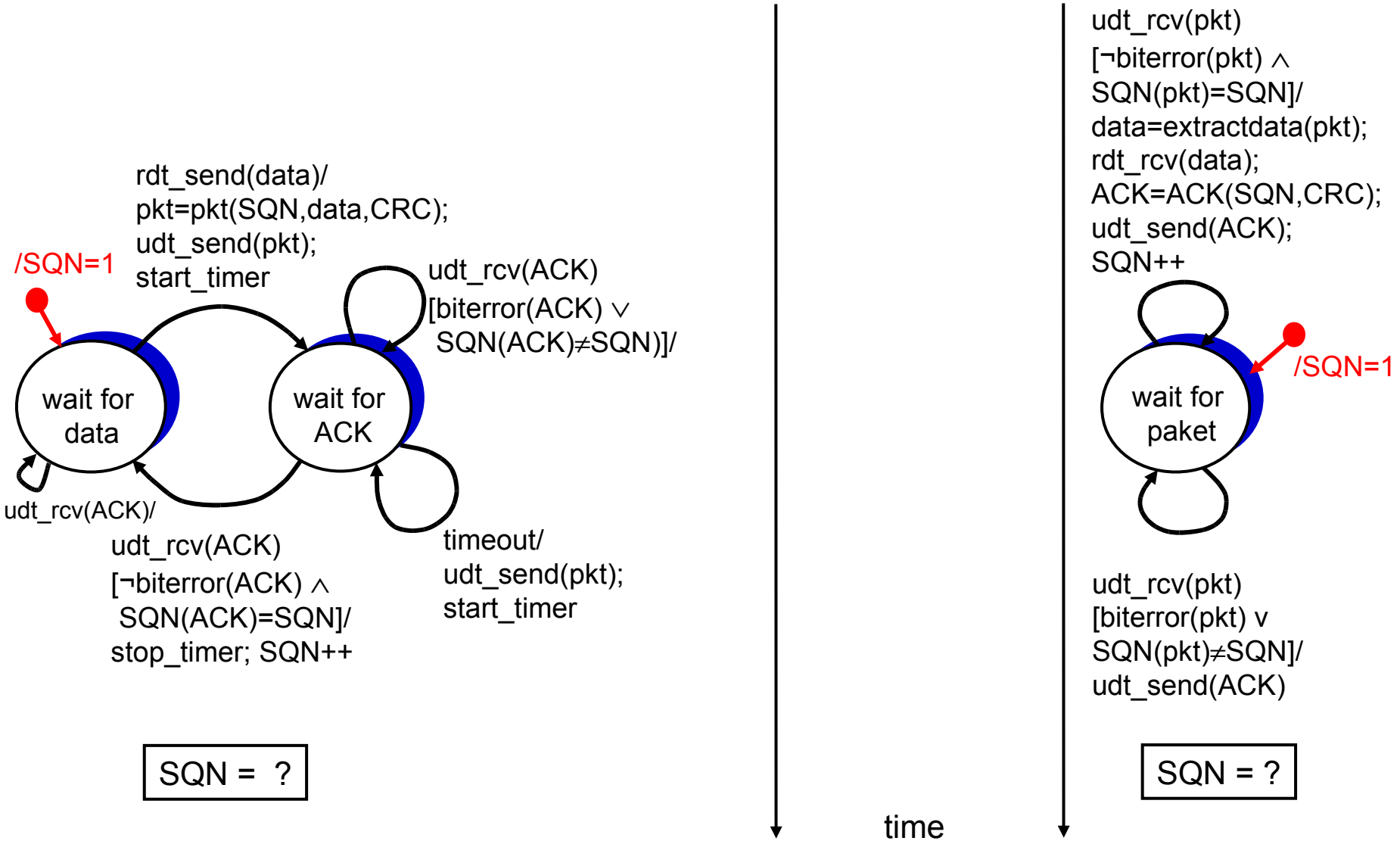
- Primatelj:



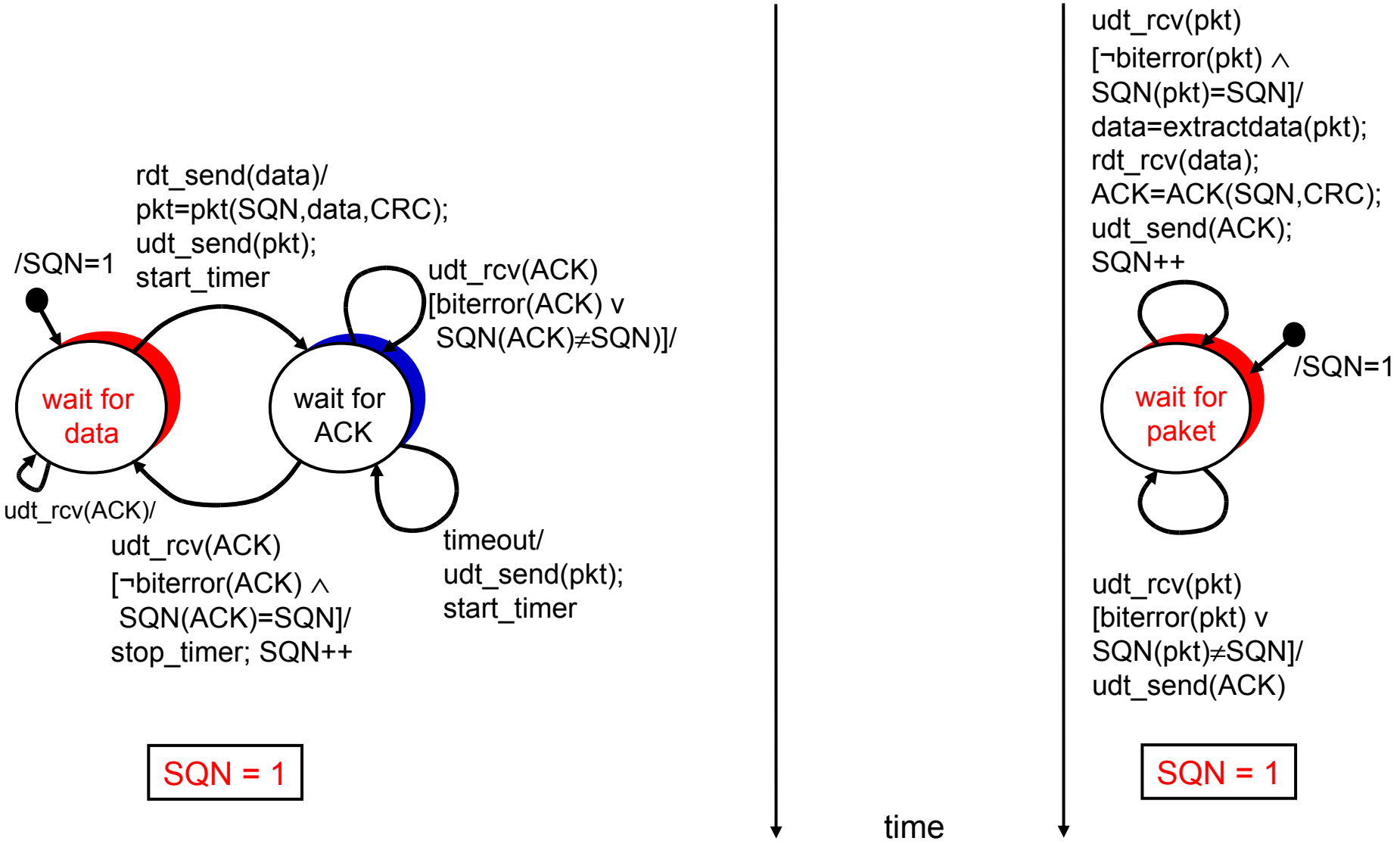
udt_rcv(pkt)
[¬biterror(pkt) ∧ SQN(pkt)=SQN]/
data=extractdata(pkt);
rdt_rcv(data);
ACK=ACK(SQN,CRC);
udt_send(ACK);
SQN++

udt_rcv(pkt)
[biterror(pkt) ∨ SQN(pkt)≠SQN]/
udt_send(ACK)

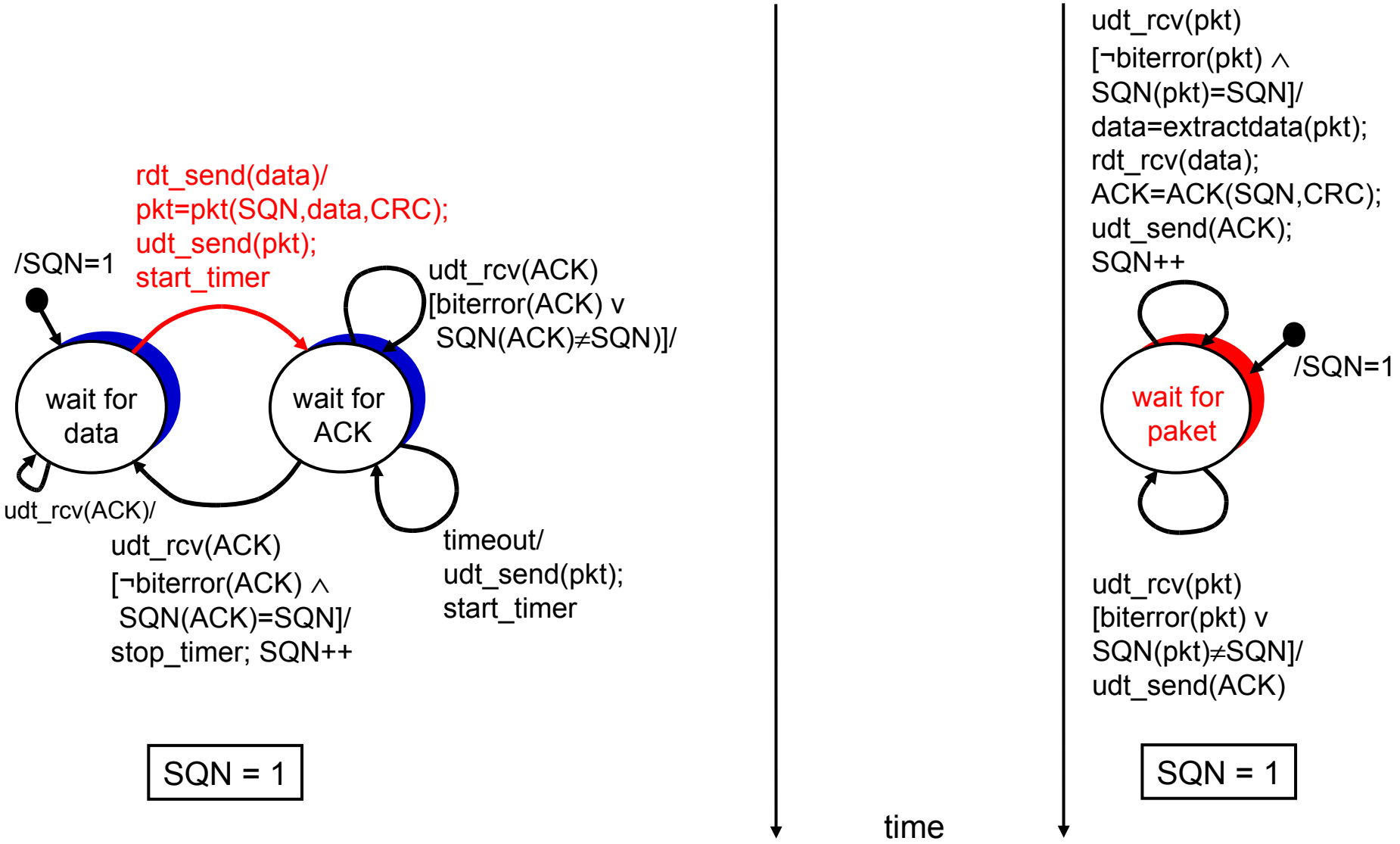
Stop-and-Wait: normalan slijed (*flow*)



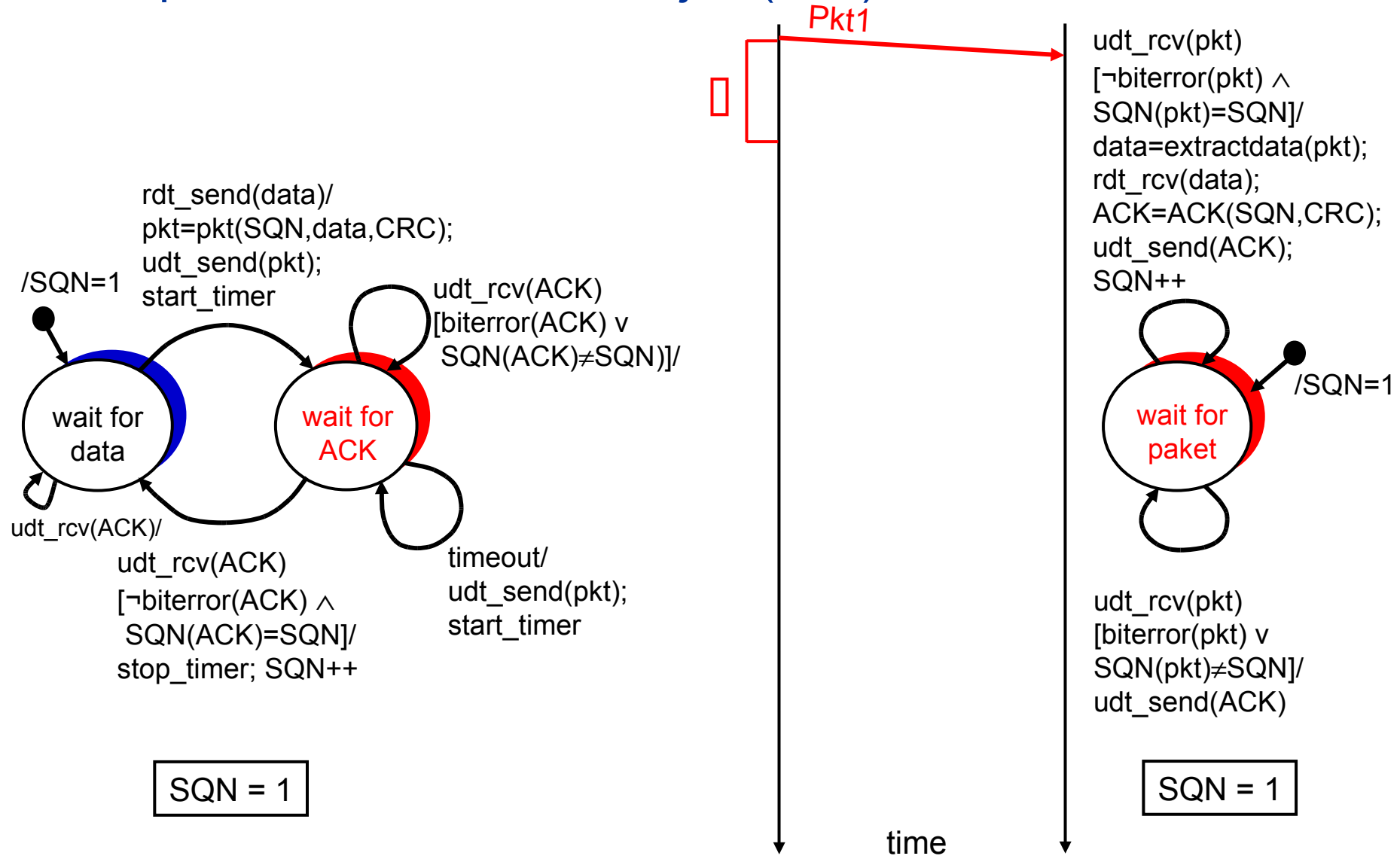
Stop-and-Wait: normalan slijed (*flow*)



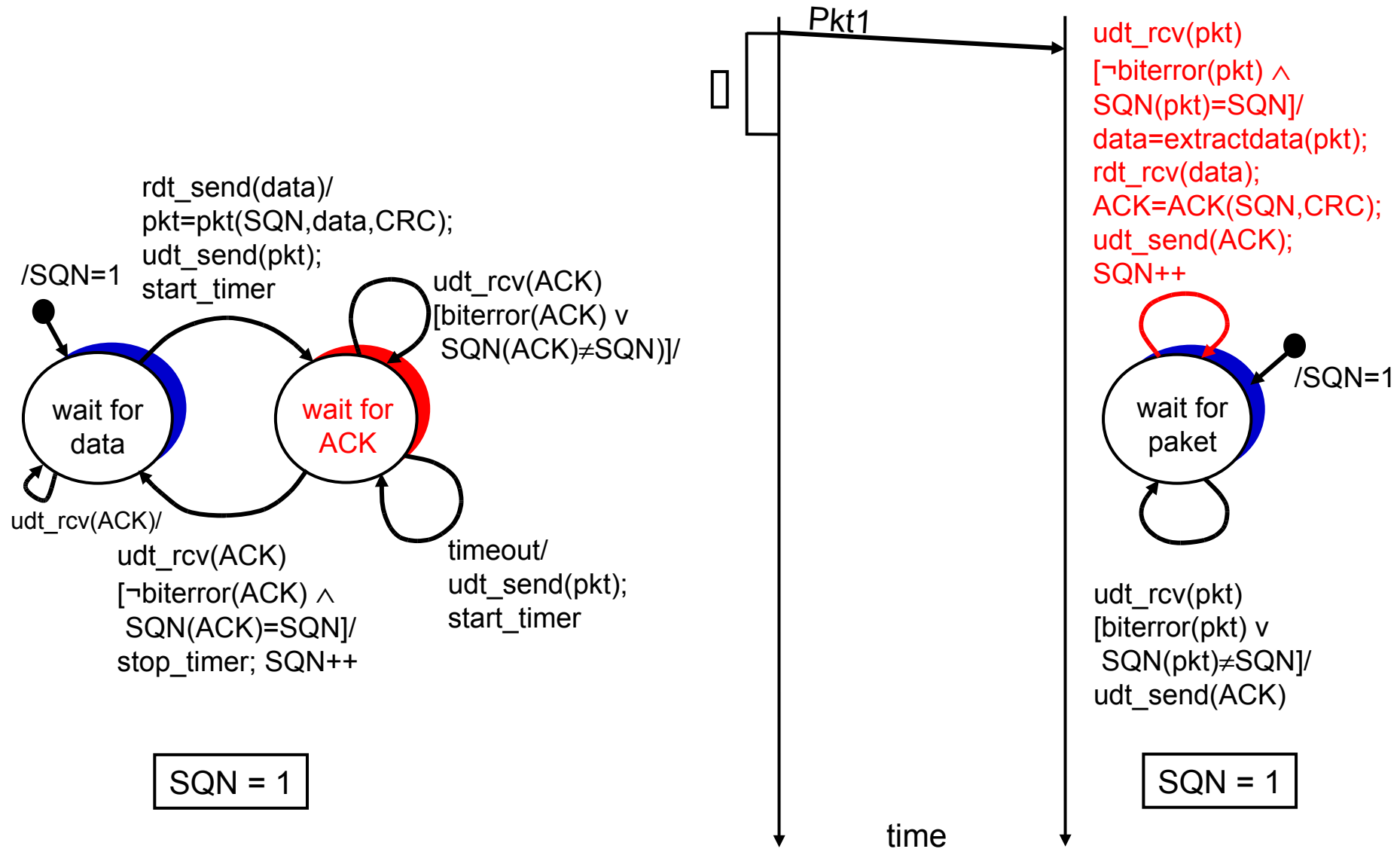
Stop-and-Wait: normalan slijed (*flow*)



