Kompiuterių tinklai - Transporto sluoksnis

Aštunta paskaita (6 skyrius),
http://computernetworks5e.org/chap06.html

lekt. Vytautas Jančauskas

Transporto sluoksnis

Transporto sluoksnis

- Transporto sluoksnis skirtas tam, kad siuntėjo kompiuteryje veikiantis procesas galėtų perduoti duomenis gavėjo kompiuteryje veikiančiam procesui.
- Naudoja tinklo sluoksnio paslaugas.
- Tai papildomas abstrakcijos sluoksnis kai norima perduoti duomenis nesirūpinant kokiomis techninėmis galimybėmis tai bus padaryta.
- Aptarsim transport sluoksnio paslaugas, API, patikimumą, efektyvumą ir tokius protokolus kaip TCP ir UDP.

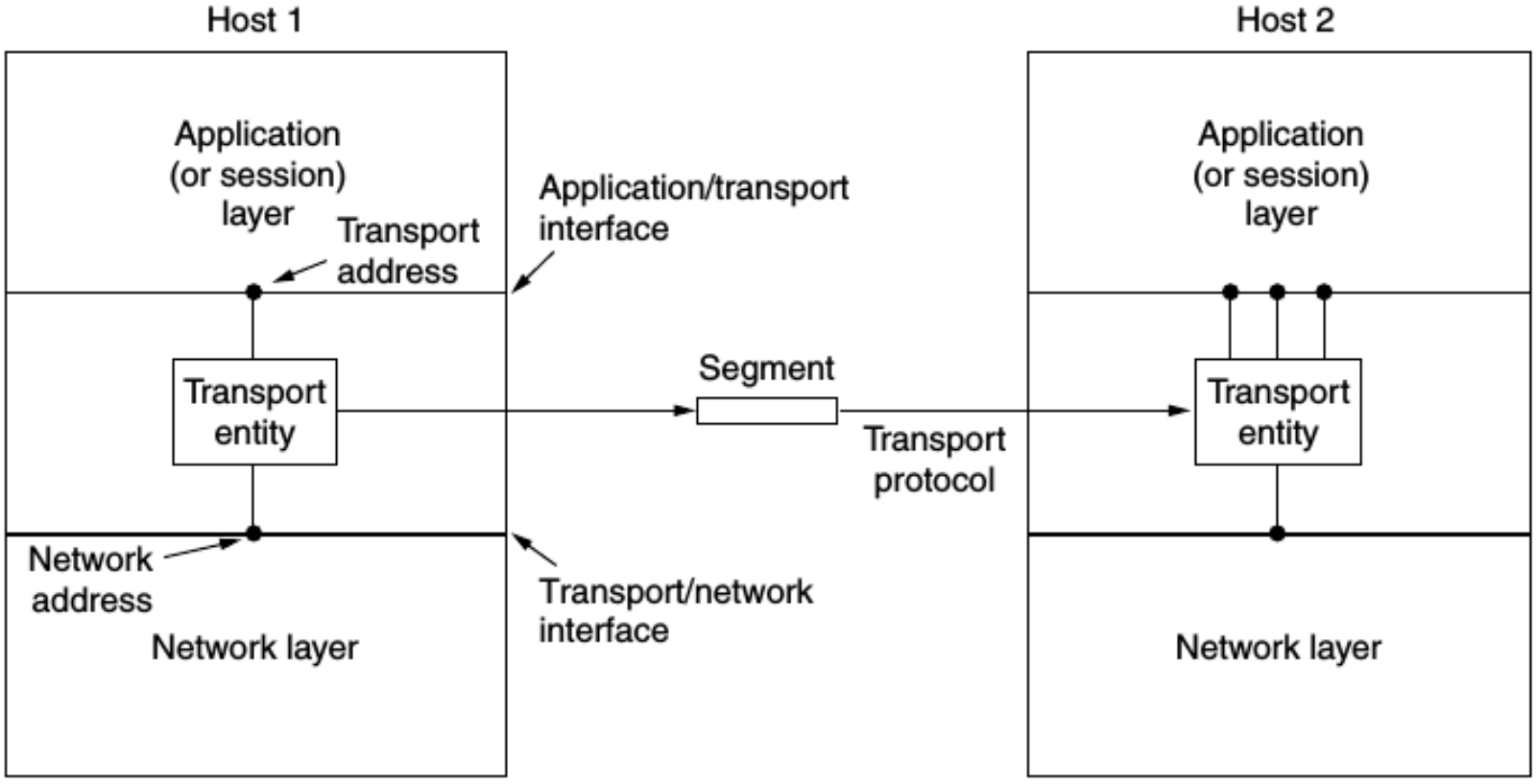


Figure 6-1. The network, transport, and application layers.

Transporto sluoksnio teikiamos paslaugos

- Transporto sluoksnis kaip ir mūsų anksčiau nagrinėti gali teikti paslaugus su ir be sujungimo.
- Šios paslaugos analogiškos panašiom tinklo sluoksnio paslaugoms.
- Jei jie tokie panašūs kodėl reikia transporto sluoksnio?
- Transporto sluoksnio programinė įranga veikia tik siuntėjo ir gavėjo kompiuteriuose. Tinklo sluoksnio programinė įranga veikia ir maršrutizatoriuose.
- Transporto sluoksnis užtikrina naudotojui patikimą prieigą prie tinklo sluoksnio paslaugų. Pavyzdžiui nutrūkus sujungimui jis atnaujinamas taip, kad programos naudotojas apie tai nesužinos.

Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	Request a release of the connection

Figure 6-2. The primitives for a simple transport service.