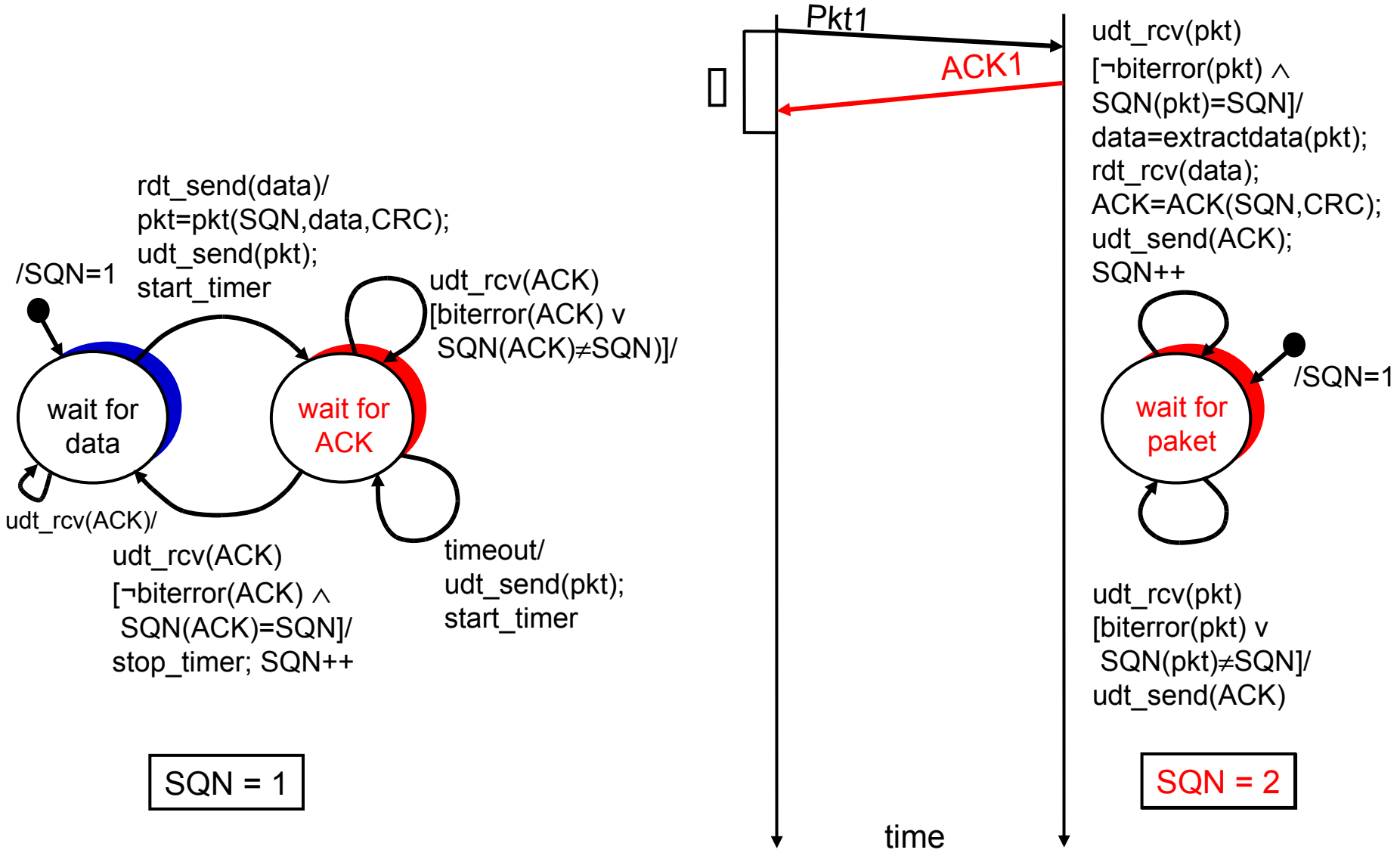
Stop-and-Wait: normalan slijed (*flow*)



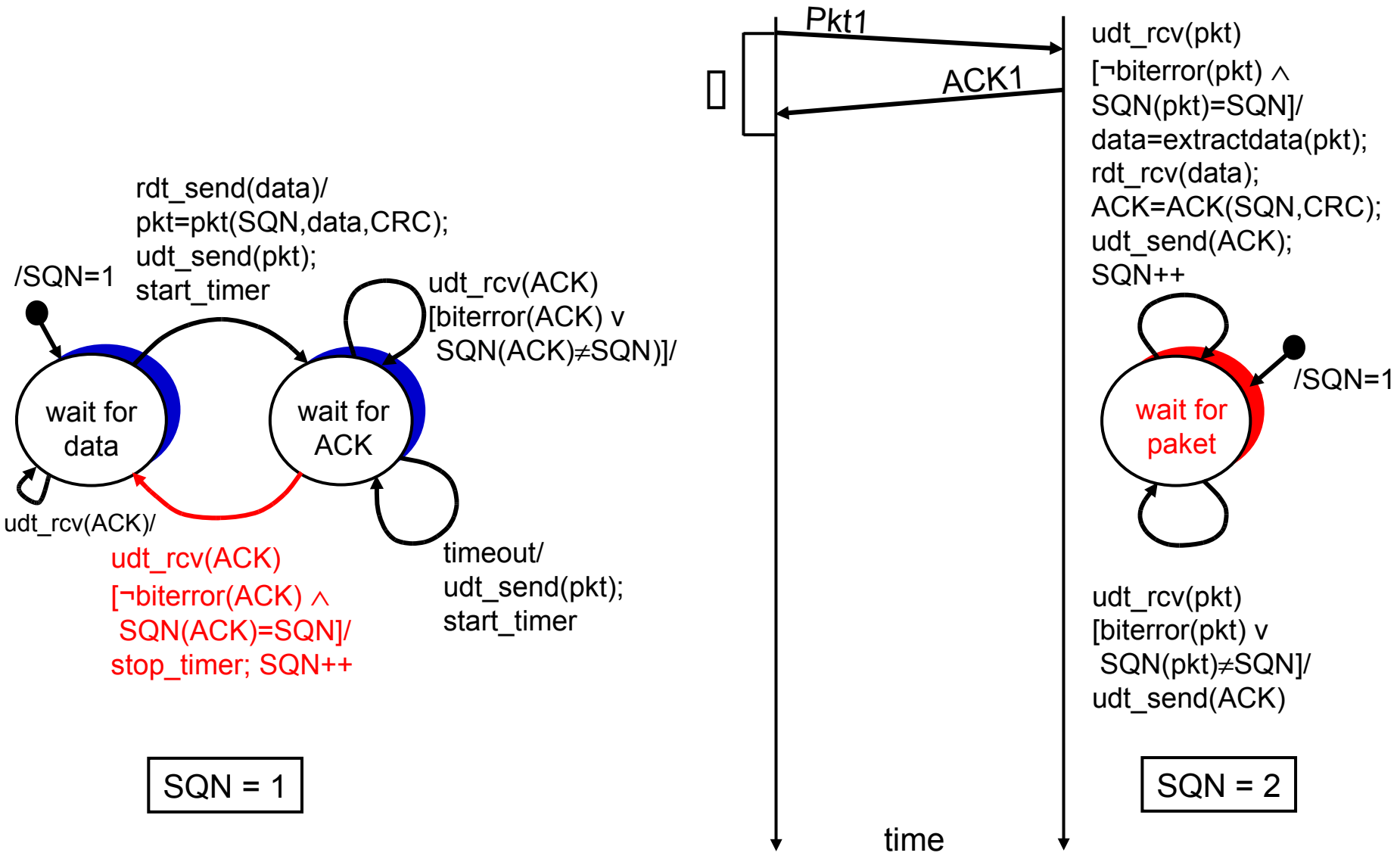
Stop-and-Wait: normaler Ablauf



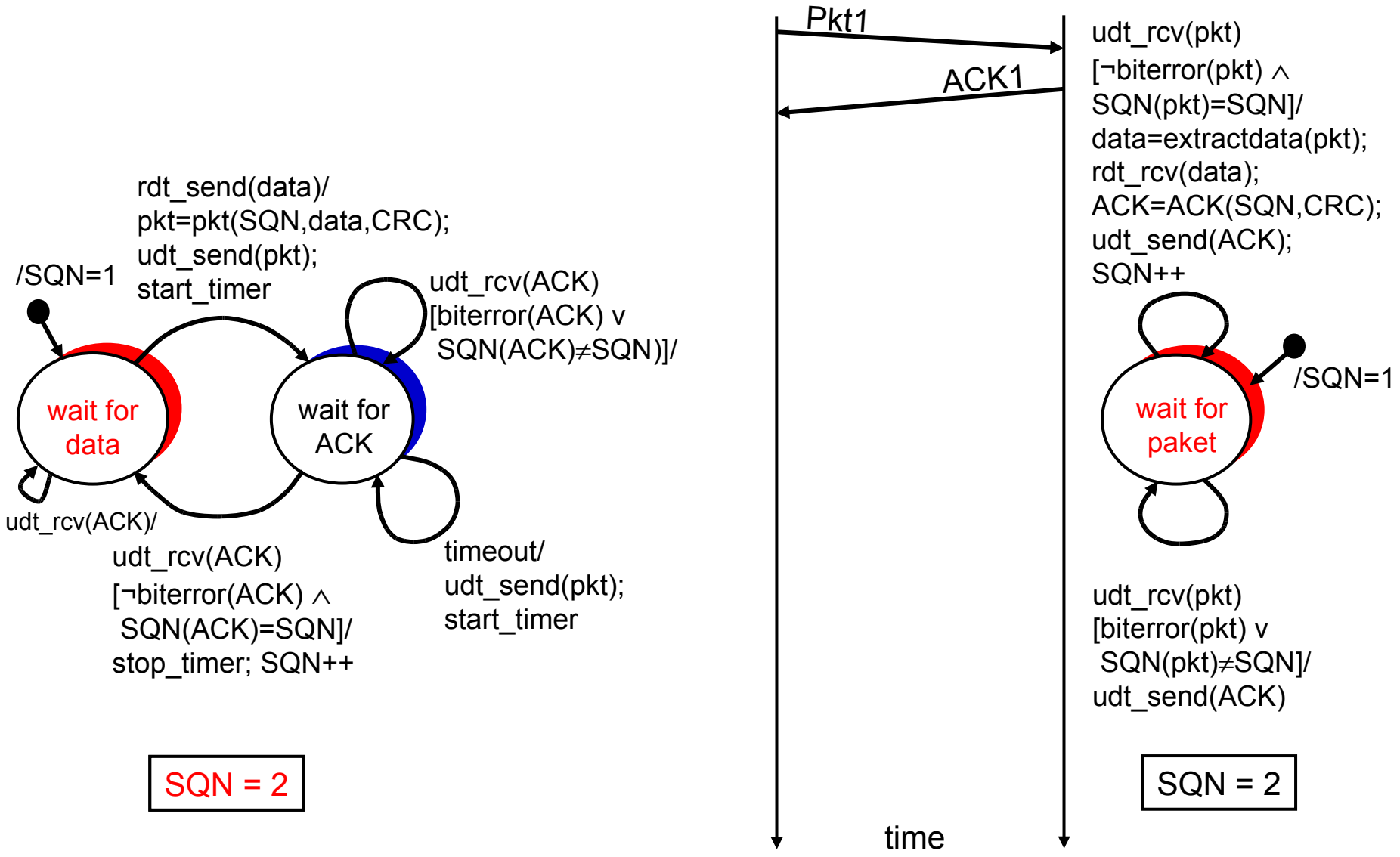
Stop-and-Wait: normalan slijed (*flow*)



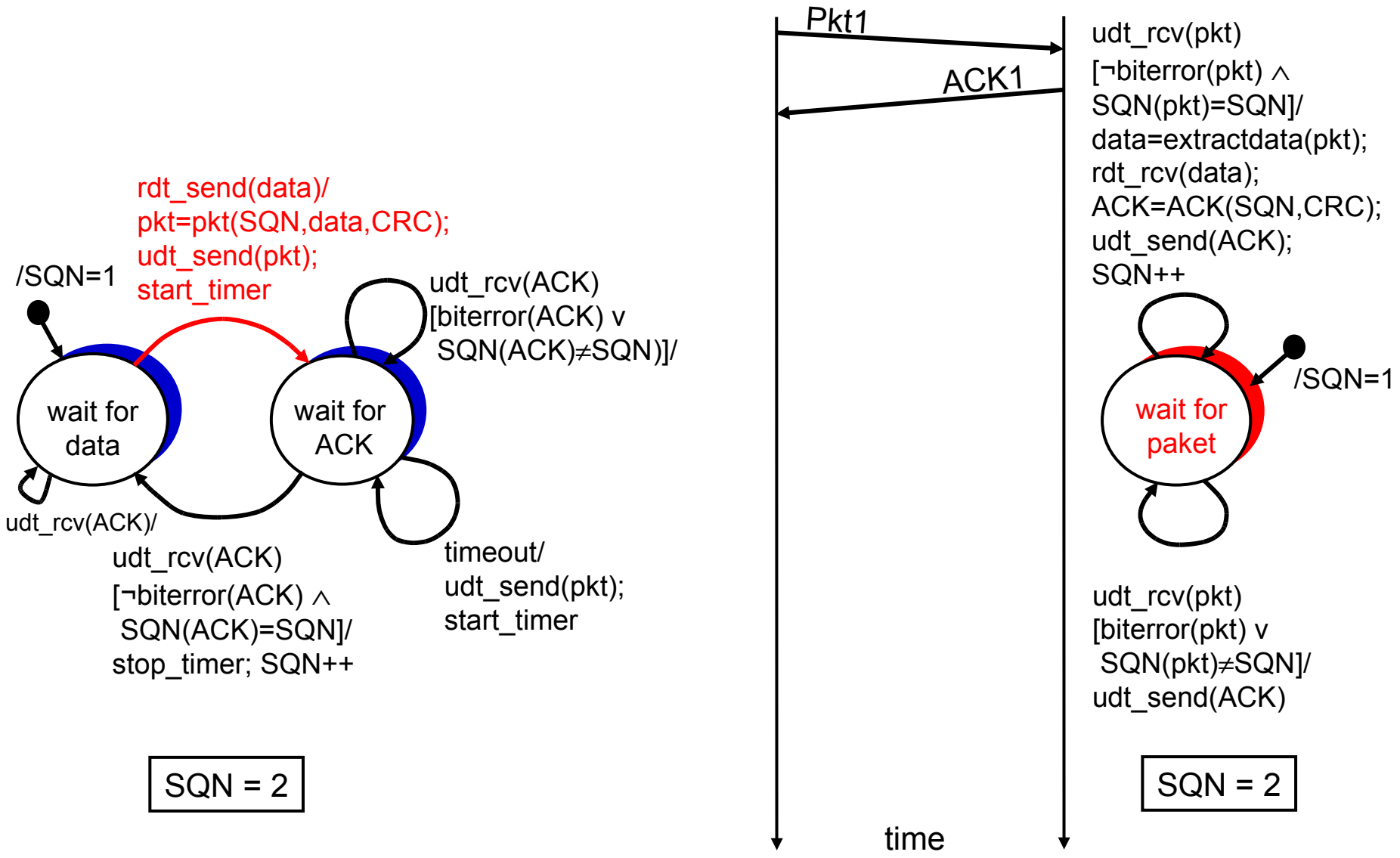
Stop-and-Wait: normalan slijed (*flow*)



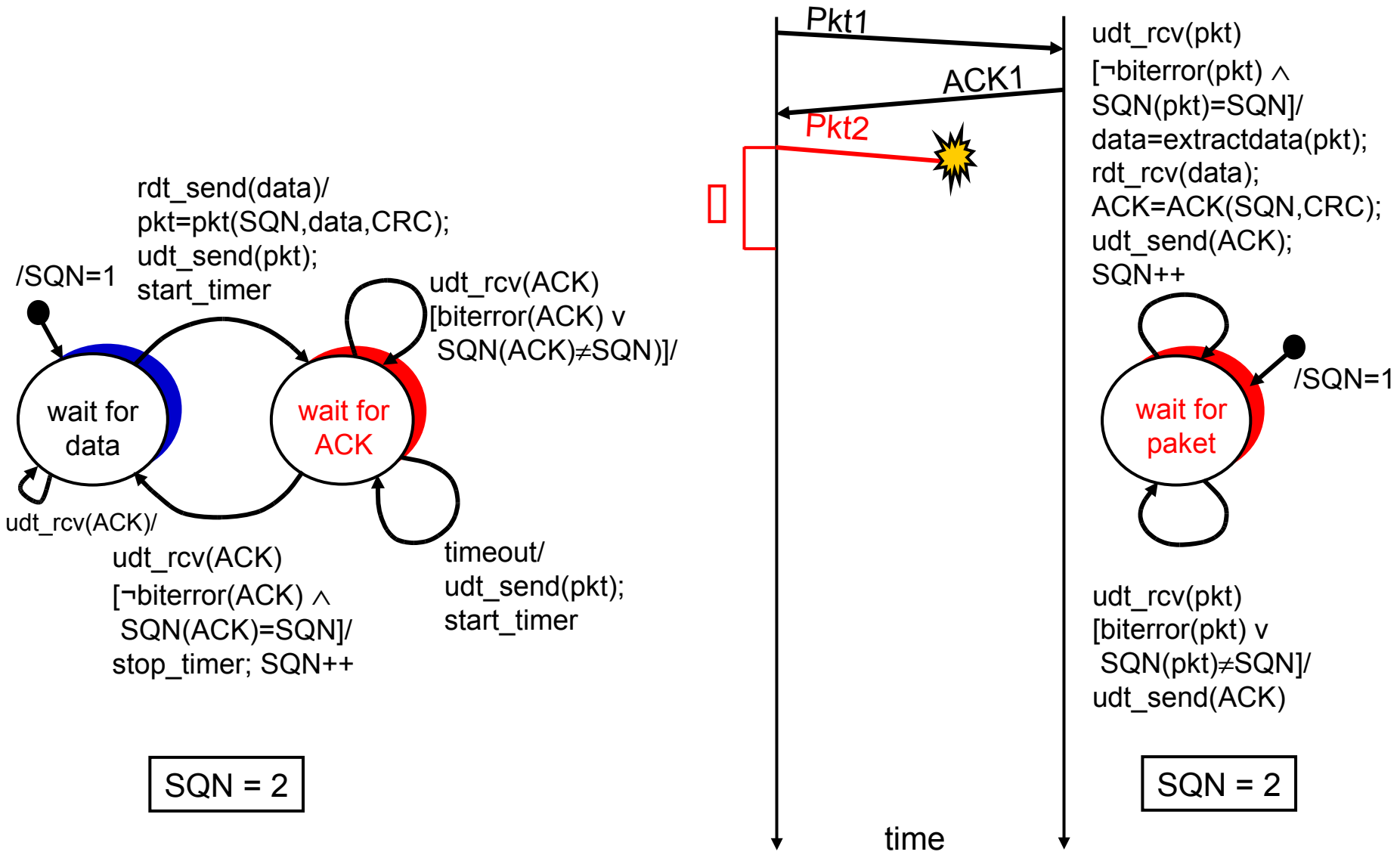
Stop-and-Wait: normalan slijed (*flow*)