Transporto sluoksnio interfeisas

- Transporto sluoksnio duomenų vienetą vadinsime segmentu.
- CONNECT išsiunčia CONNECTION REQUEST segmentą. Tikrinama ar gavėjęs užsiblokavęs po LISTEN komandos. Jei taip, išsiunčiamas CONNECTION ACCEPTED segmentas ir užmezgamas sujungimas.
- Duomenimis keičiamasi naudojant SEND ir RECEIVE primityvus.
- Visi segmentai turi būti patvirtinami. Tačiau to naudotojas nemato ir nejaučia.
- Kai sujungimo nebereikia siunčiamas DISCONNECT segmentas.

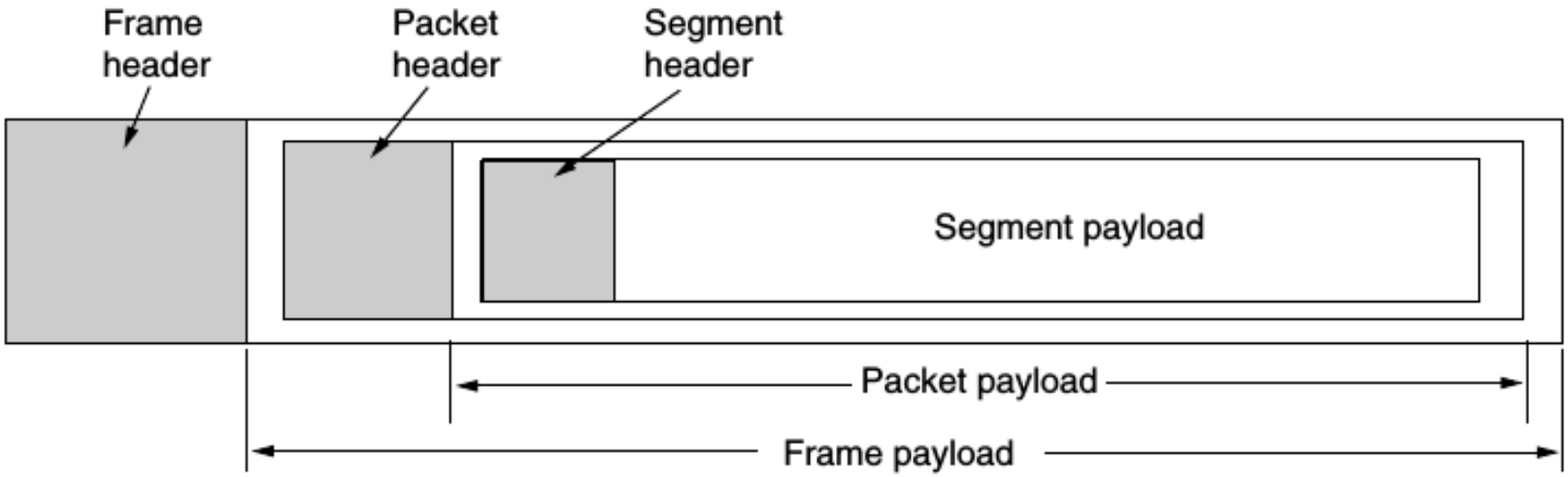


Figure 6-3. Nesting of segments, packets, and frames.

Berkeley sockets

- Sukurti kaip Berkeley UNIX 4.2BSD dalis 1983 metais.
- Socket'ai yra de facto standartas abstrahuojant transporto paslaugas.
- Serveris naudoja SOCKET, BIND, LISTEN ir ACCEPT primityvus sujungimui užmegsti.
- Klientas naudoja SOCKET, CONNECT.
- Duomenimis keičiamasi naudojant SEND, RECEIVE.
- Sujungimas uždaromas abiem iškvietus CLOSE primityvą.