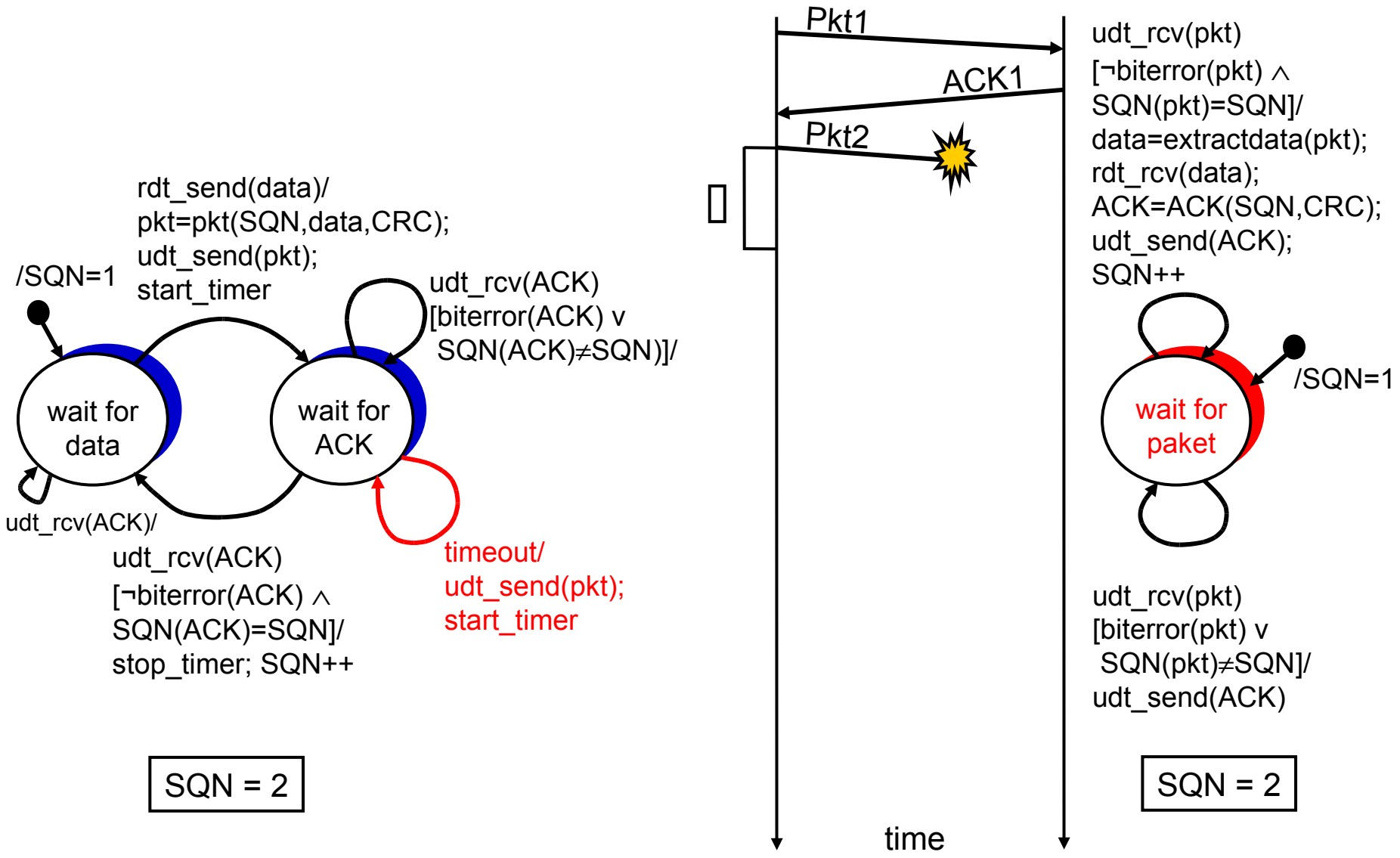
Stop-and-Wait: gubitak paketa



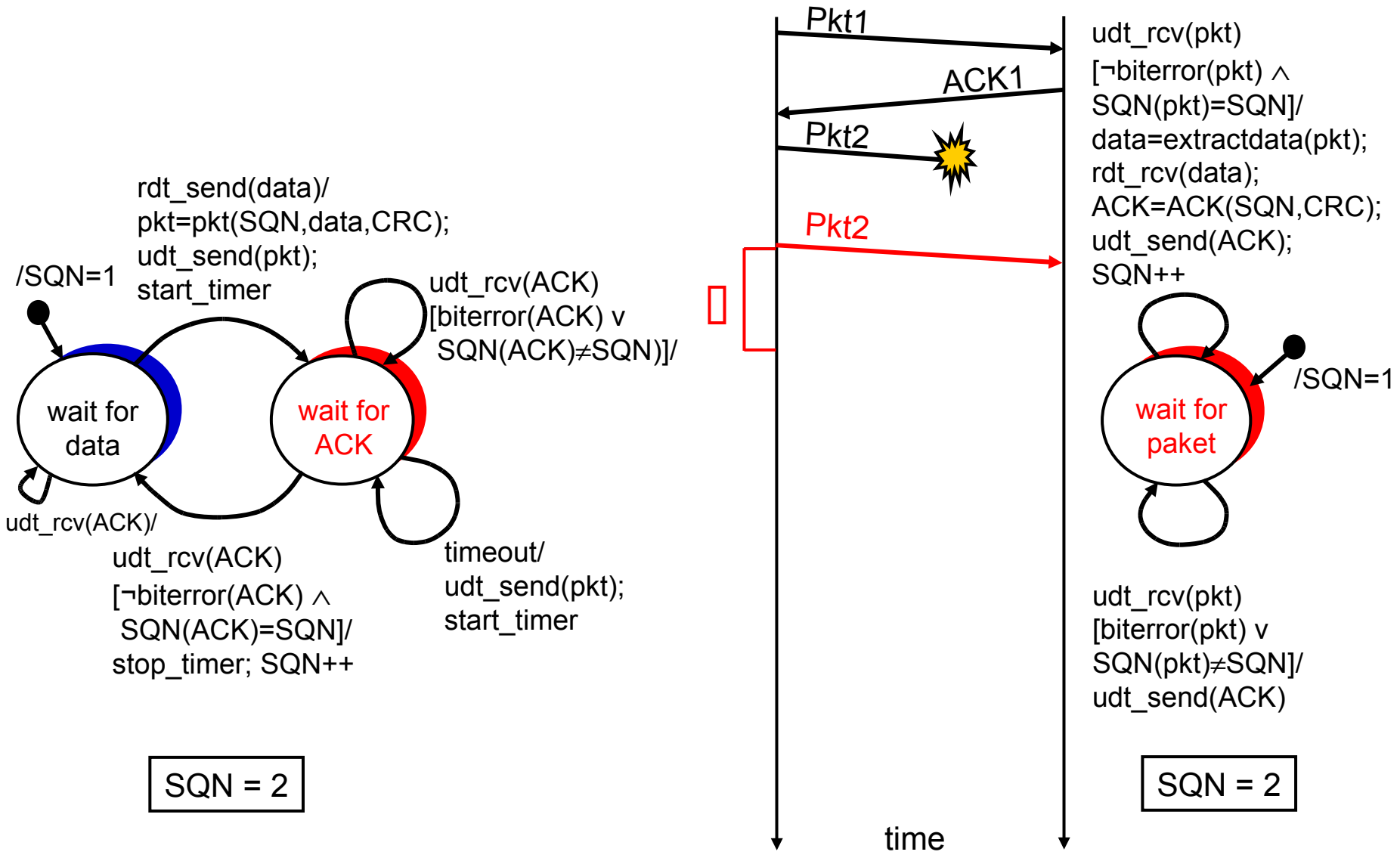
Stop-and-Wait: gubitak paketa



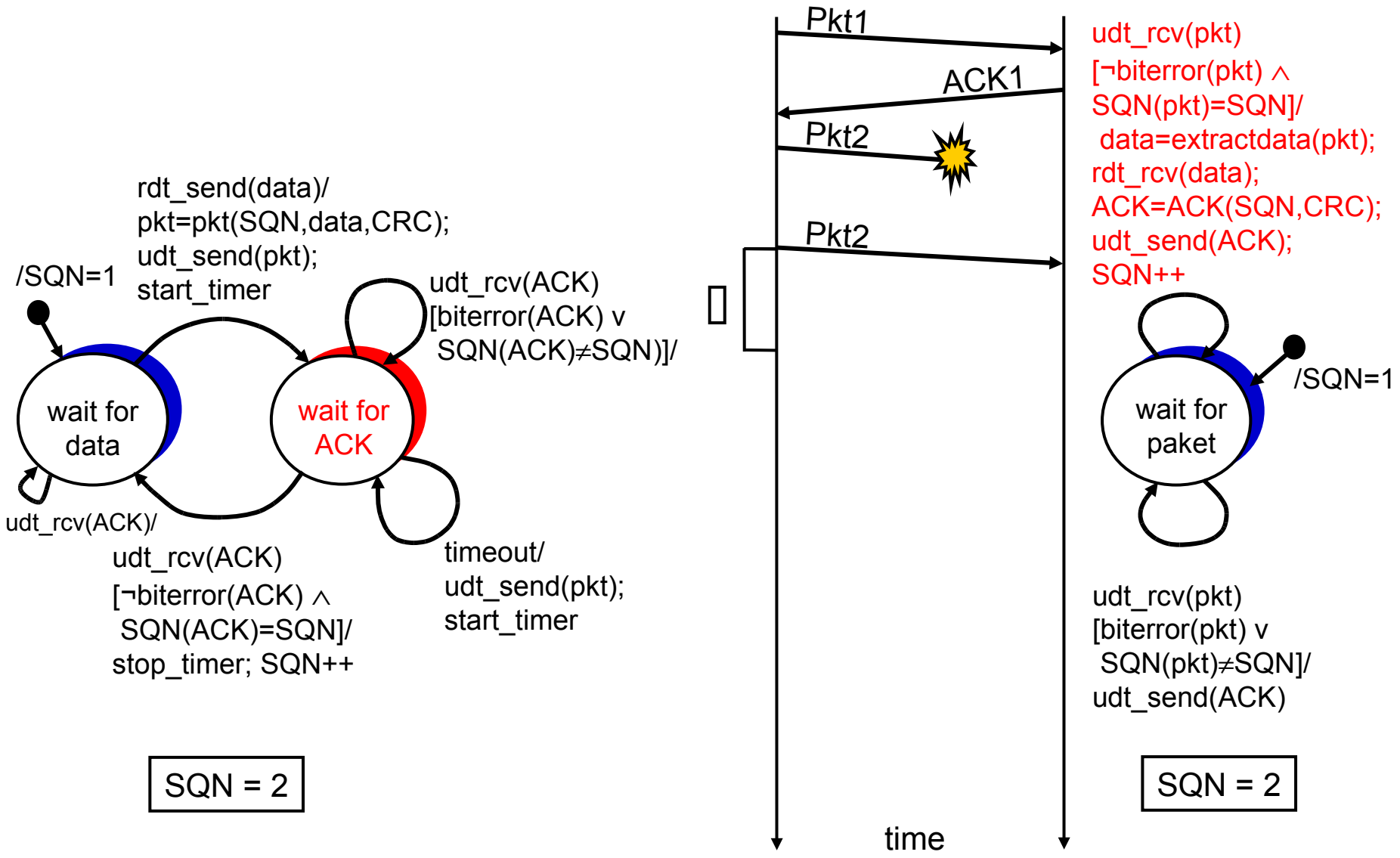
Stop-and-Wait: gubitak paketa



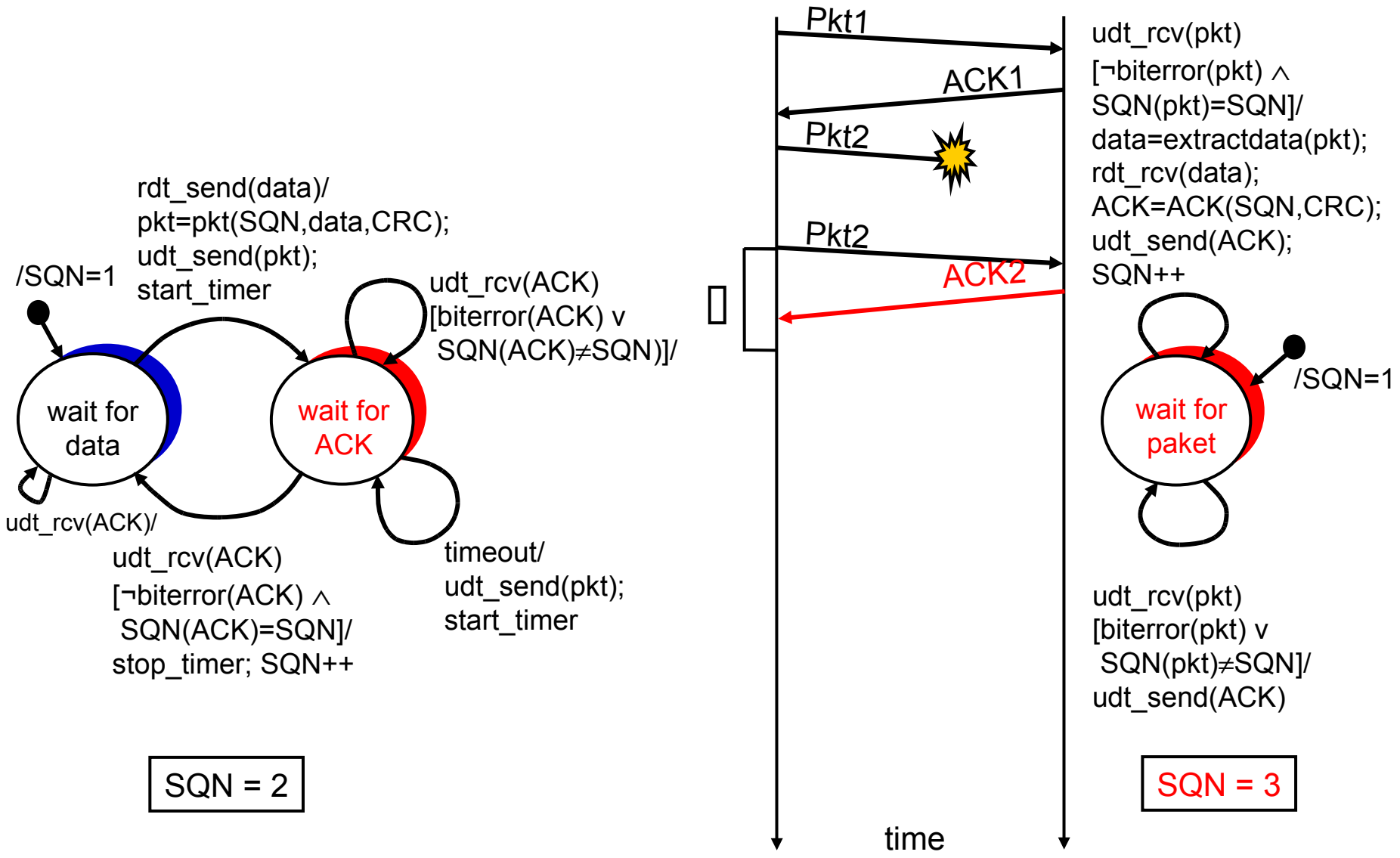
Stop-and-Wait: gubitak paketa



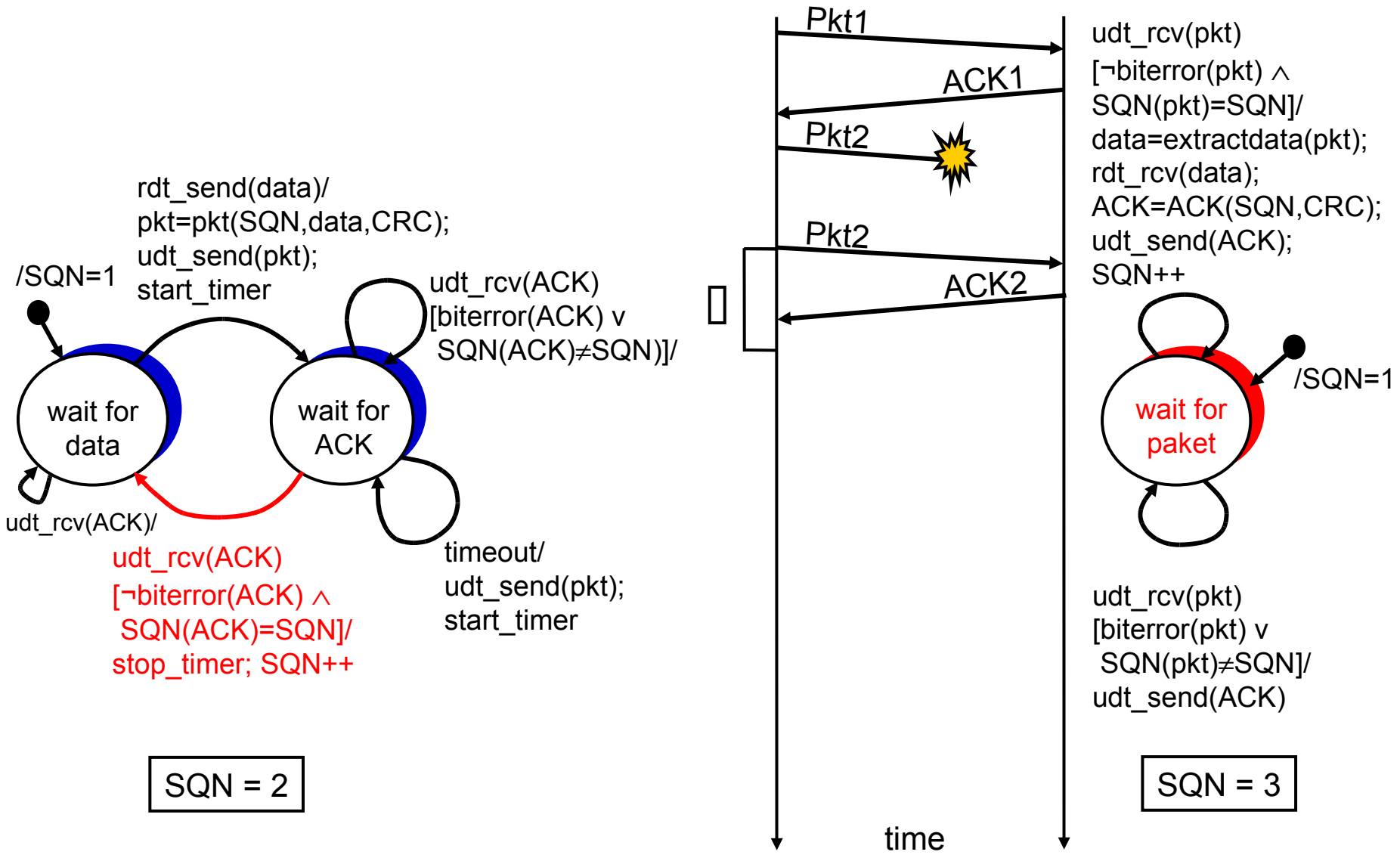
Stop-and-Wait: gubitak paketa



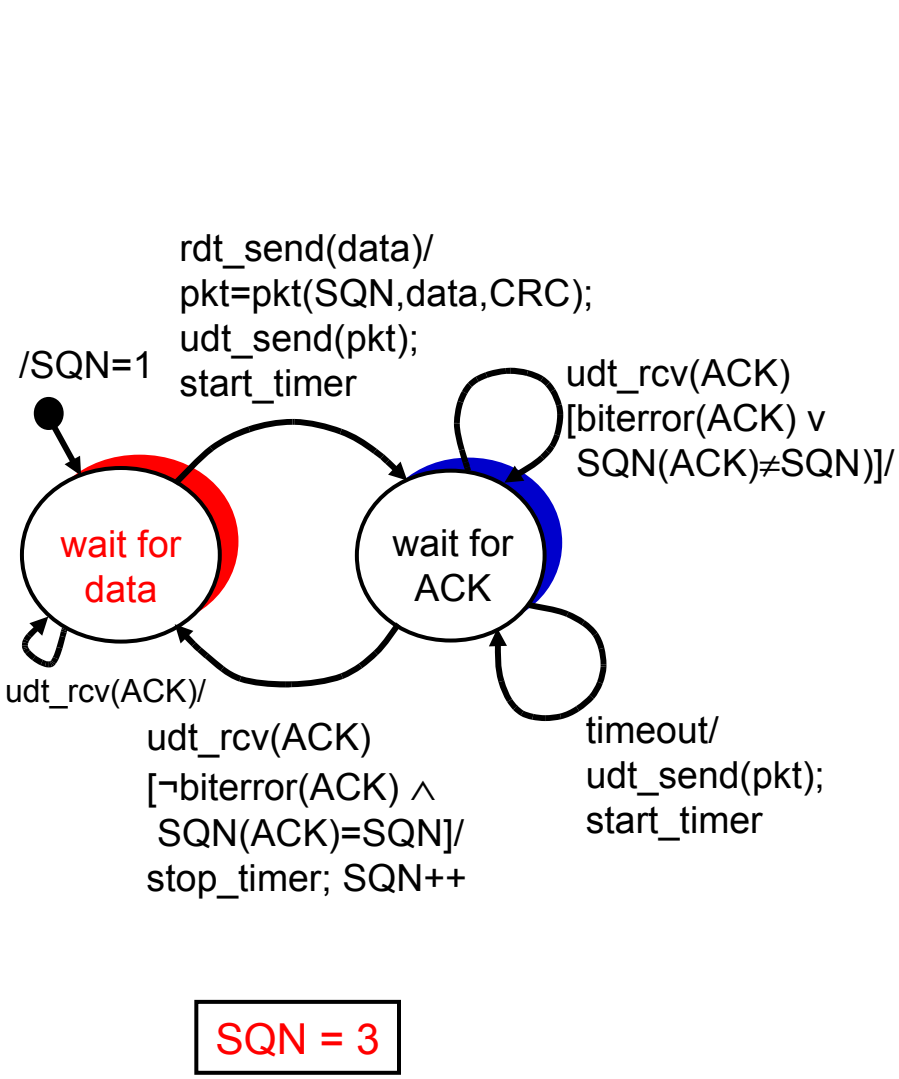
Stop-and-Wait: gubitak paketa