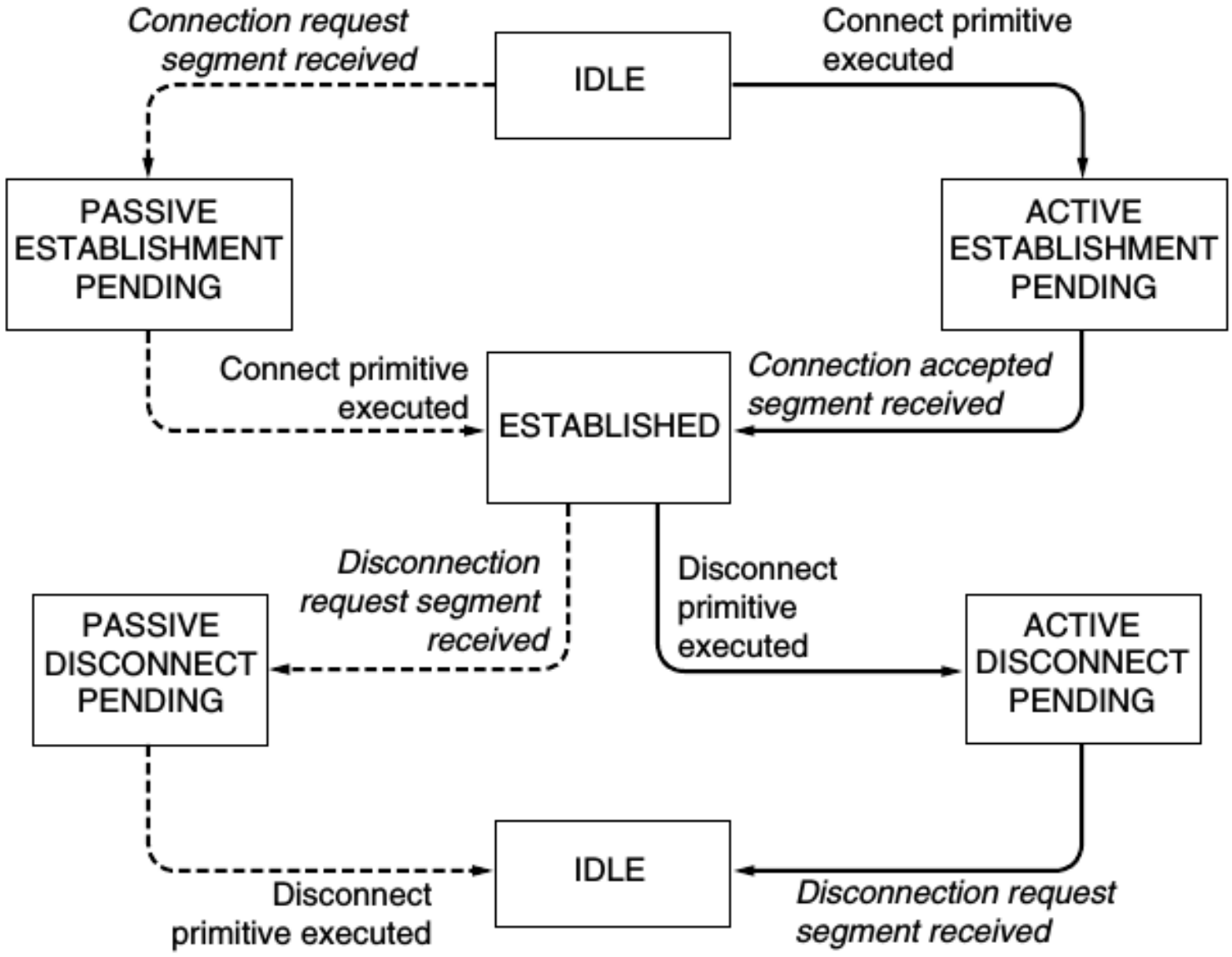


Figure 6-4. A state diagram for a simple connection management scheme. Transitions labeled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence.

Primitive	Meaning
SOCKET	Create a new communication endpoint
BIND	Associate a local address with a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Passively establish an incoming connection
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

Figure 6-5. The socket primitives for TCP.

Transporto sluoksnio protokolai

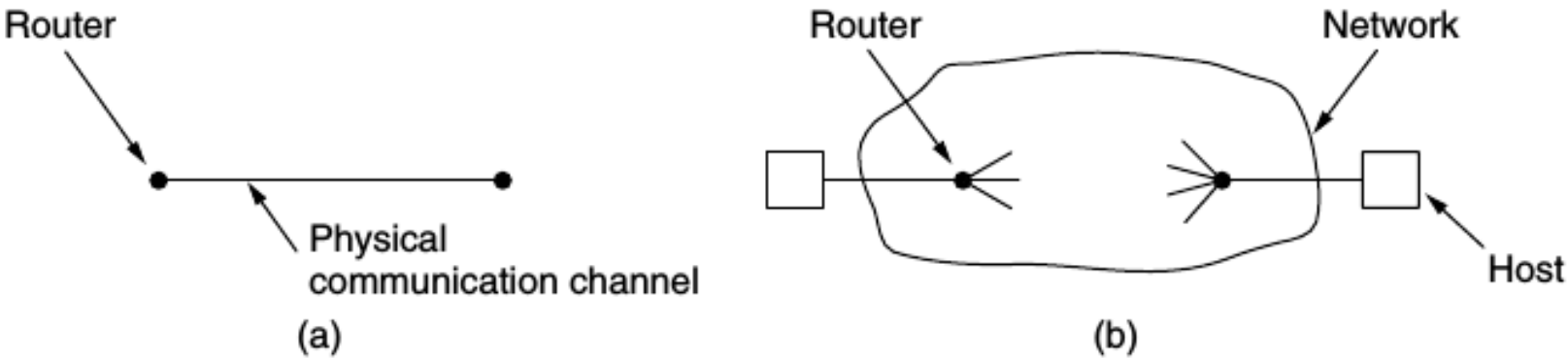


Figure 6-7. (a) Environment of the data link layer. (b) Environment of the transport layer.