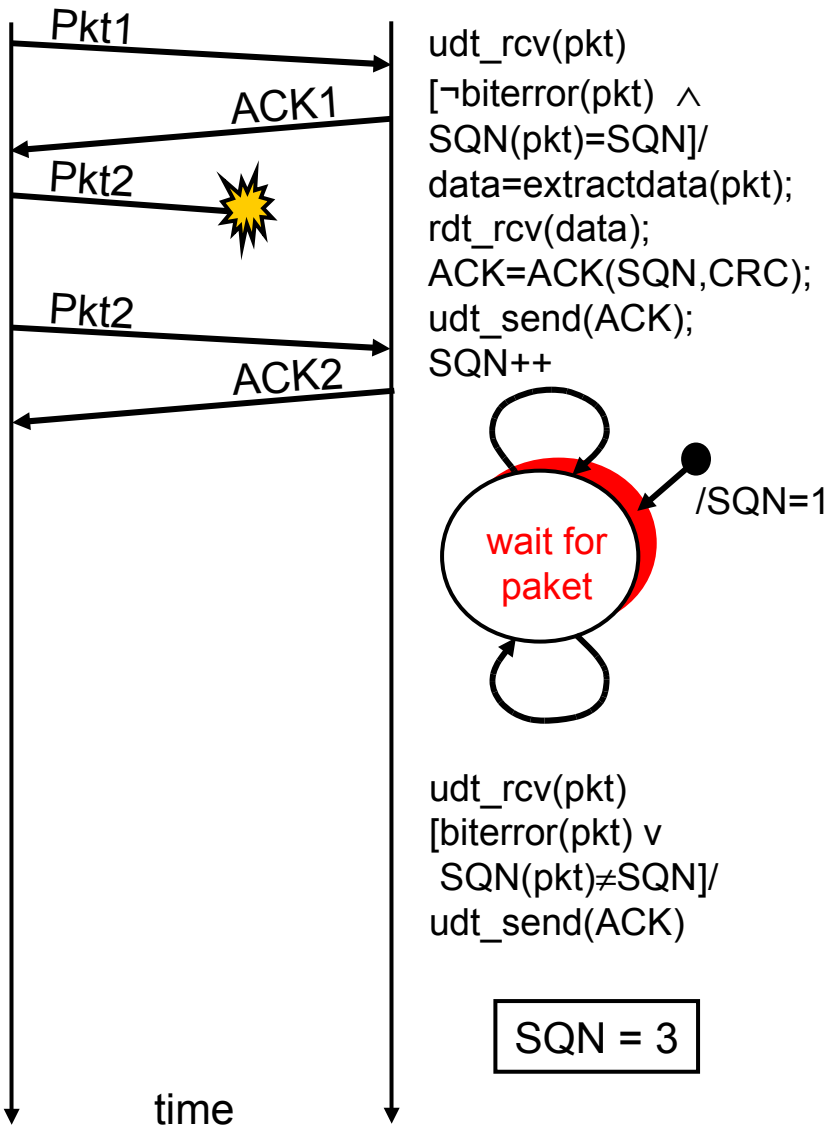


Stop-and-Wait: gubitak paketa

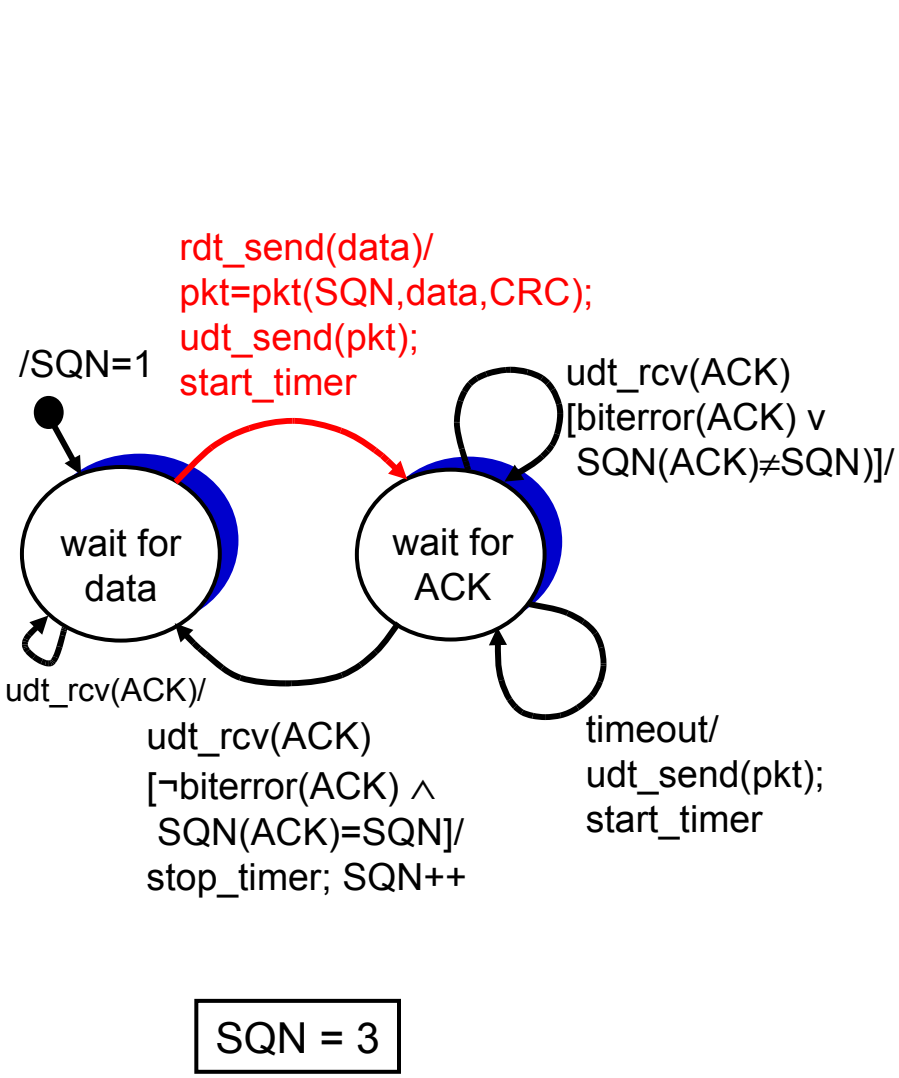


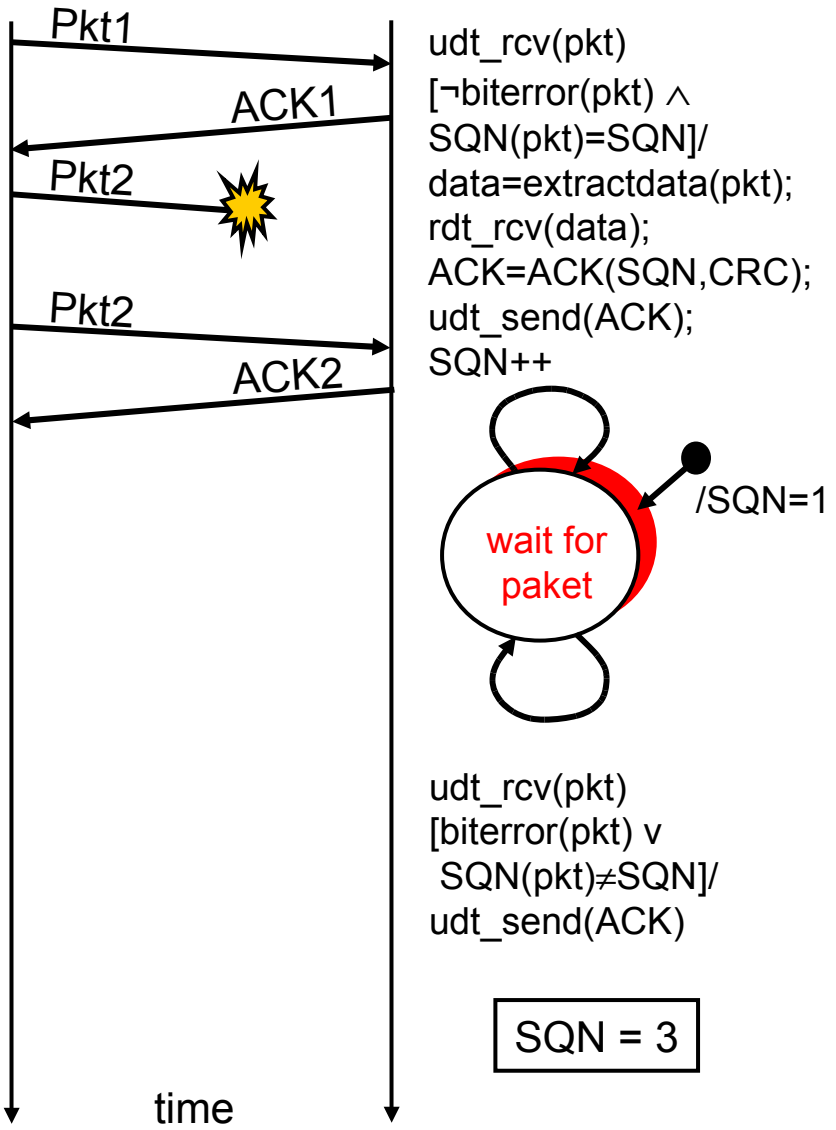
Stop-and-Wait: gubitak paketa



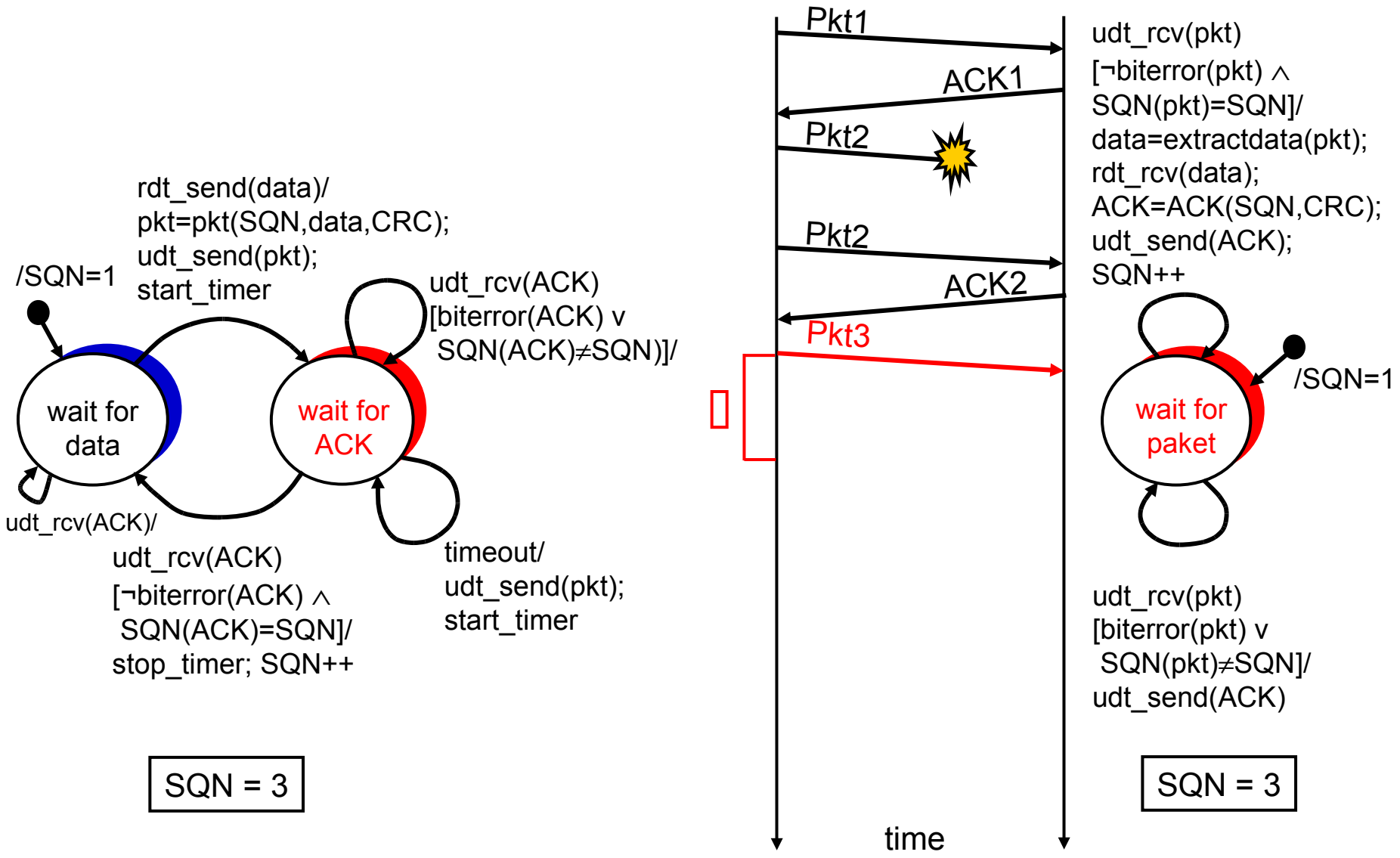


Stop-and-Wait: gubitak potvrde (ACK)

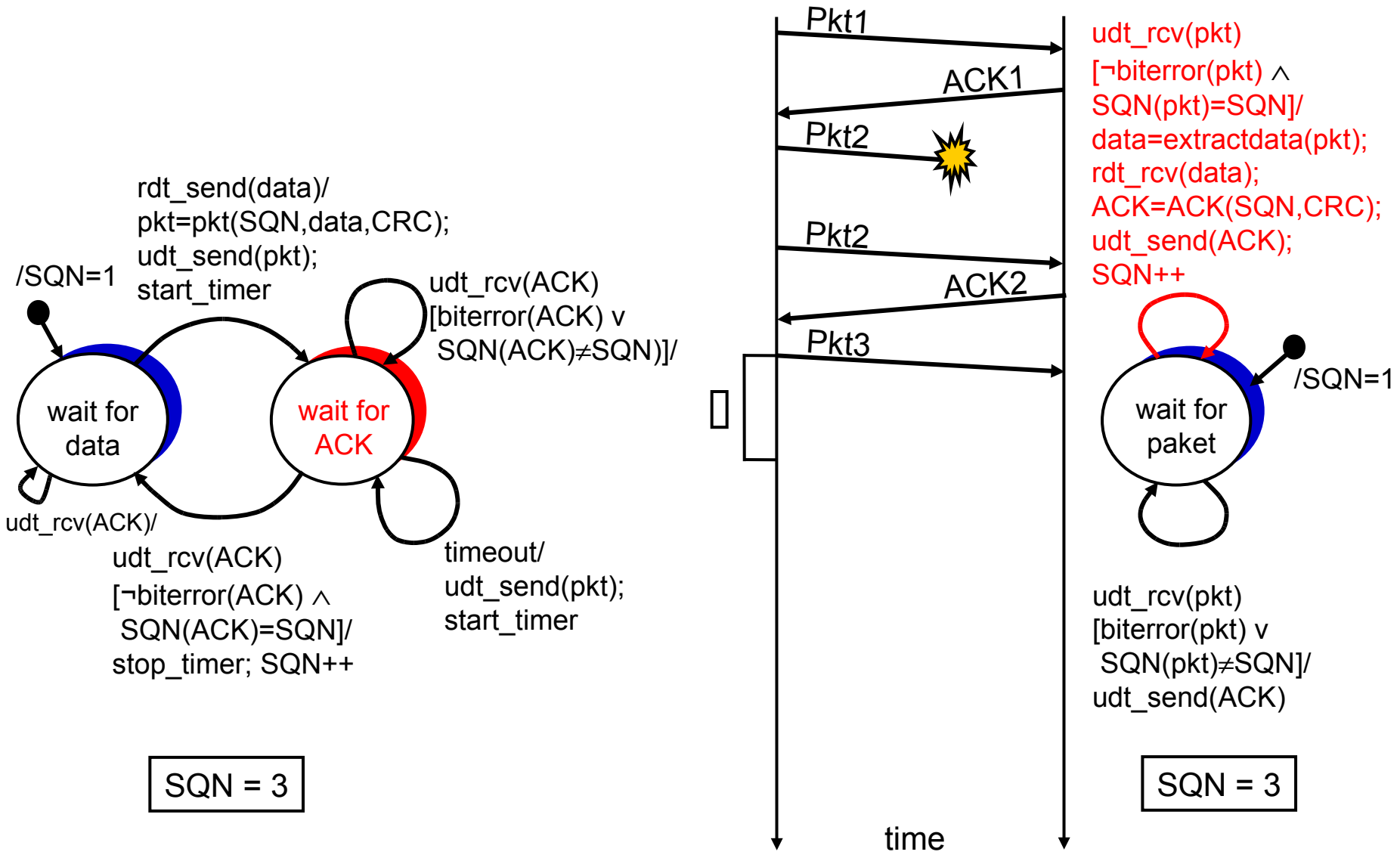




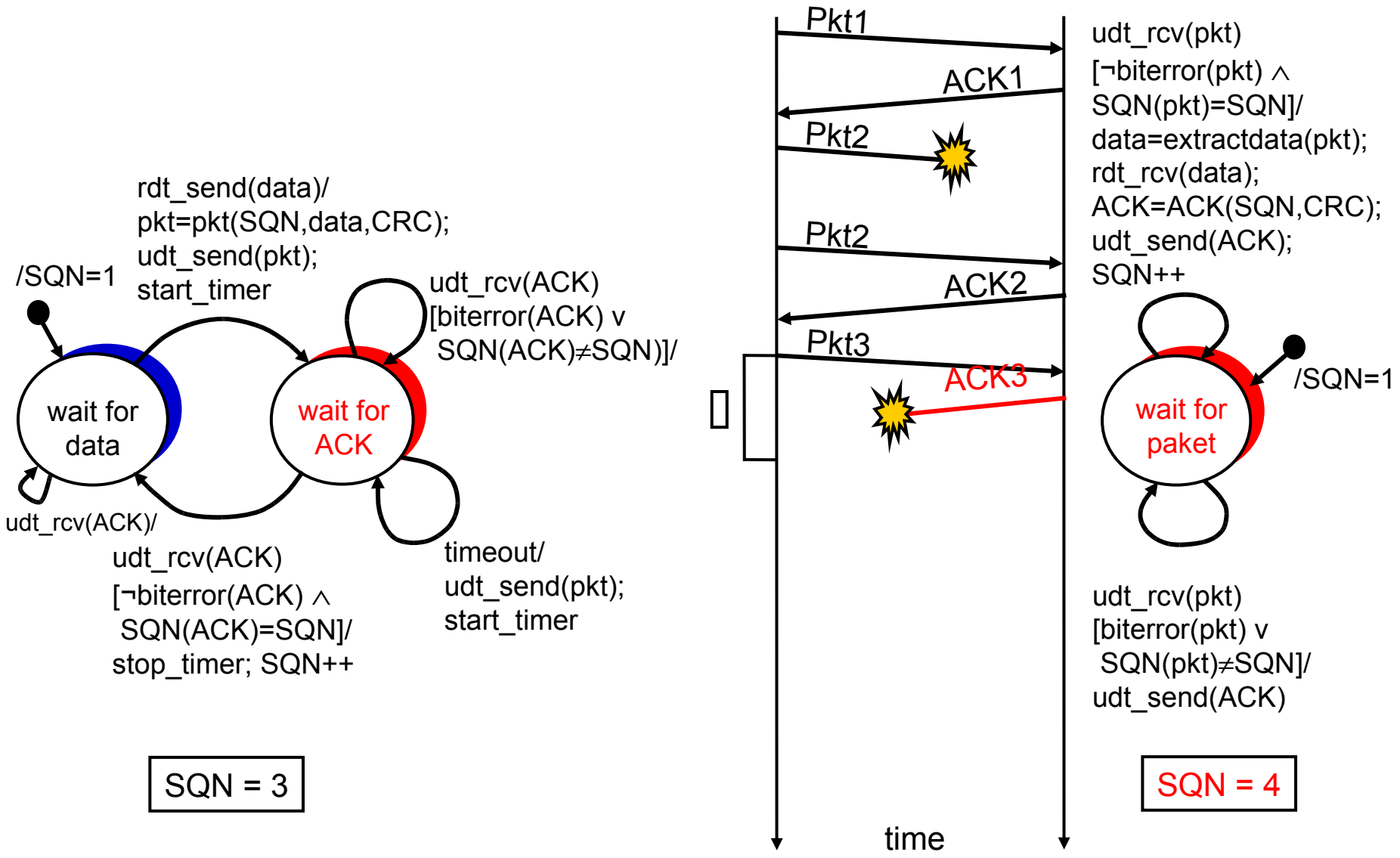
Stop-and-Wait: gubitak potvrde (ACK)



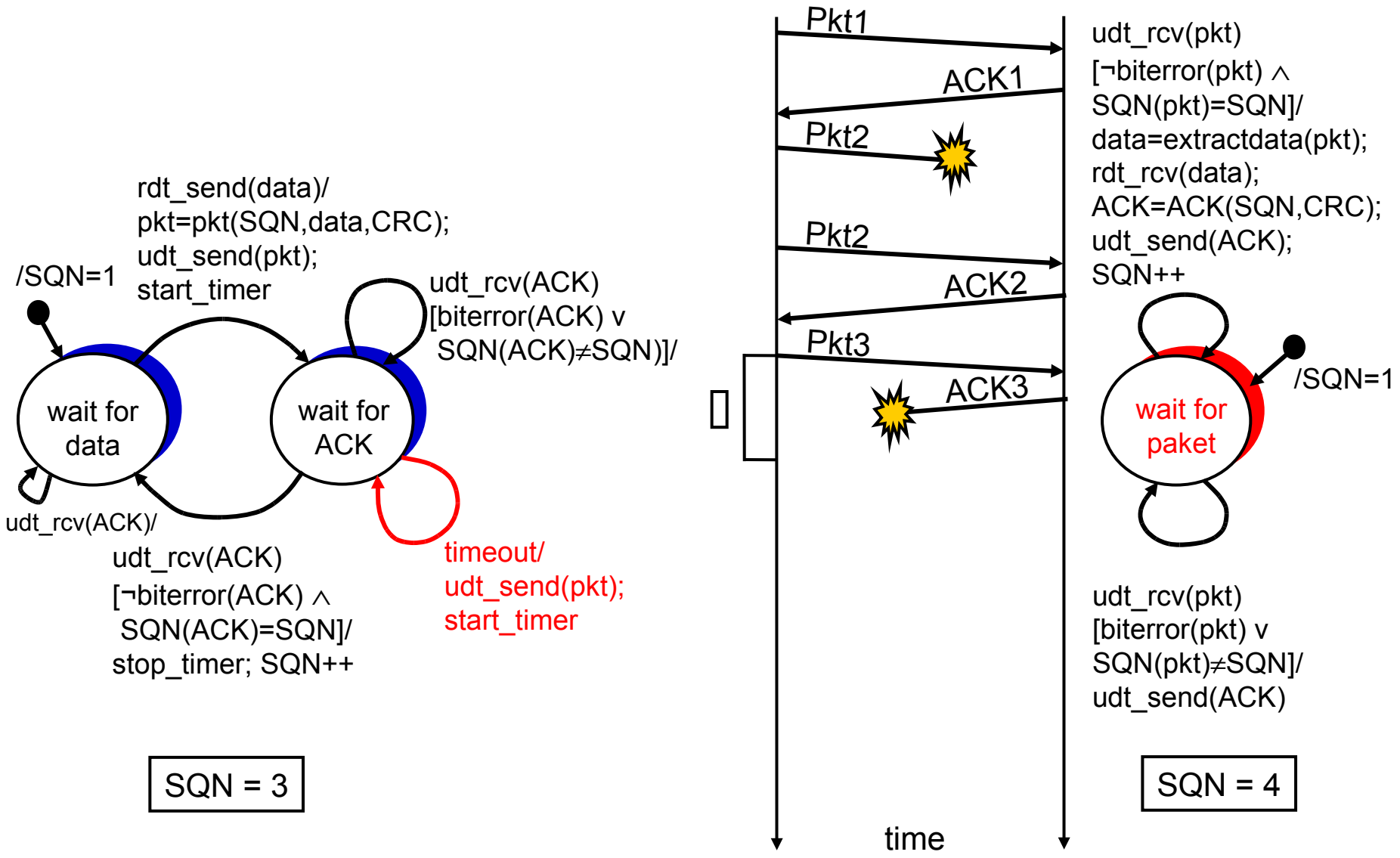
Stop-and-Wait: gubitak potvrde (ACK)