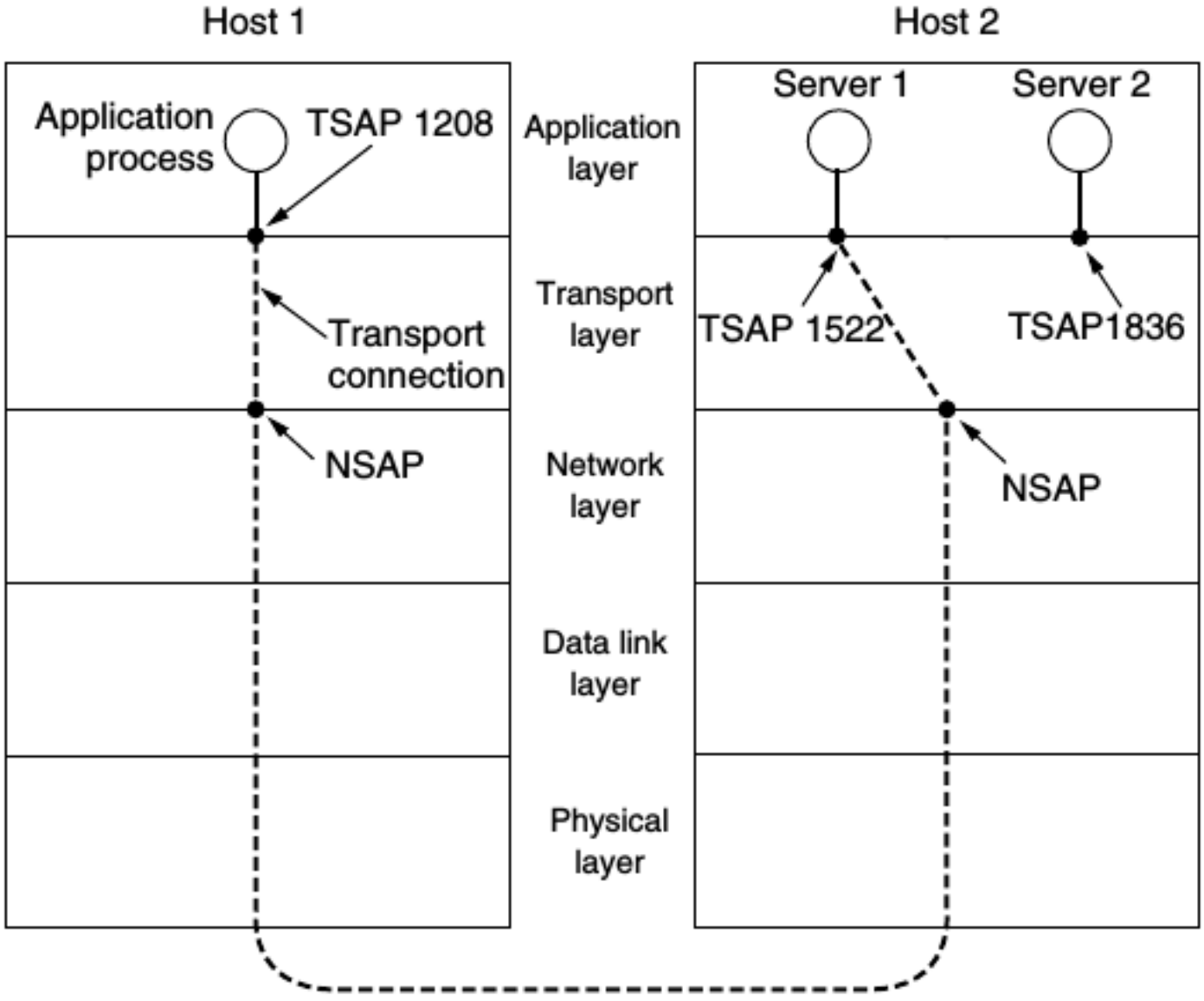


Figure 6-8. TSAPs, NSAPs, and transport connections.

Adresavimas (I)

- Transporto sluoksnyje iškyla papildoma problema kai tinklu nudojasi keli procesai. Kuriam procesui perduoti gaunamus paketus?
- Tam naudojama atskira sistema kiekvienam sujungimui priskirianti adresą - **TSAP (Transport Service Access Point)**. Vienas pavyzdys yra portai.
- Tinklo adresą vadinsime **NSAP (Network Service Access Points)**. Pavyzdys - IP adresai.

Adresavimas (II)

1. Pašto serveris prisiskiria TSAP 1522 kompiuteryje 2 ir laukia sujungimo.
2. Programa kompiuteryje 1 nori išsiųsti laišką, prisiskiria TSAP 1208 ir išsiunčia CONNECT užklausimą. Užklausime nurodoma, kad TSAP 1208 naudojamas kompiuteryje 1 yra siuntėjas, o TSAP 1522 kompiuteryje 2 yra gavėjas. Taip užmezgamas transporto ryšys tarp programų.
3. Klientas išsiunčia pašto pranešimą.
4. Serveris atsako, kad išsiųs tą pranešimą.
5. Transporto sujungimas atlaisvinamas.

Iš kur žinoti, kad pašto serverio TSAP yra 1522?