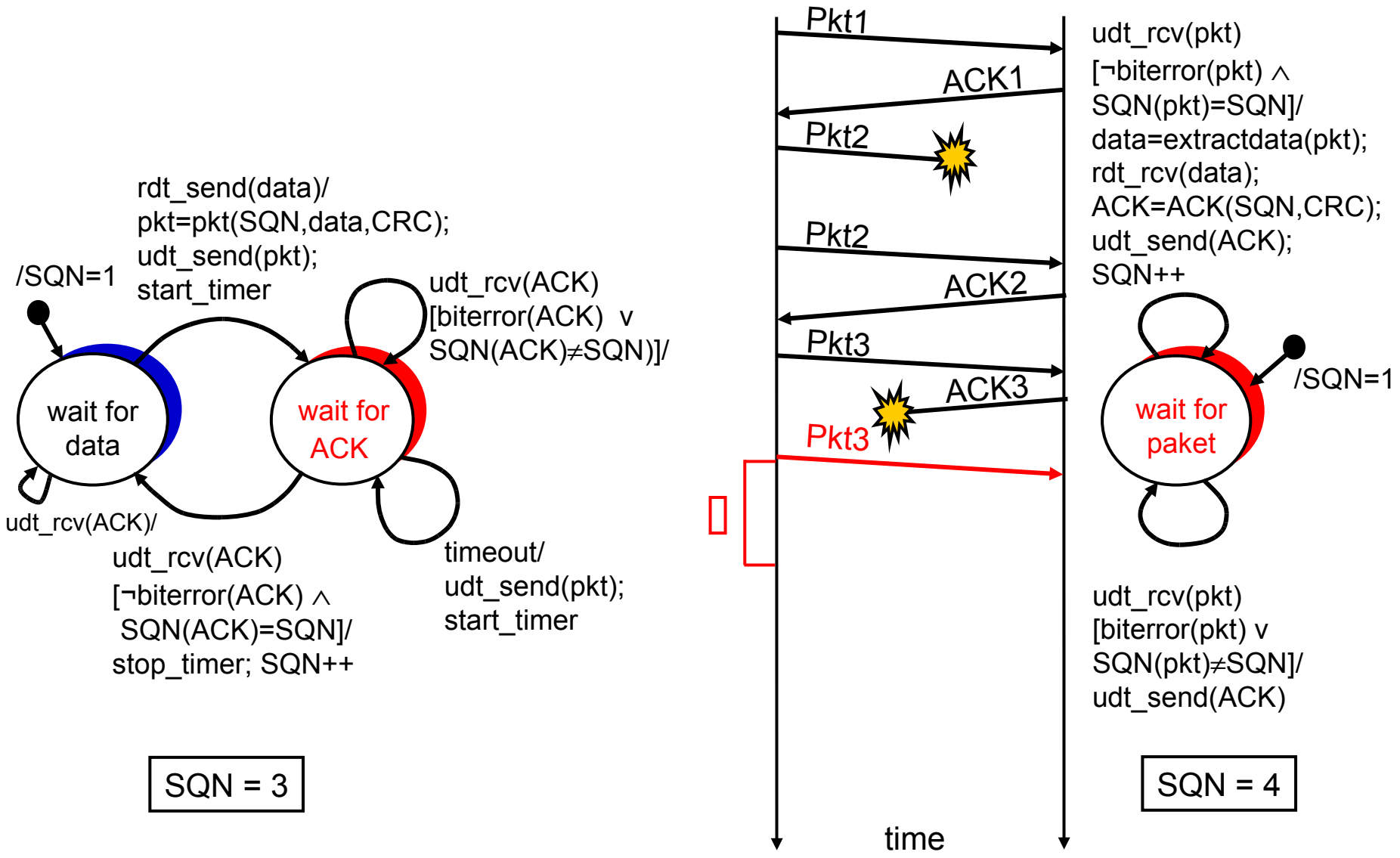
Stop-and-Wait: gubitak potvrde (ACK)



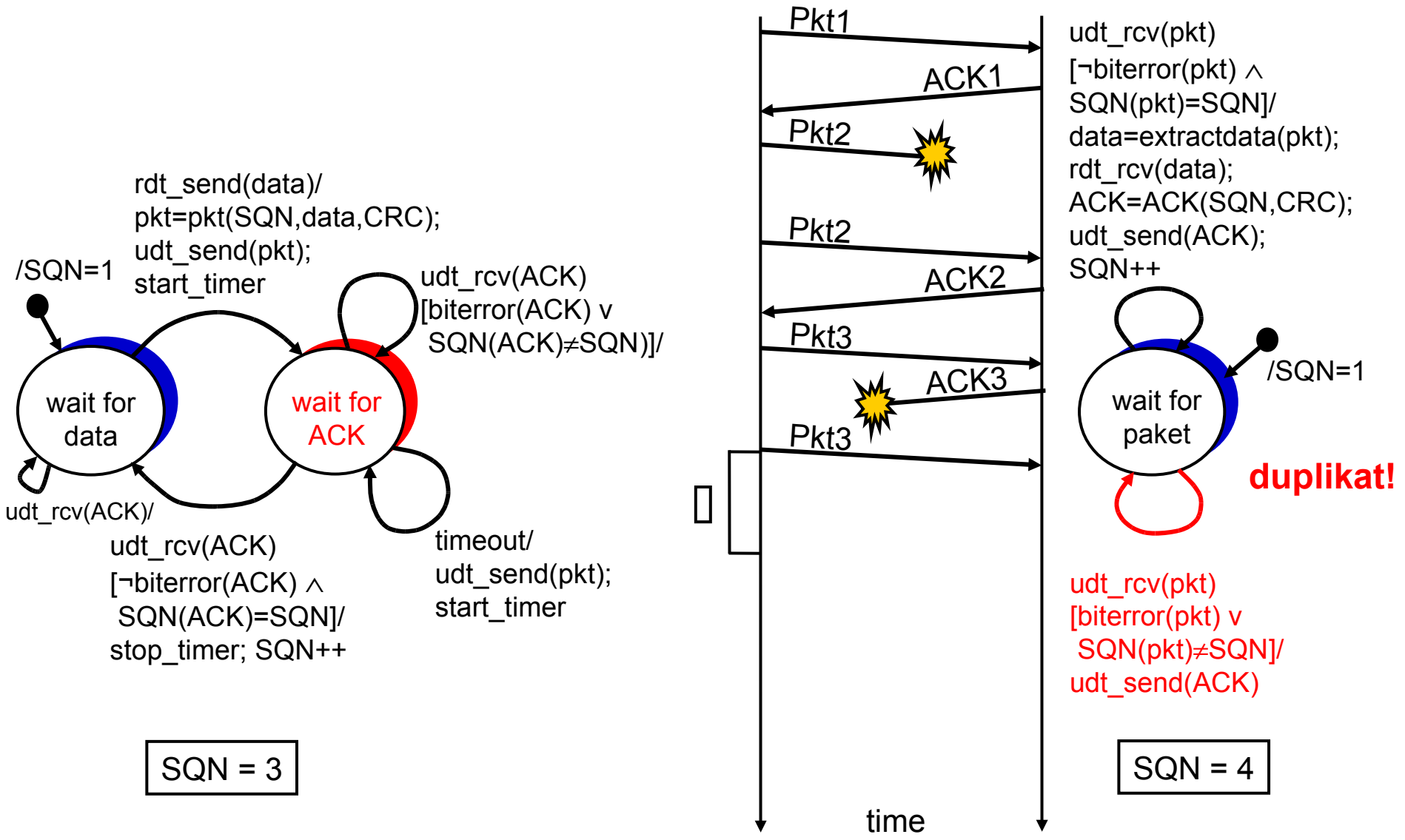
Stop-and-Wait: gubitak potvrde (ACK)



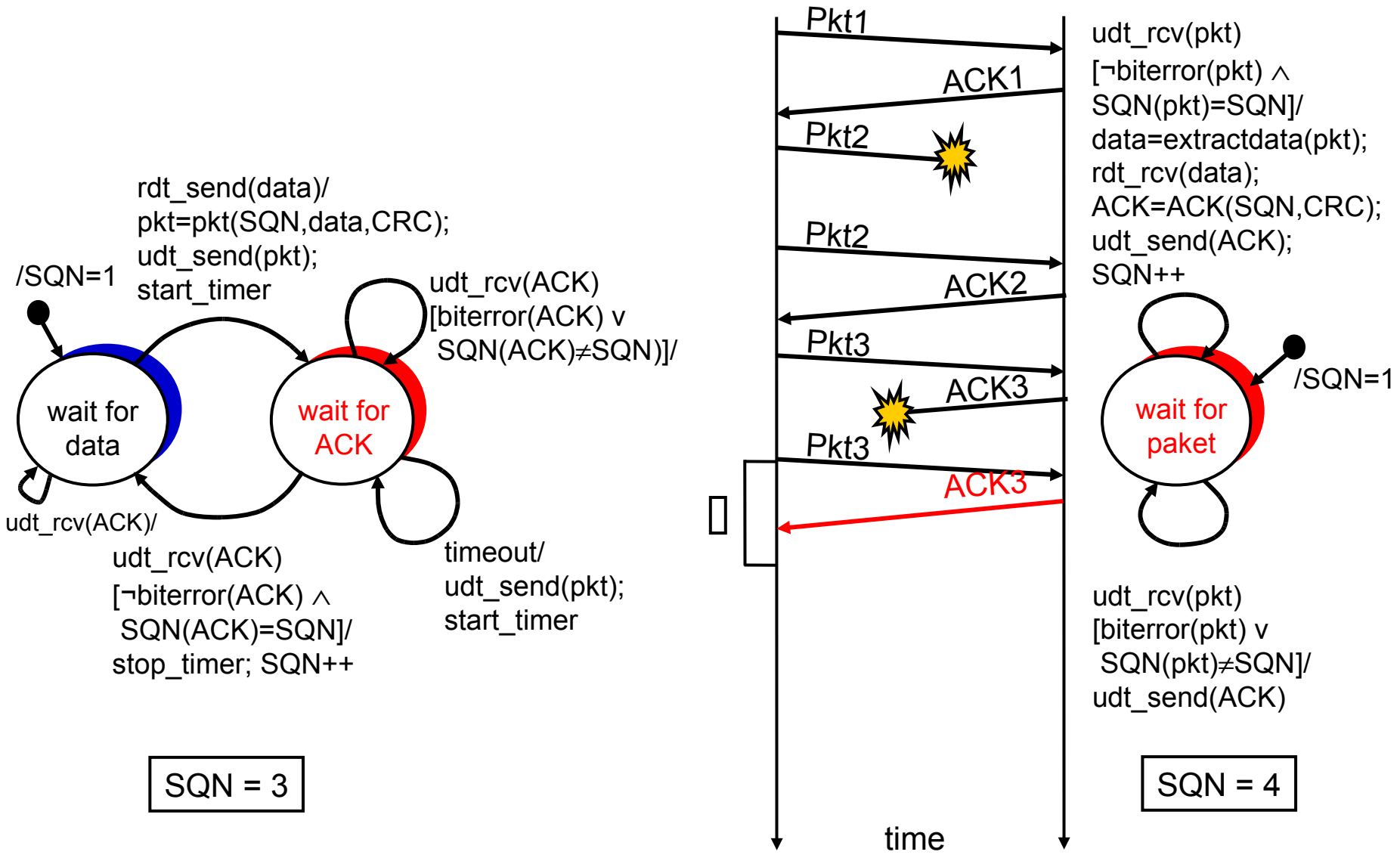
Stop-and-Wait: gubitak potvrde (ACK)



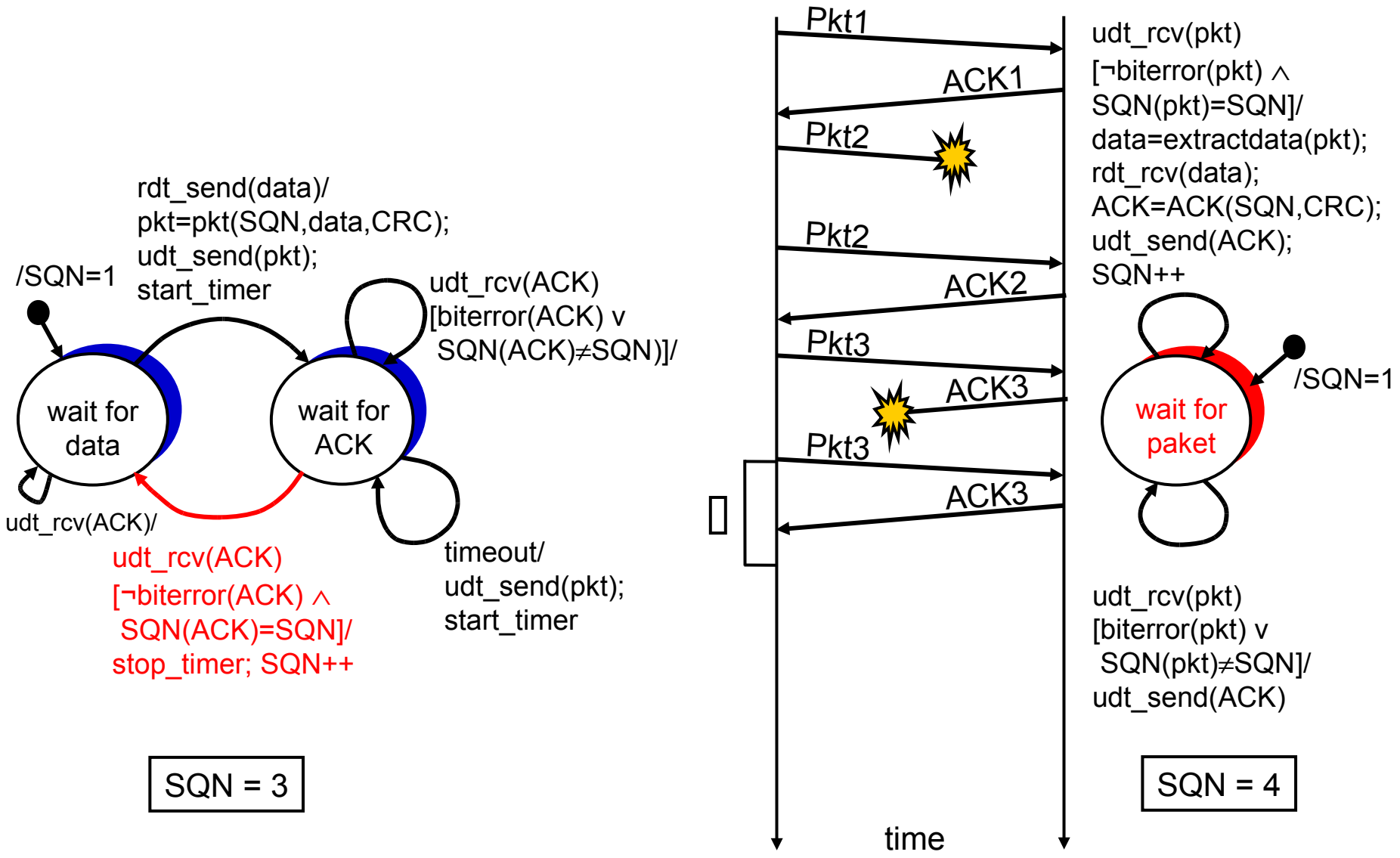
Stop-and-Wait: gubitak potvrde (ACK)



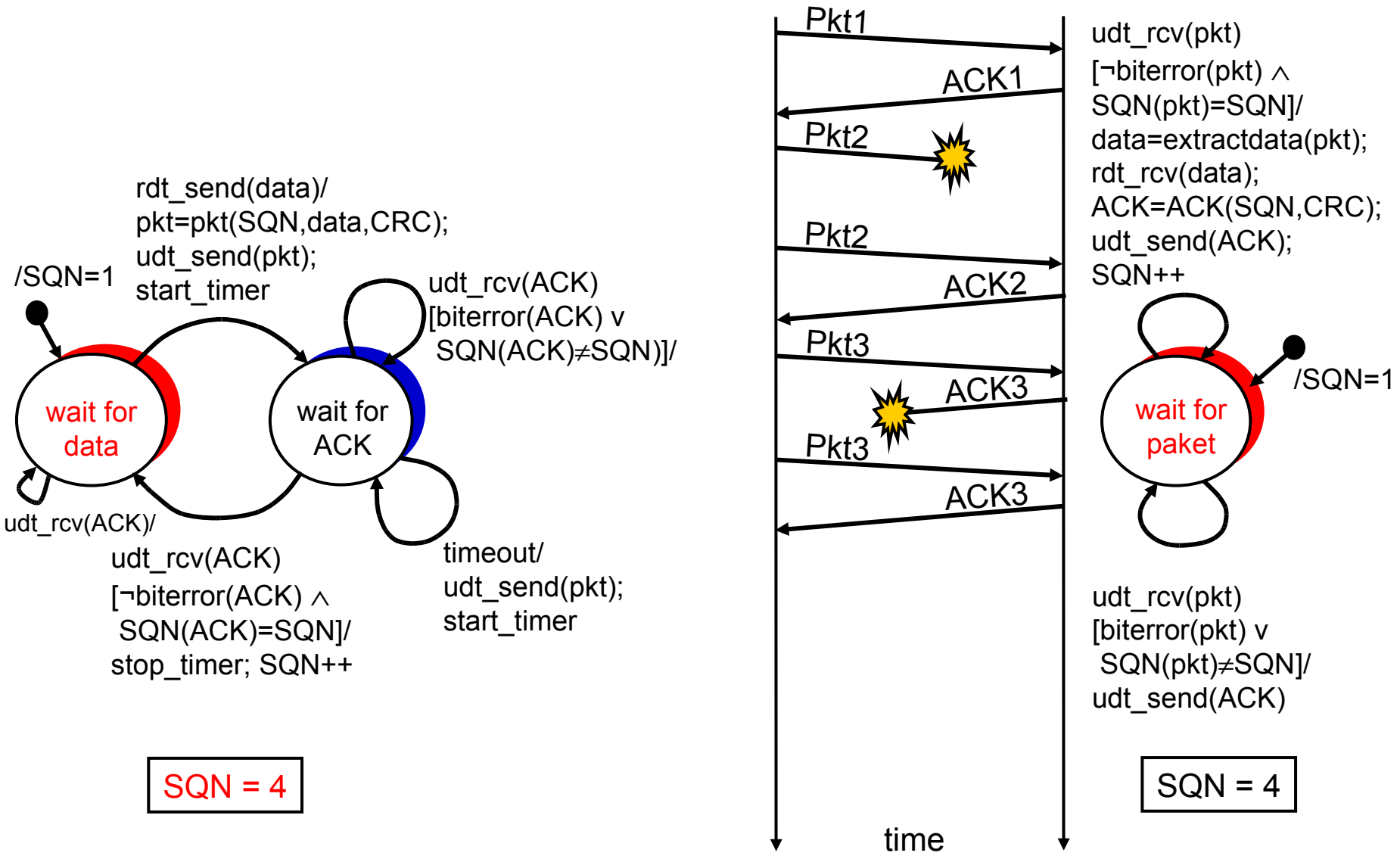
Stop-and-Wait: gubitak potvrde (ACK)