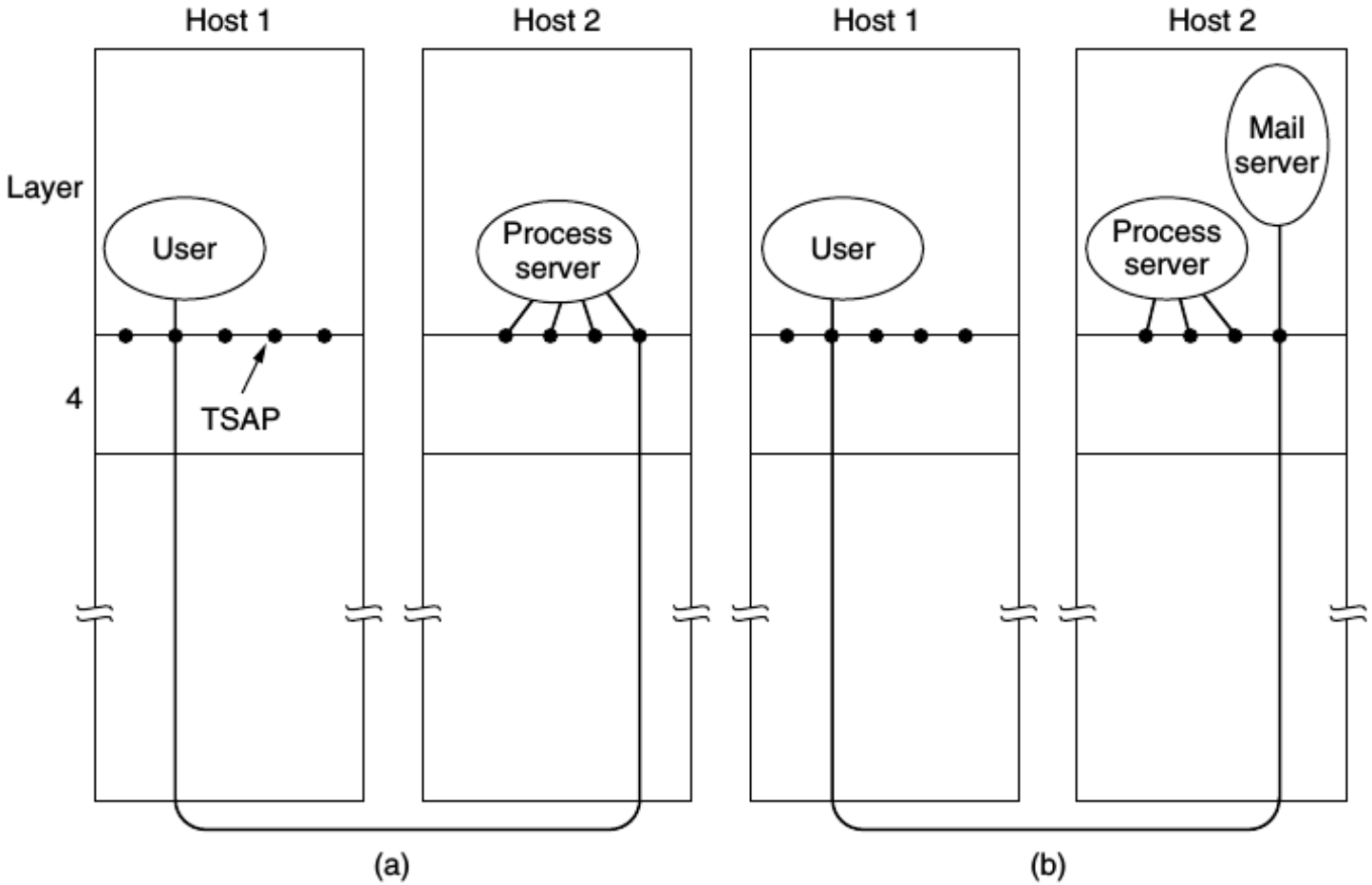


Figure 6-9. How a user process in host 1 establishes a connection with a mail server in host 2 via a process server.

Sujungimų sukūrimas (I)

- Atrodytų sukurti sujungimą yra paprastą - nusiuntei segmentą CONNECTION REQUEST, sulaukei CONNECTION ACCEPTED ir viskas.
- Problemų atsiranda nes datagramų tinklai gali sugadinti, duplikuoti, prarasti ir uždelsti paketus.
- Pavyzdžiui paketai gali būti užlaikyti tinkle, pasibaigti kliento nustatytas laukimo periodas ir negavus patvirtinimo jis juos išsiųs iš naujo.
- Serveris gaus tuos pačius duomenis du kartus ir gali atlikti operaciją du kartus, pavyzdžiui pervesti pinigus.

Sujungimų sukūrimas (II)

- Paketai gyvuoja ribotą laiką T.
- Jeigu lauksime T sekundžių prieš išsiųsdami duomenis iš naujo užtikrinsime, kad duomenys du kartus nenukeliaus gavėjui.
- Siuntėjas žymės paketus eilės numeriais taip, kad numeriai nesikartotų per laiko tarpą T.
- Ką daryti, jeigu vienas iš kompiuterių "nulūžta"? Tada ji nežinos kokio eilės numerio paketų tikėtis.
- Kiekvienas hostas naudoja laikrodį, nustatyti koks segmentų numerių intervalas tuo metu yra validus.

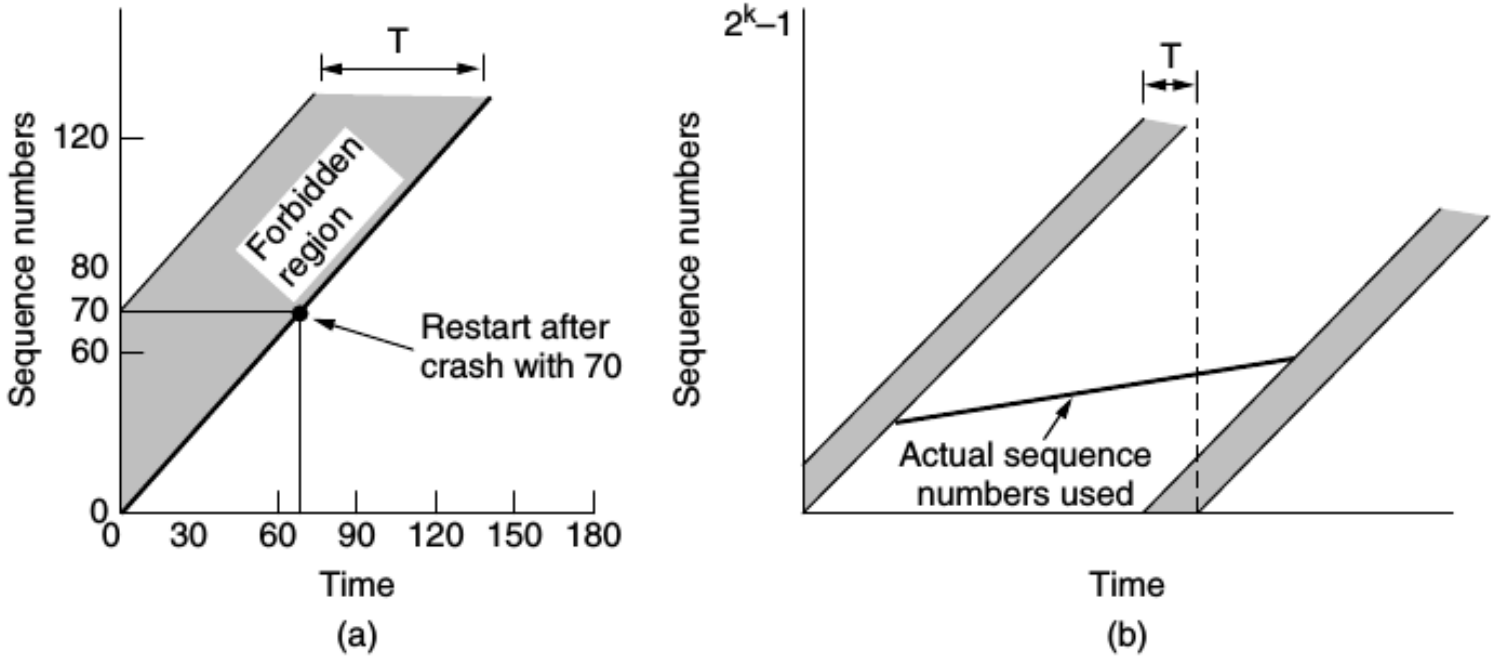
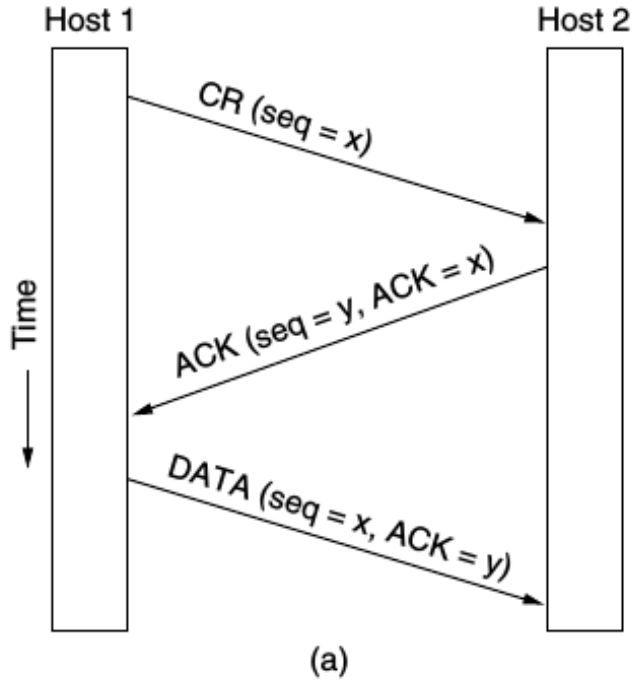


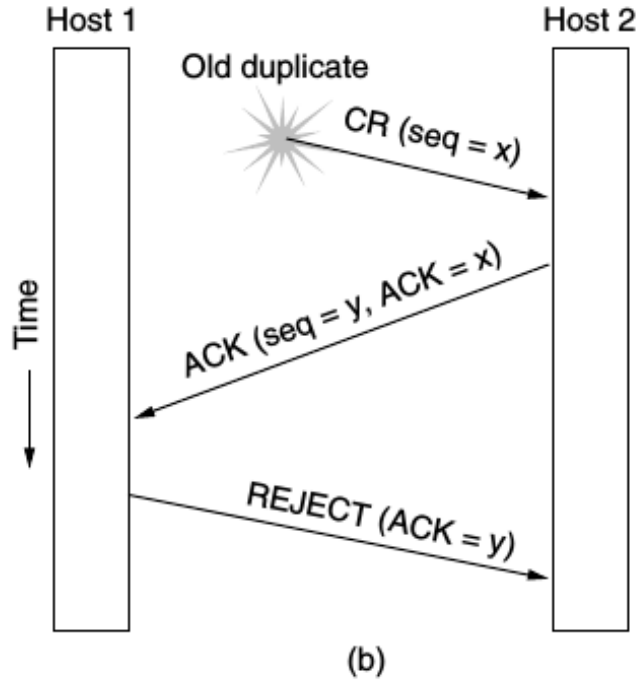
Figure 6-10. (a) Segments may not enter the forbidden region. (b) The resynchronization problem.

Three-way handshake

1. Hostas 1 parenka eilės numerį x ir išsiunčia CONNECTION REQUEST segmentą hostui 2.
2. Hostas 2 atsako ACK segmentu patvirtintamas x ir išsiunčia savo eilės numerį y.
3. Hostas 1 patvirtina hosto 2 eilės numerį.


Host 1
Host 2
CR (seq = x)
ACK (seq = y, ACK = x)
DATA (seq = x, ACK = y)
Time

(a)


Host 1
Host 2
Old duplicate
CR (seq = x)
ACK (seq = y, ACK = x)
REJECT (ACK = y)
Time

(b)

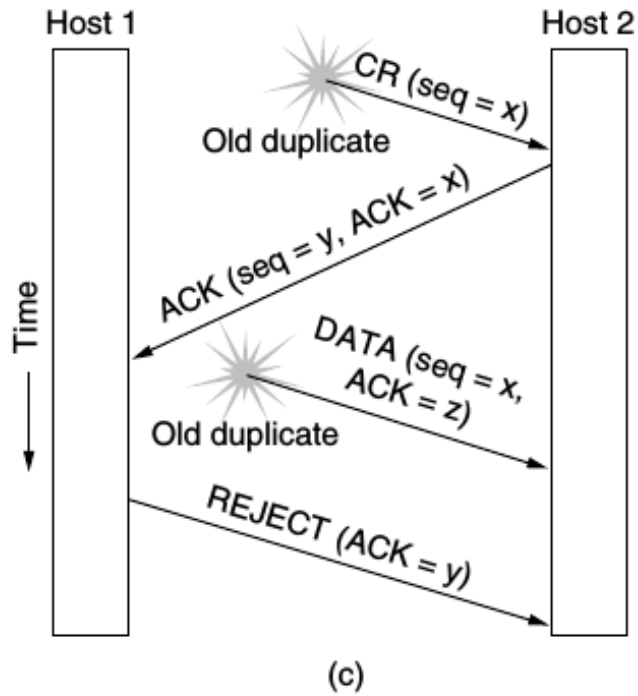


Figure 6-11. Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNECTION REQUEST. (a) Normal operation. (b) Old duplicate CONNECTION REQUEST appearing out of nowhere. (c) Duplicate CONNECTION REQUEST and duplicate ACK.