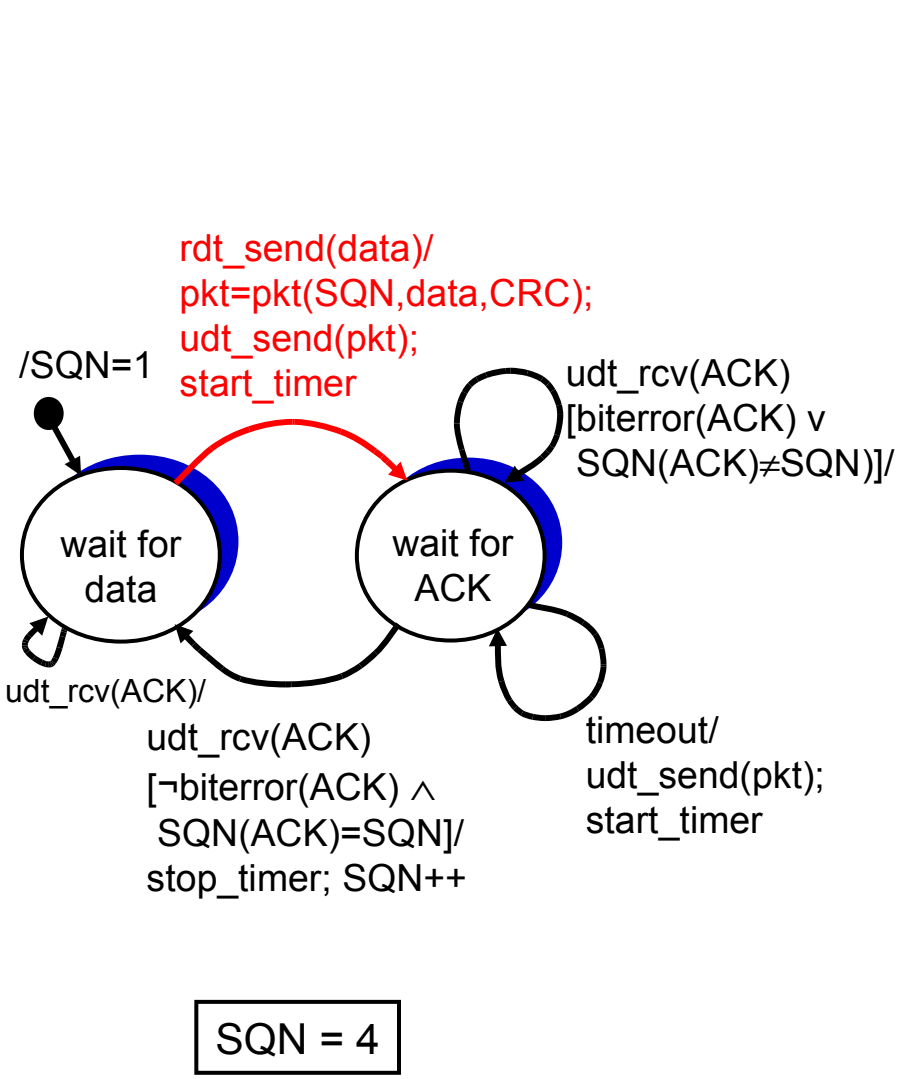
Stop-and-Wait: gubitak potvrde (ACK)

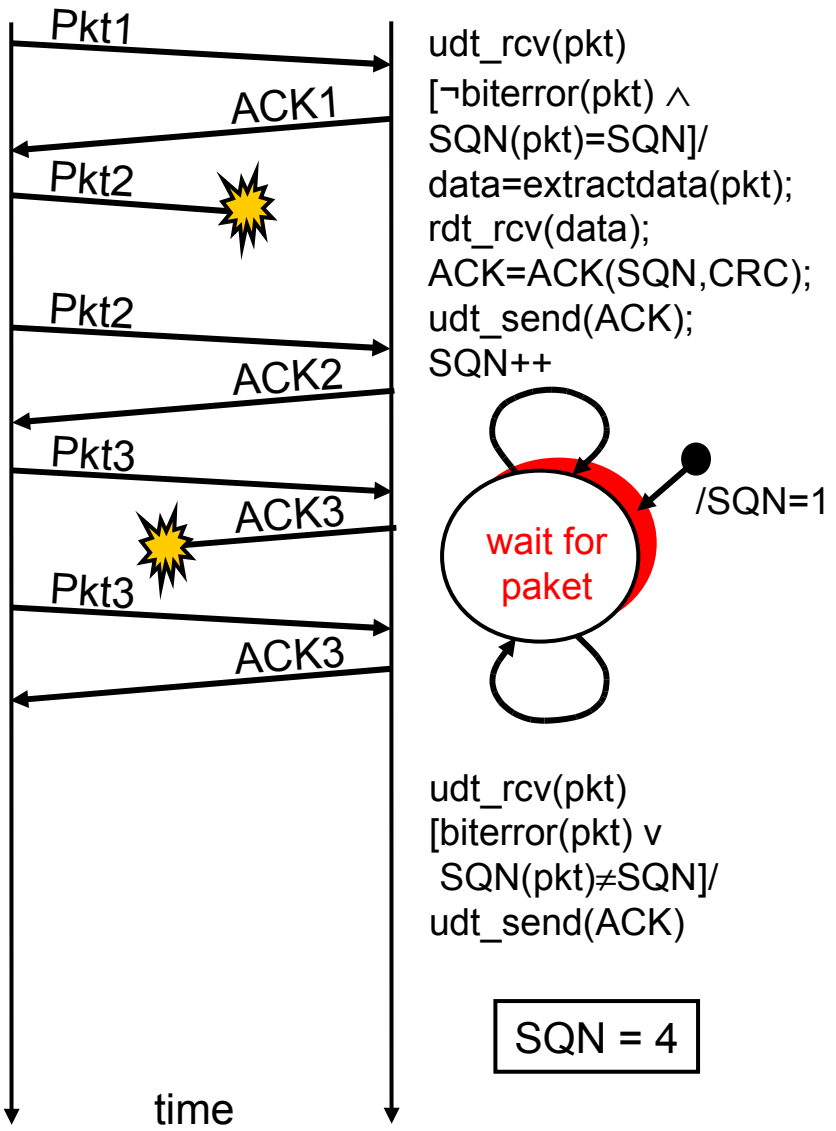


Stop-and-Wait: gubitak potvrde (ACK)