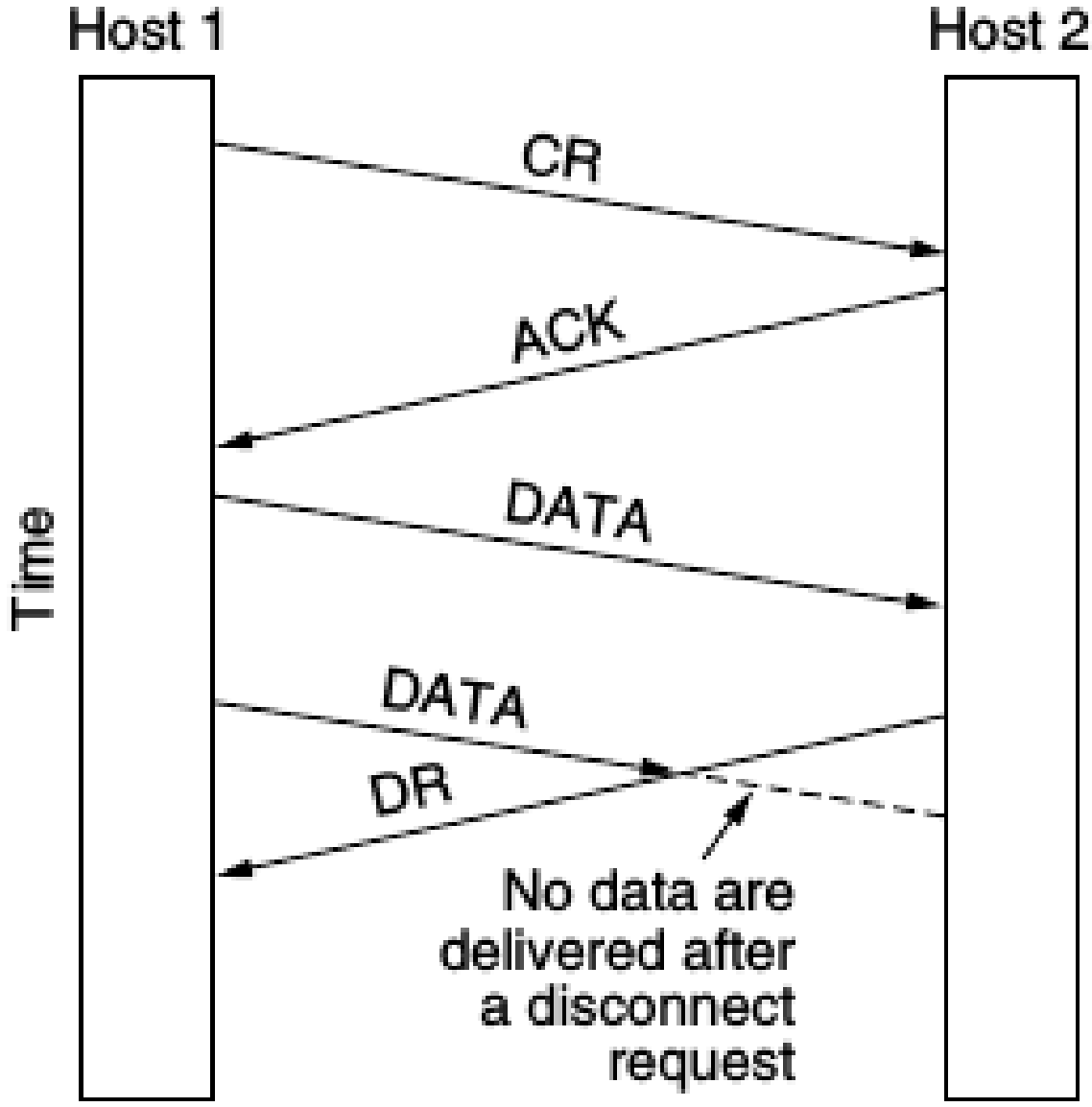


Figure 6-12. Abrupt disconnection with loss of data.

Dviejų armijų problema

- Balta armija didesnė negu kiekviena iš mėlynų armijų atskirai. Jeigu mėlynos armijos užpuls baltą kartu jos laimės, jeigu atskirai pralaimės.
- Mėlynom armijom reikia sinchronizuoti puolimą, tačiau vienintelis jų komunikacijos kanalas yra siųsti žmogų kuris gali būti pagautas ir pranešimas prarastas.
- Įsivaizduokite, kad mėlyna armija 1 išsiunčia pranešimą mėlynai armijai 2 pranešimą apie tai kad nori pulti tokią ir tokią dieną, taip pat gauna iš mėlynos armijos 2 patvirtinimą kad pranešimas gautas. Ar ataka įvyks sutartą dieną? Kodėl?
- Ar galima problemą išspręsti?

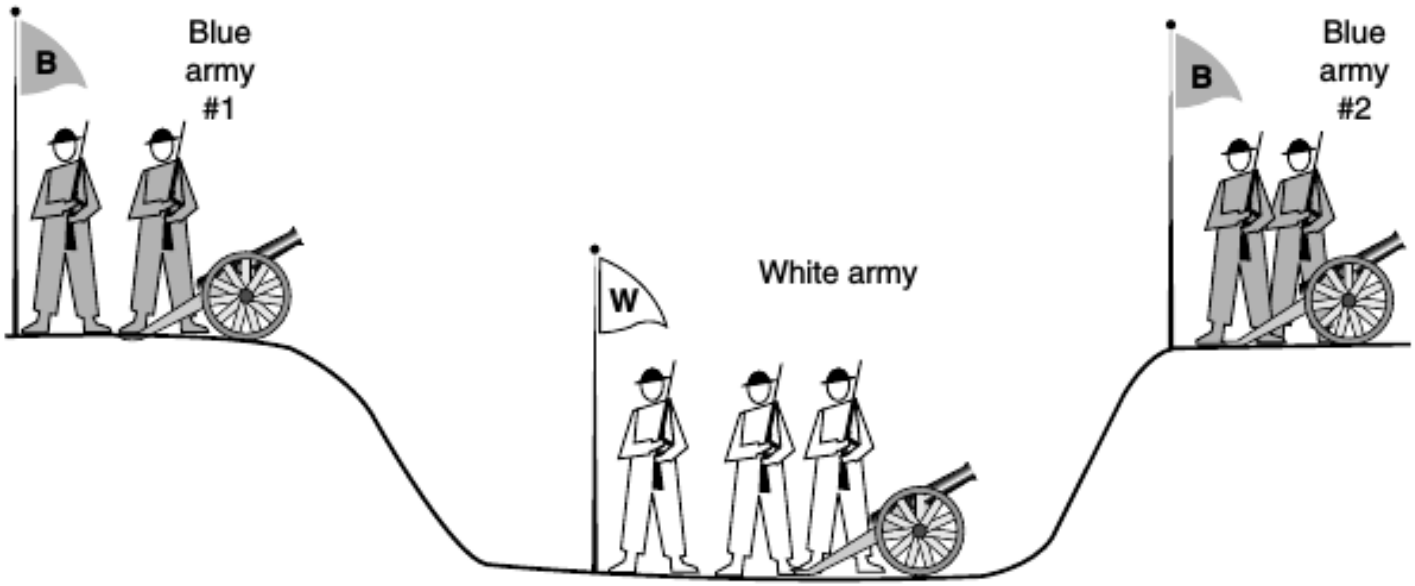


Figure 6-13. The two-army problem.