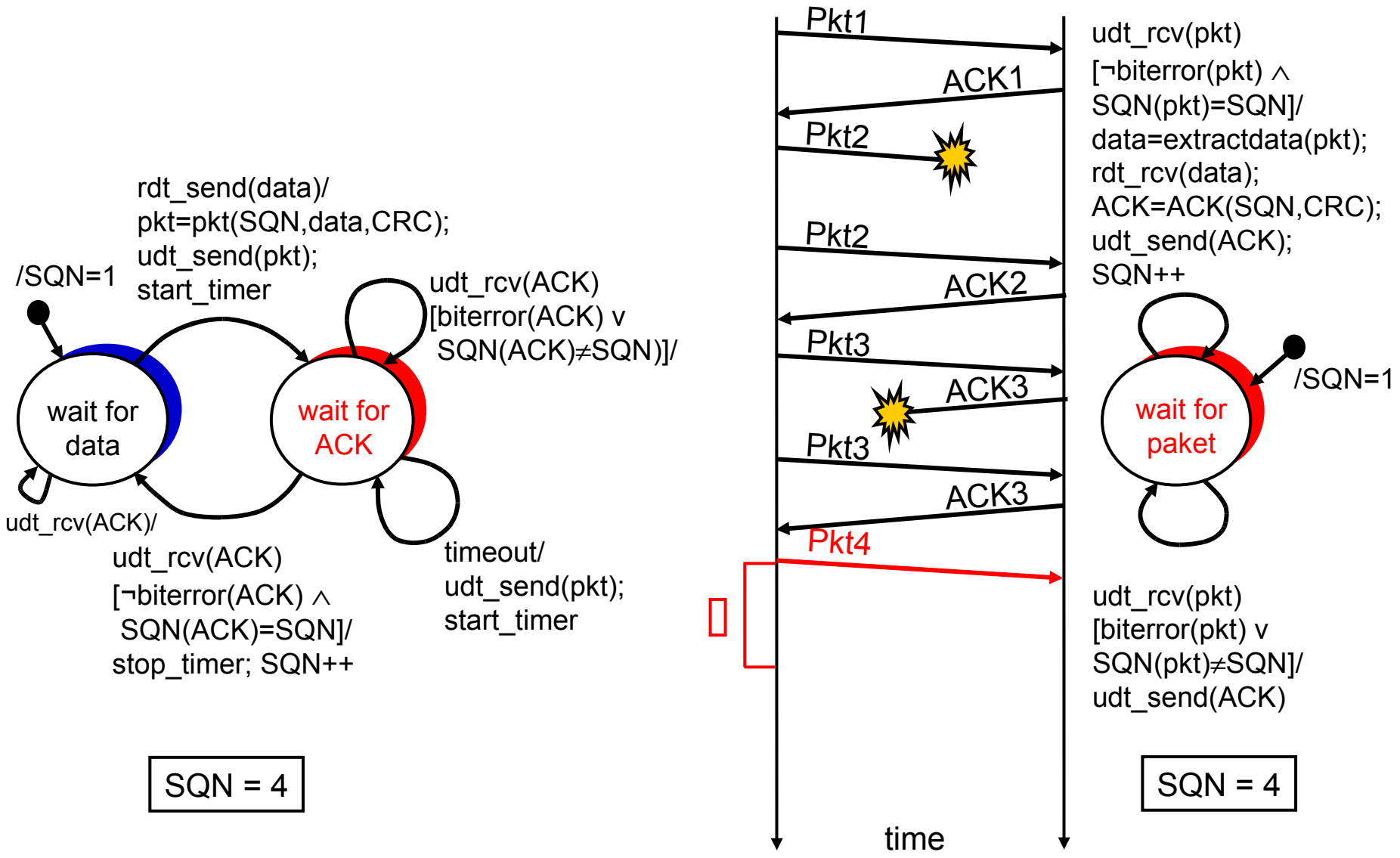


Stop-and-Wait: zakašnjeli ACK

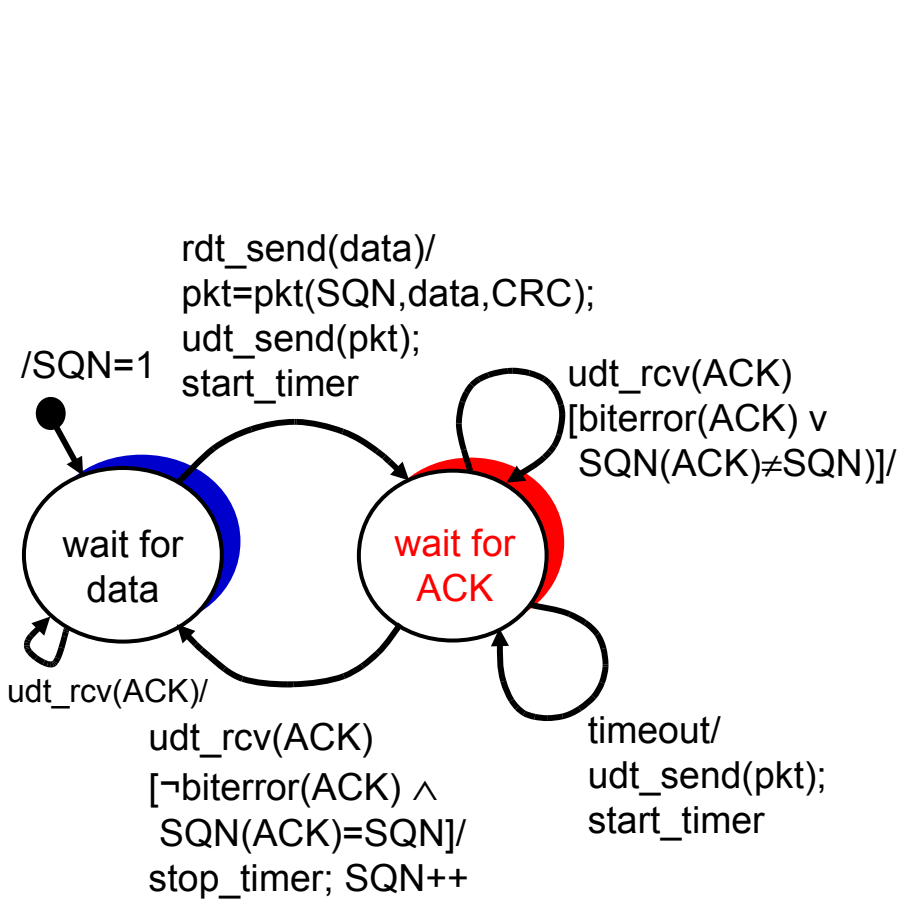


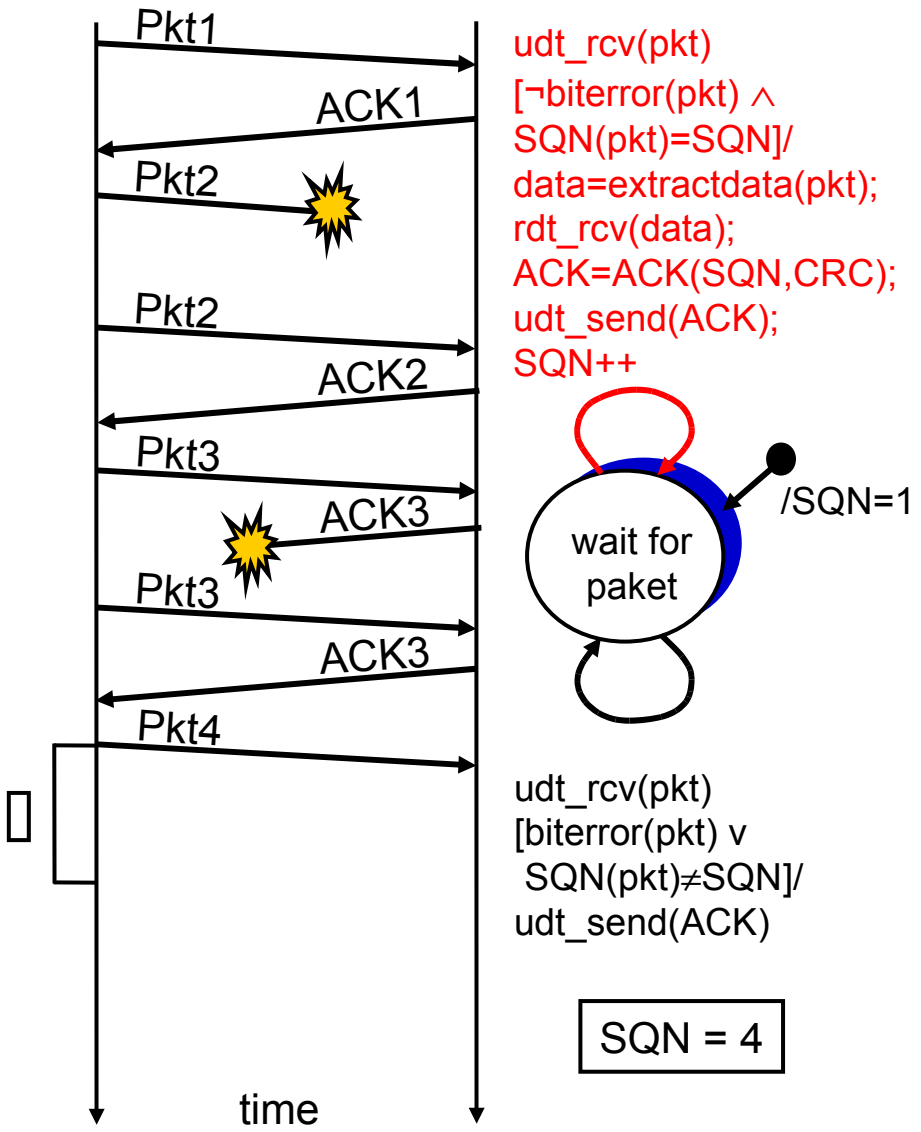


Stop-and-Wait: zakašnjeli ACK

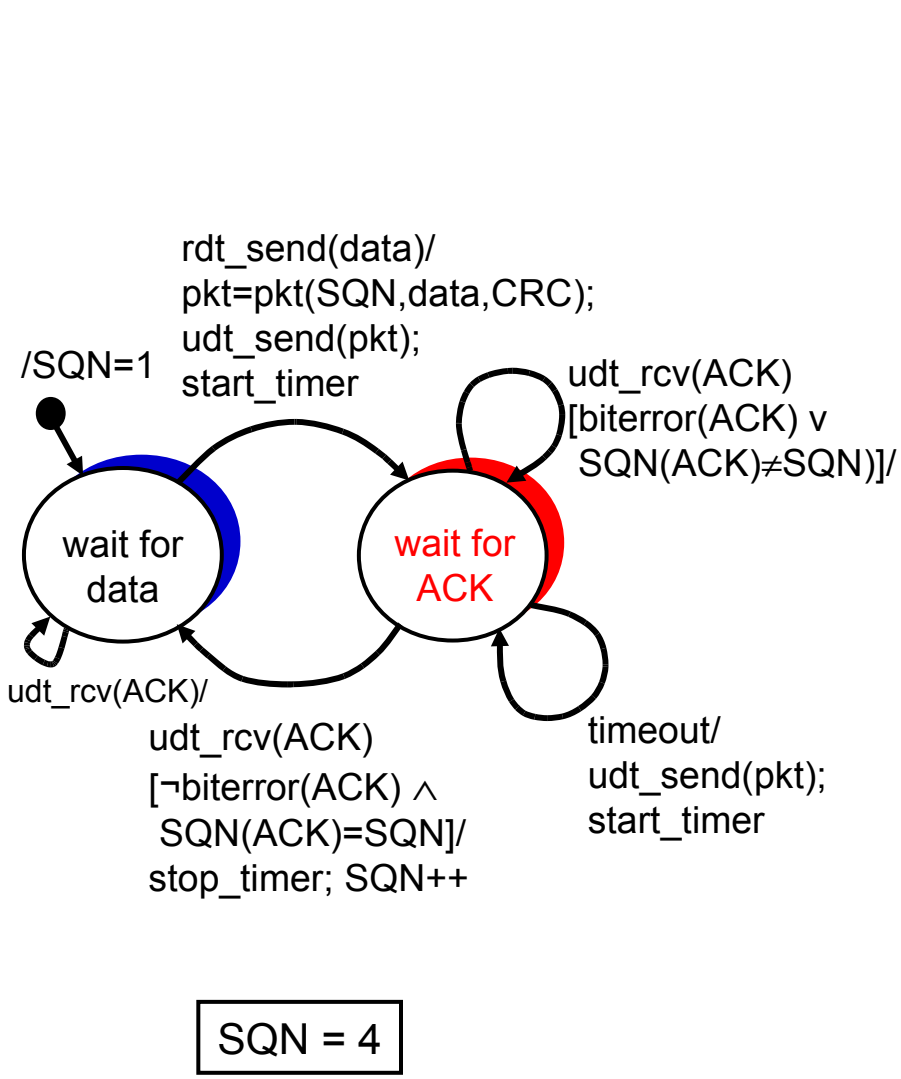


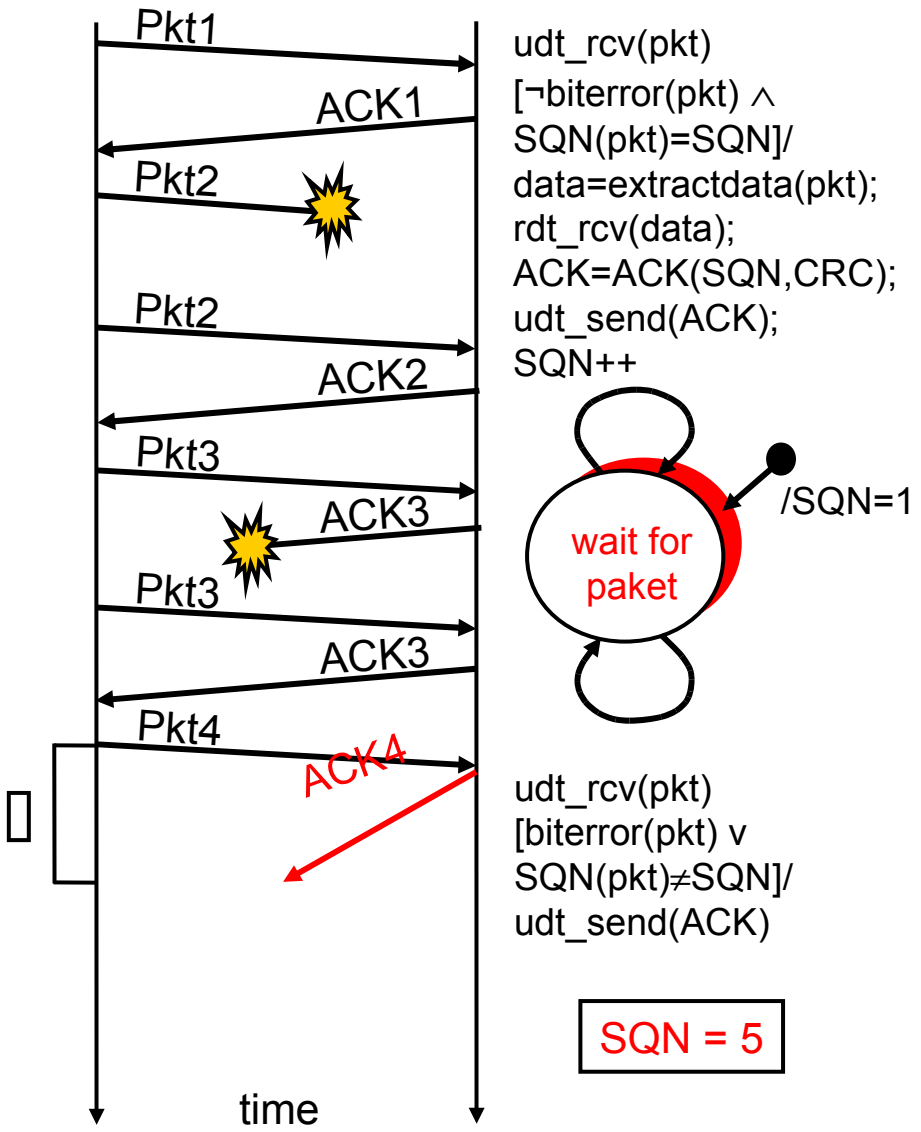
Stop-and-Wait: zakašnjeli ACK



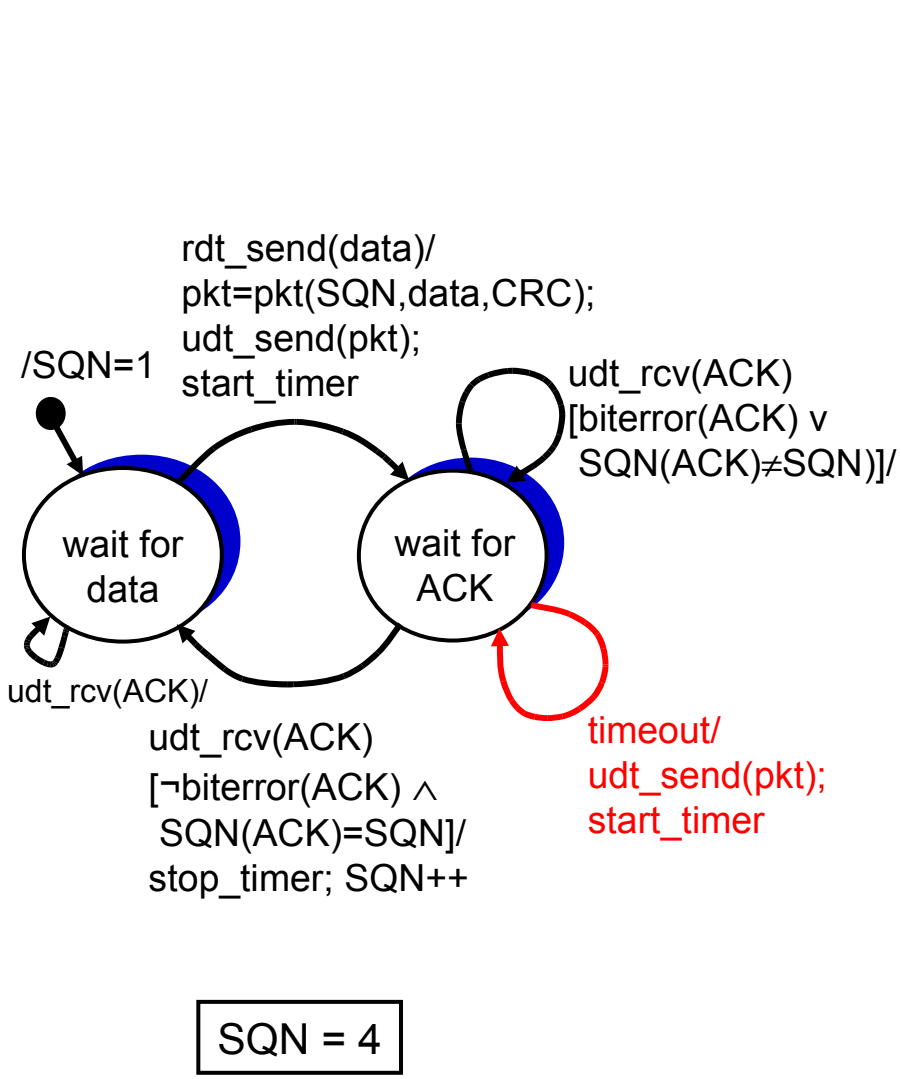


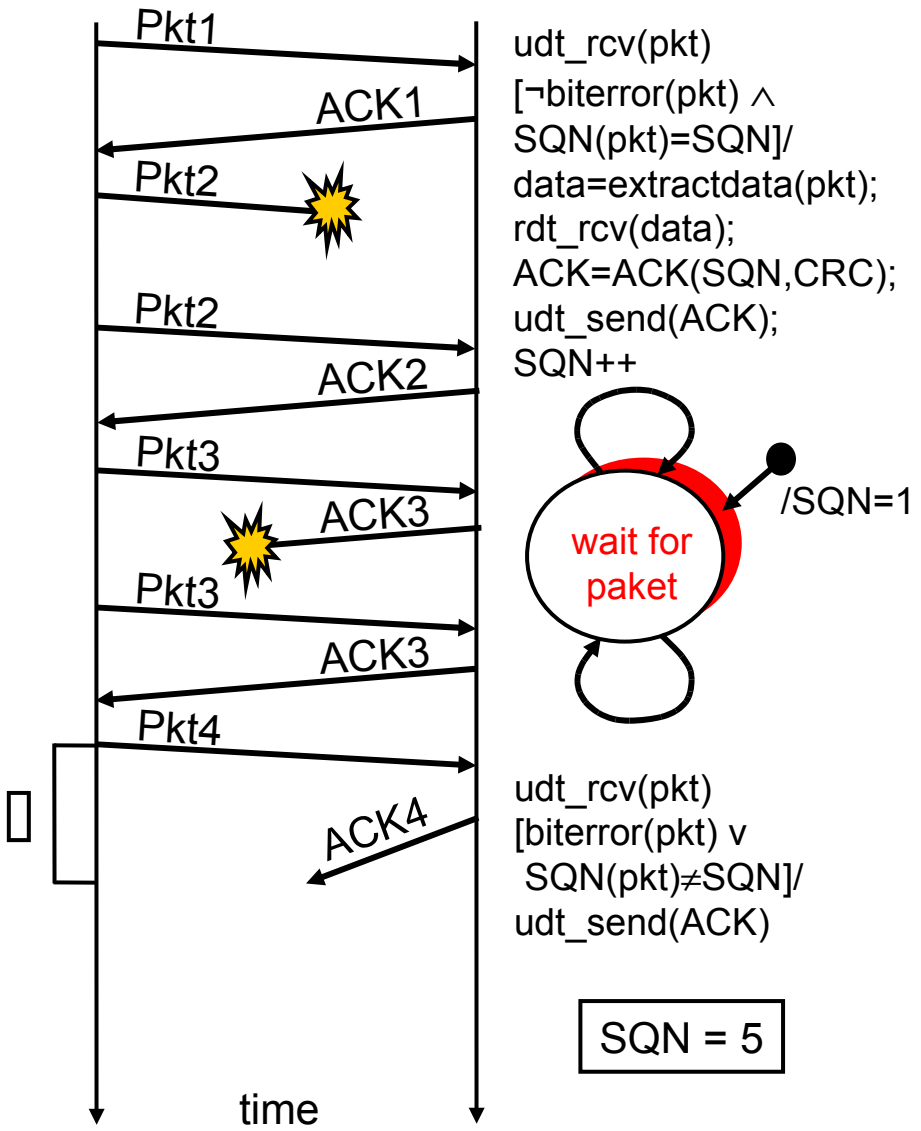
Stop-and-Wait: zakašnjeli ACK



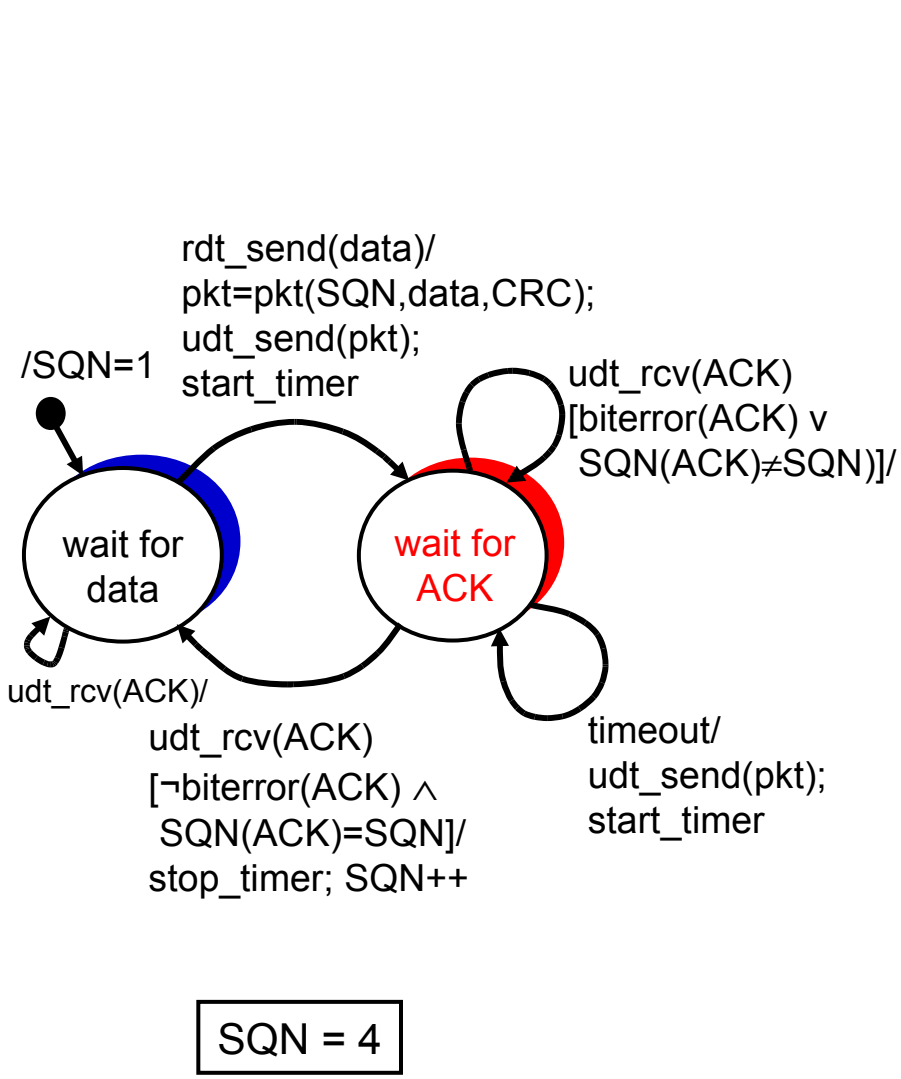


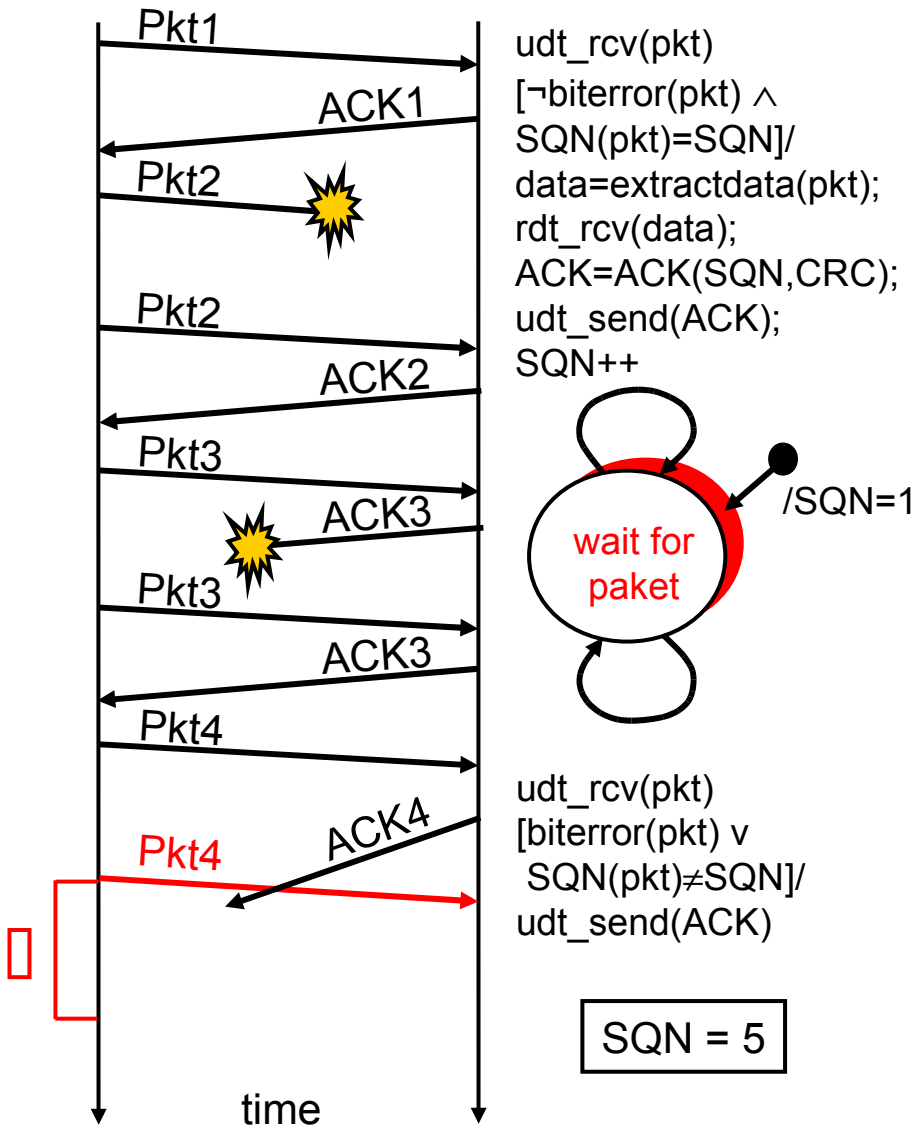
Stop-and-Wait: zakašnjeli ACK



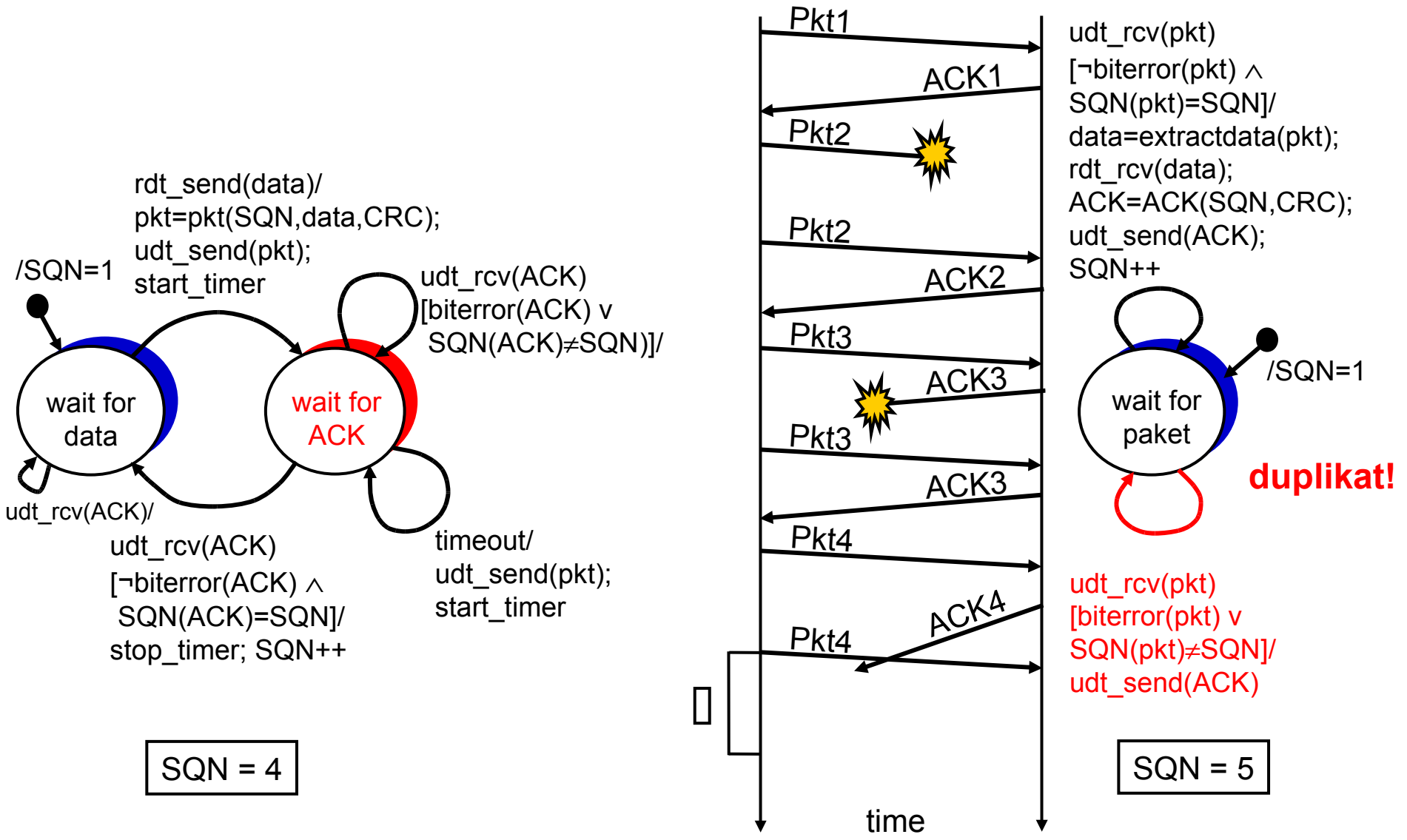


Stop-and-Wait: zakašnjeli ACK

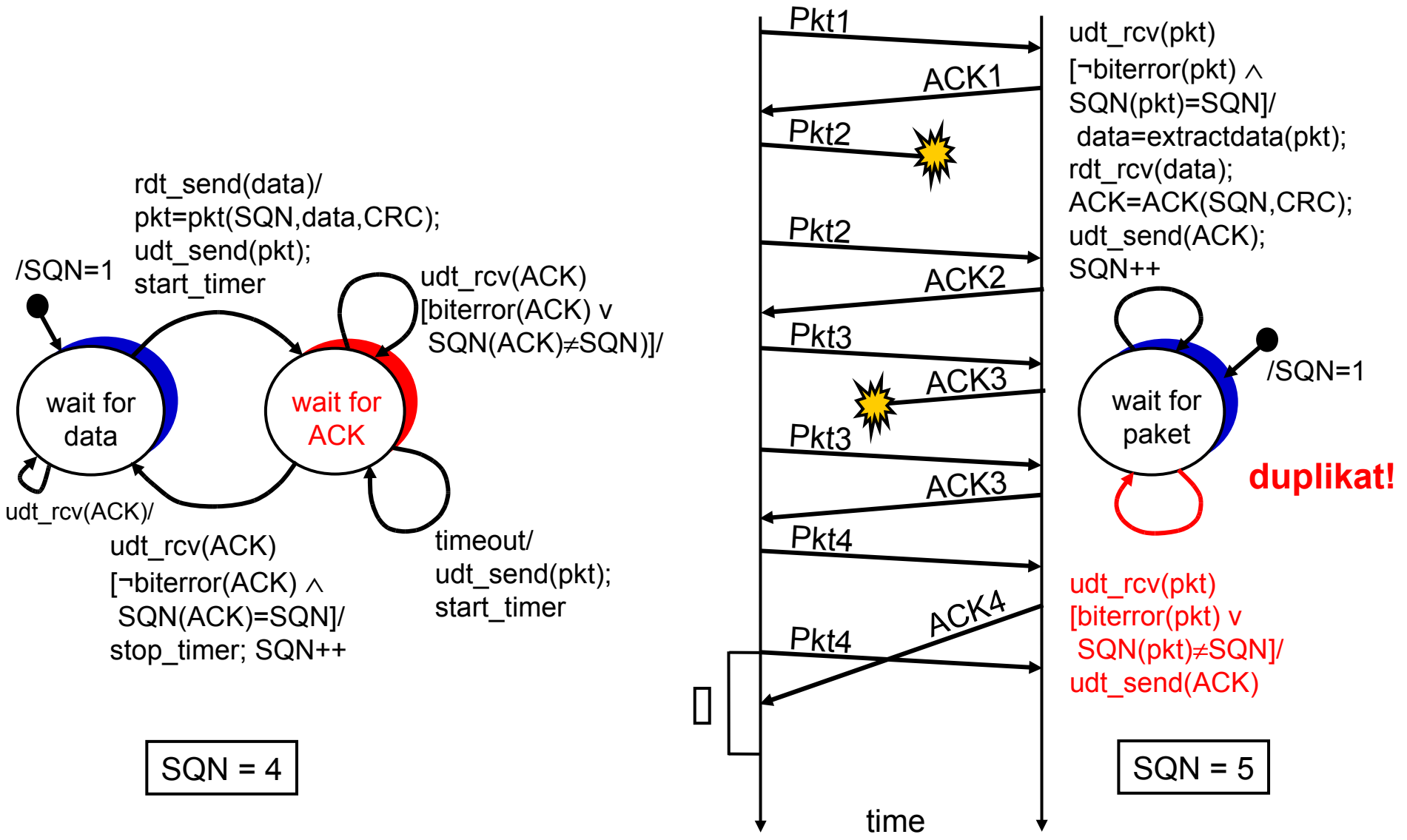




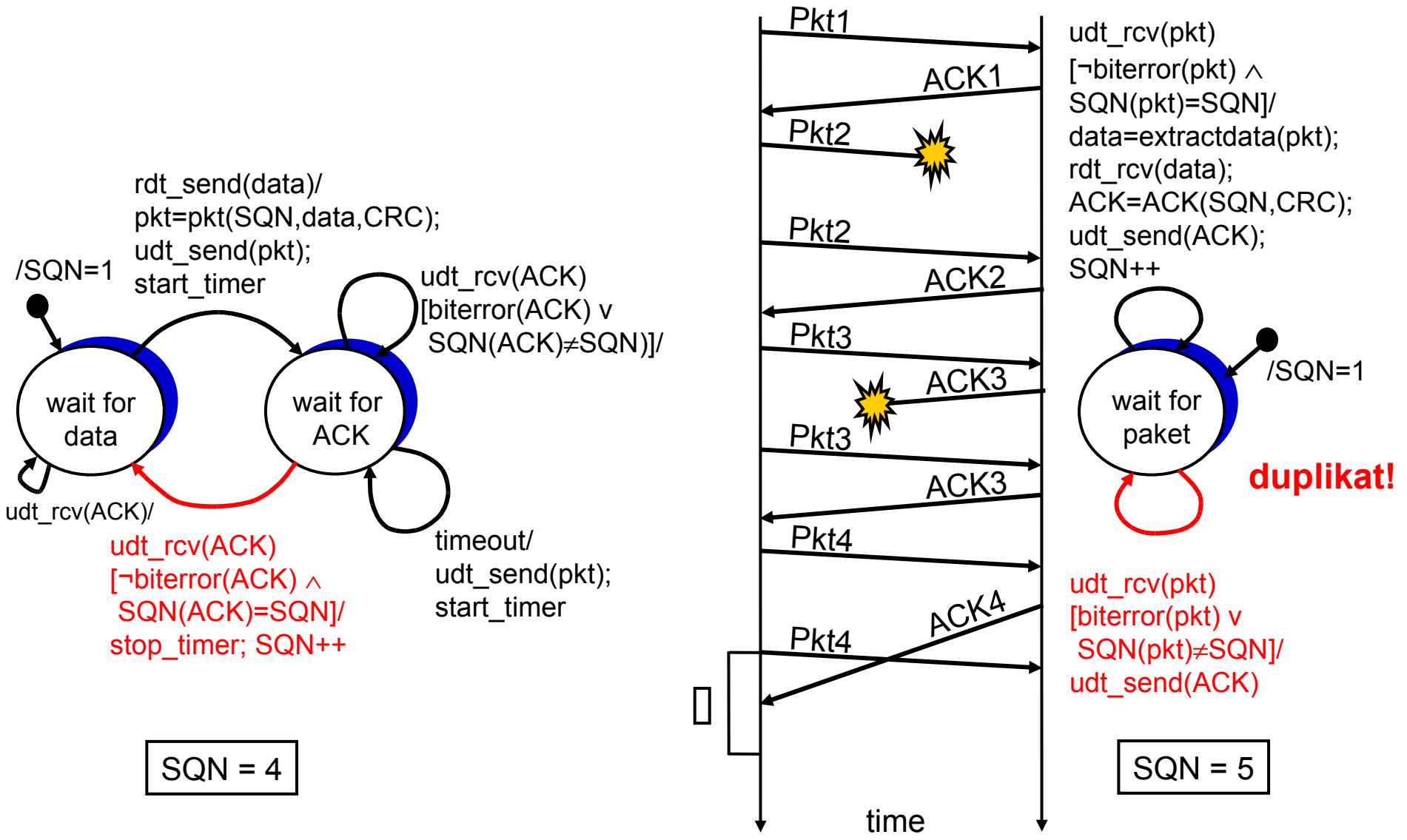
Stop-and-Wait: zakašnjeli ACK



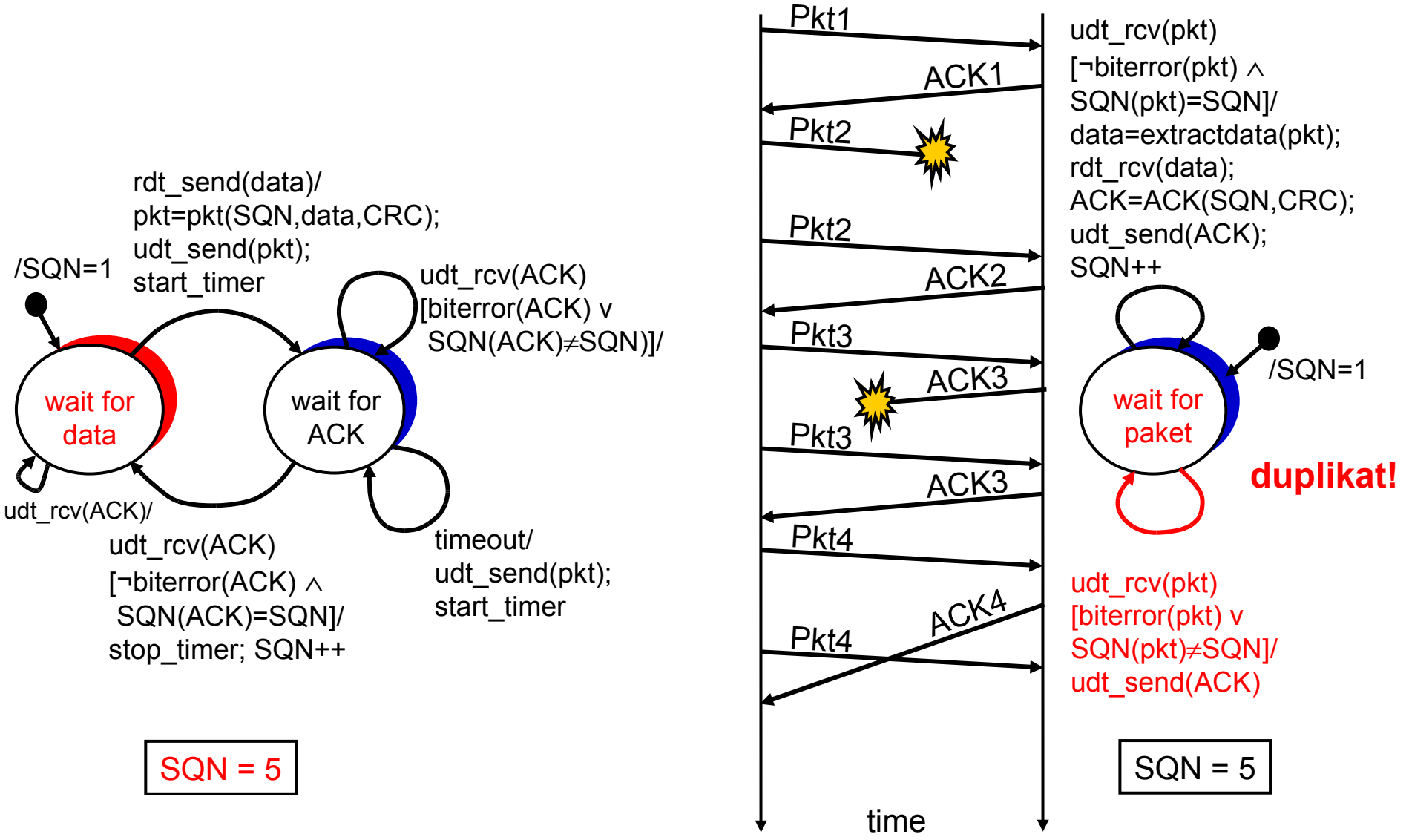
Stop-and-Wait: zakašnjeli ACK



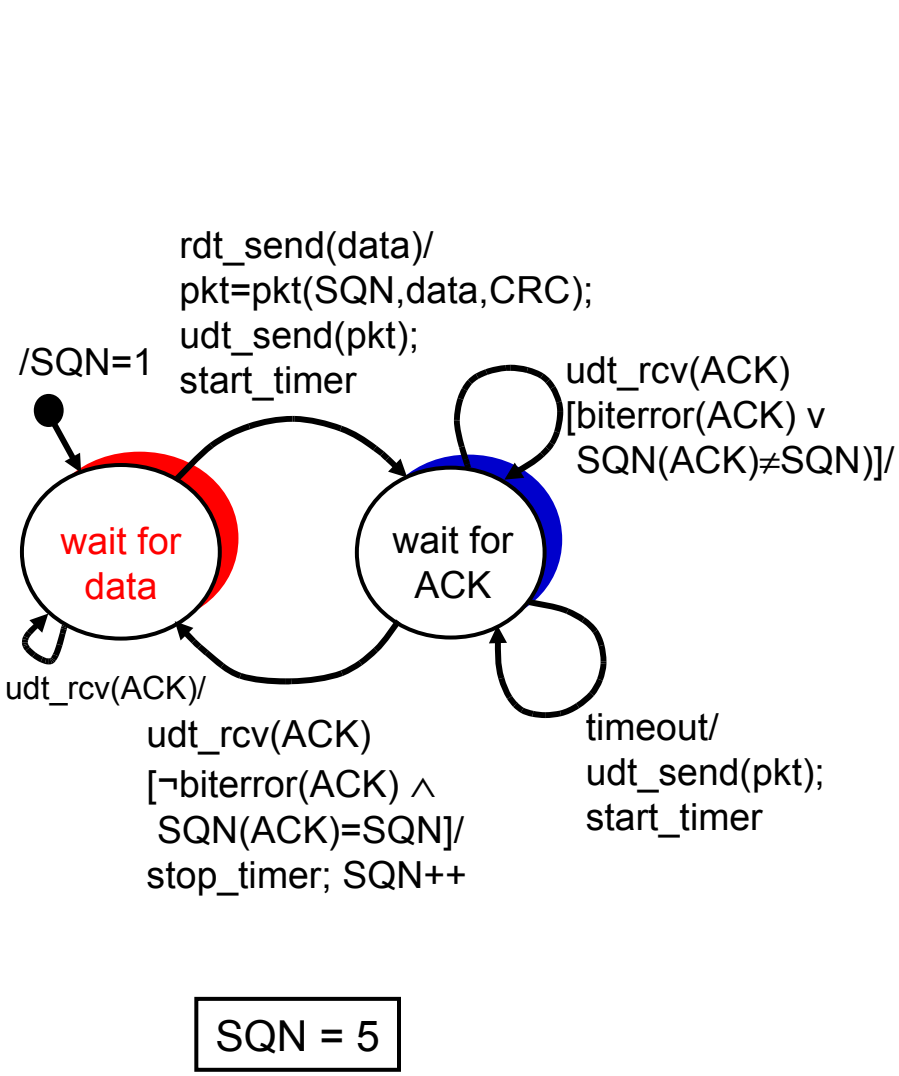
Stop-and-Wait: zakašnjeli ACK

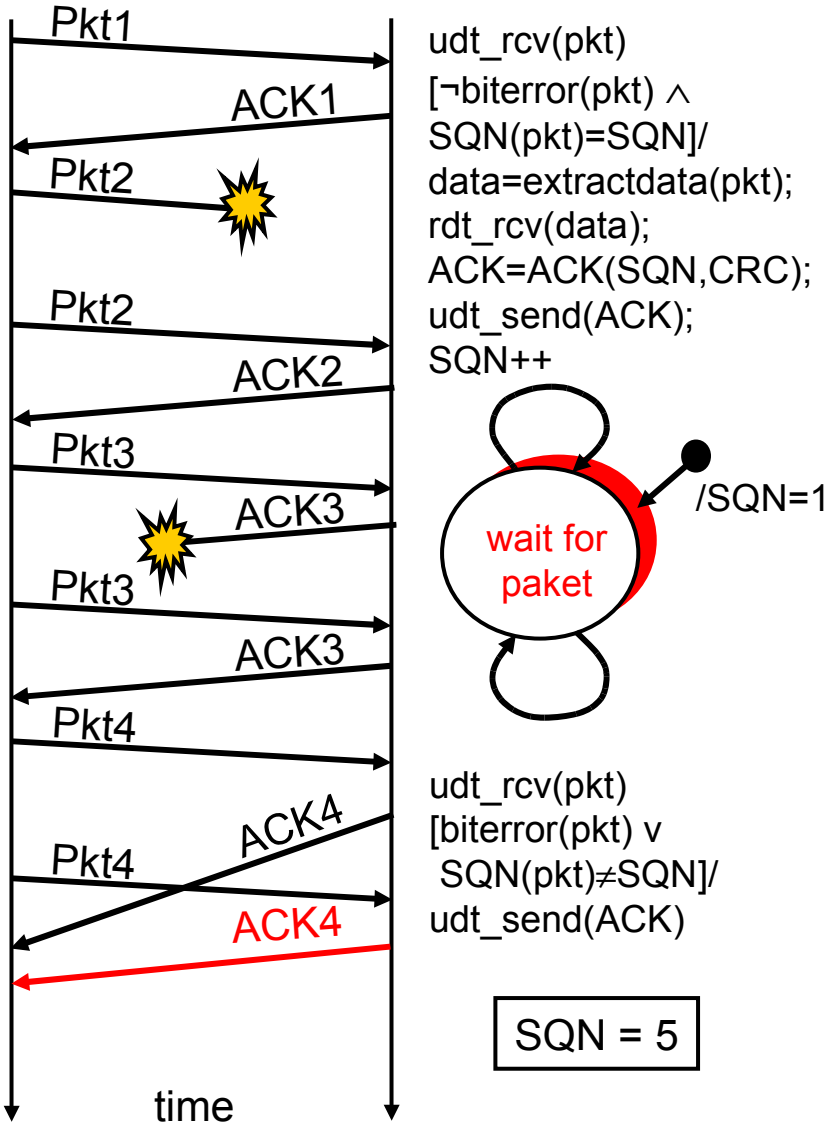


Stop-and-Wait: zakašnjeli ACK

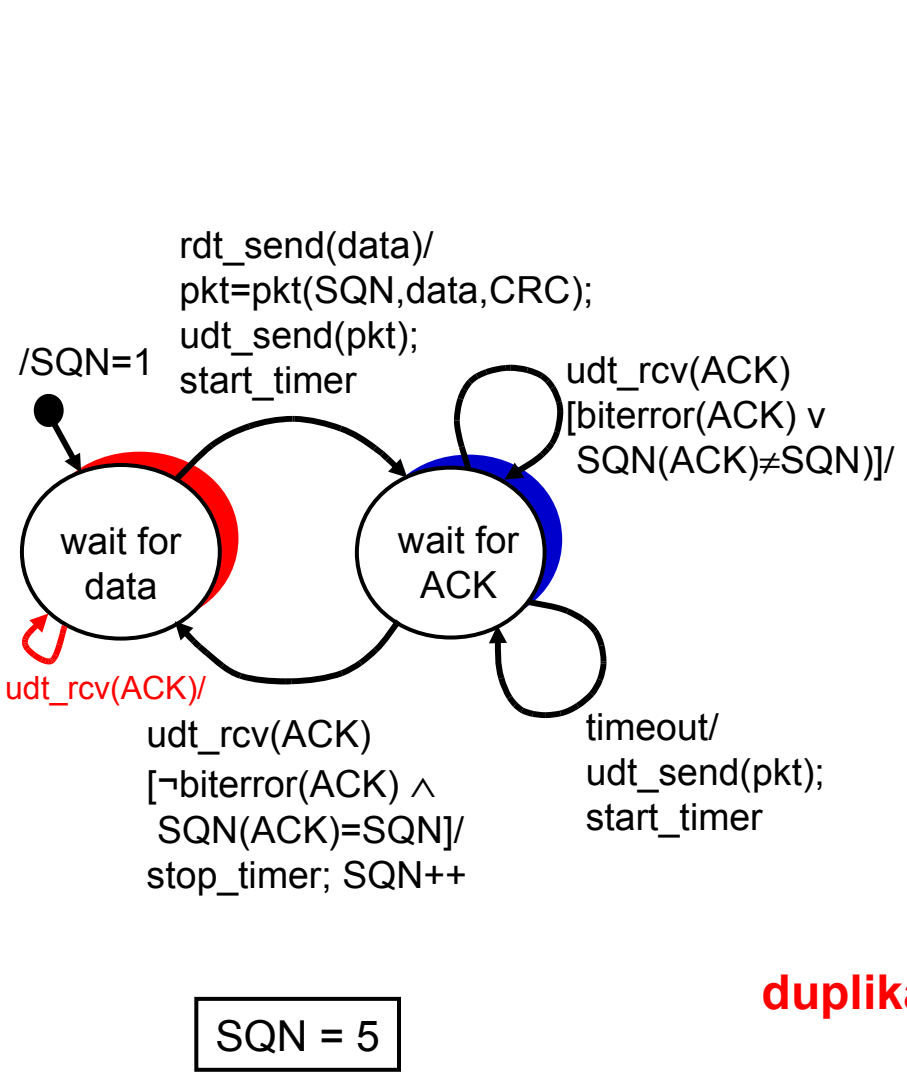


Stop-and-Wait: zakašnjeli ACK

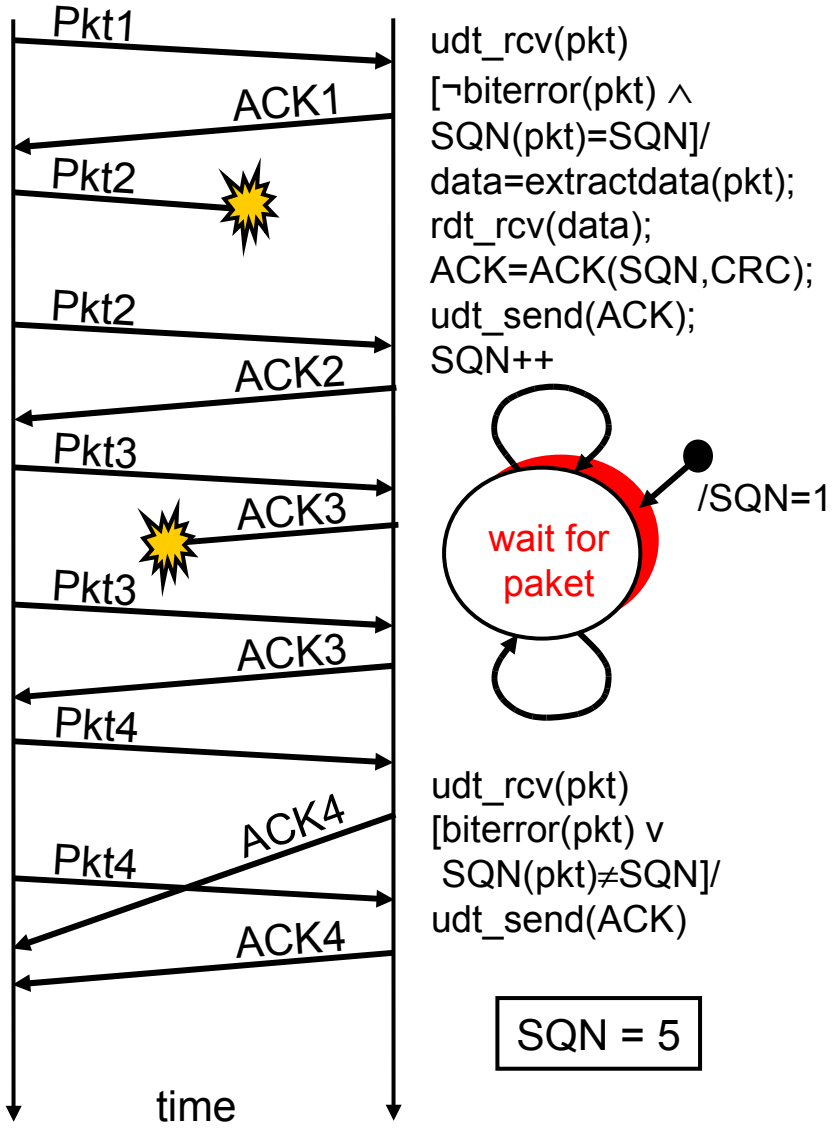




Stop-and-Wait: zakašnjeli ACK



duplikat!



Stop-and-Wait

- Prostor rednih brojeva (*sequence number space*)
  - prikaz rednih brojeva je konačan: polje s n bitova omogućuje 2^n rednih brojeva
  - višestruka primjena kroz ciklički prolaz
  - za Stop-and-Wait dovoljan je jedan bit za prikaz 2 redna broja: 0 i 1
  - Stop-and–Wait s 0 i 1 kao rednim brojevima zove se i Alternating-Bit-Protocol