UDP

UDP

- UDP naudojamas siųsti IP datagramas be sujungimo.
- UDP aprašytas RFC 768.
- UDP segmentas sudarytas iš 8 baitų antraštės, po kurios seka duomenys.
- Antraštėje yra nurodyti siuntėjo ir gavėjo portai.
- Gavus UDP paketą, jo turinys yra perduodamas procesui kuriam yra priskirtas gavėjo portas.
- UPD segmentas nuo IP paketo skiriasi iš principo tik tuo, kad jame nurodyti portai.

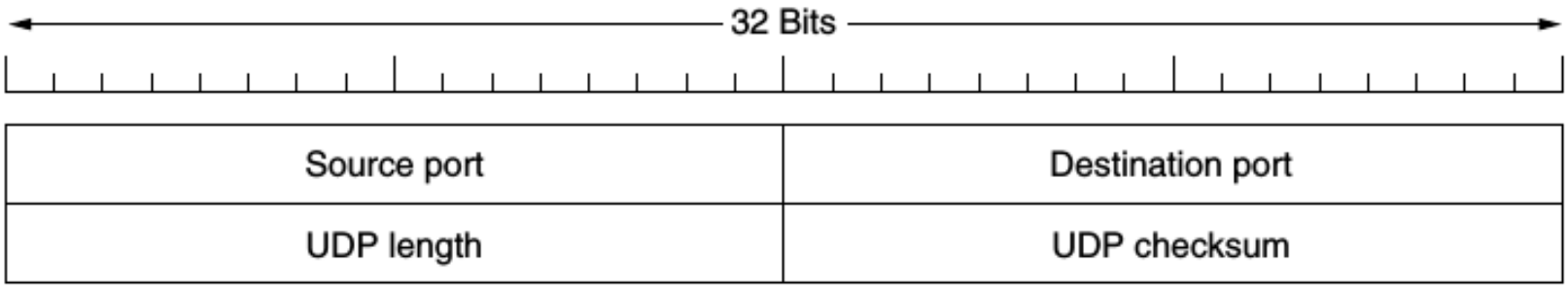


Figure 6-27. The UDP header.

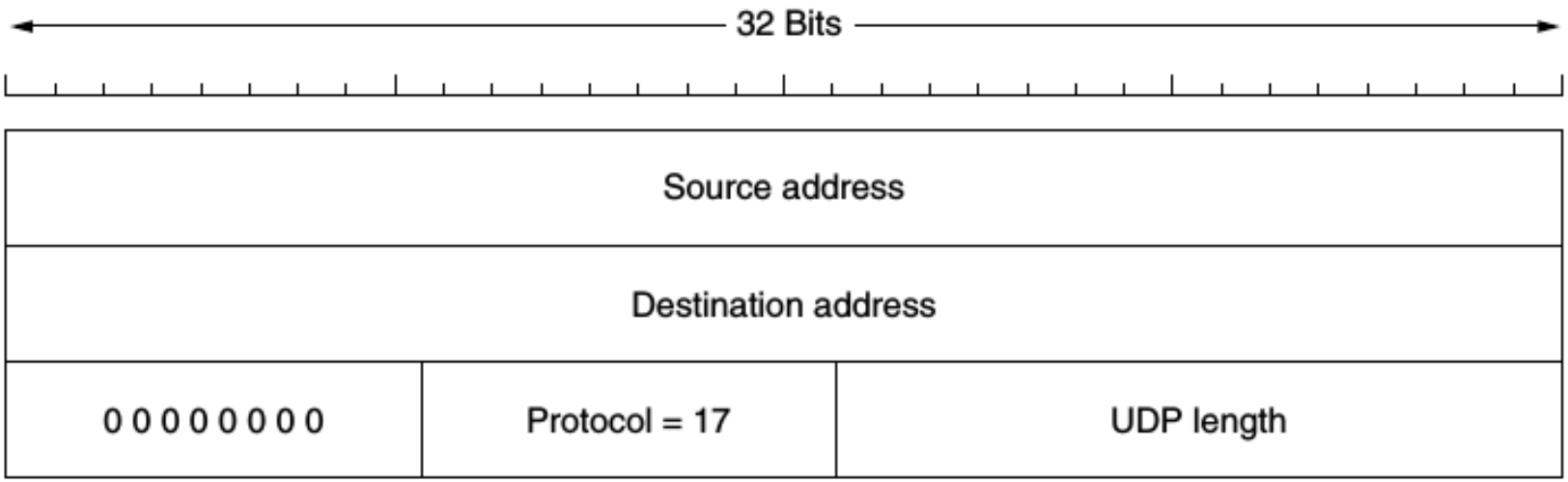


Figure 6-28. The IPv4 pseudoheader included in the UDP checksum.

Remote Procedure Call

- RPC leidžia vykdyti funkcijų iškvietimus nutolusiuose kompiuteriuose taip, kad jie atrodytų tarsi vykstantys lokaliai.
- Iškvietus funkciją išsiunčiamas UDP paketas ir laukiama atsakymo. Negavus kartojama kol gaunama.
- Kliento pusėje funkcija tiesiog užsiblokuoja.
- Ką daryti su argumentais kurie yra rodyklės?

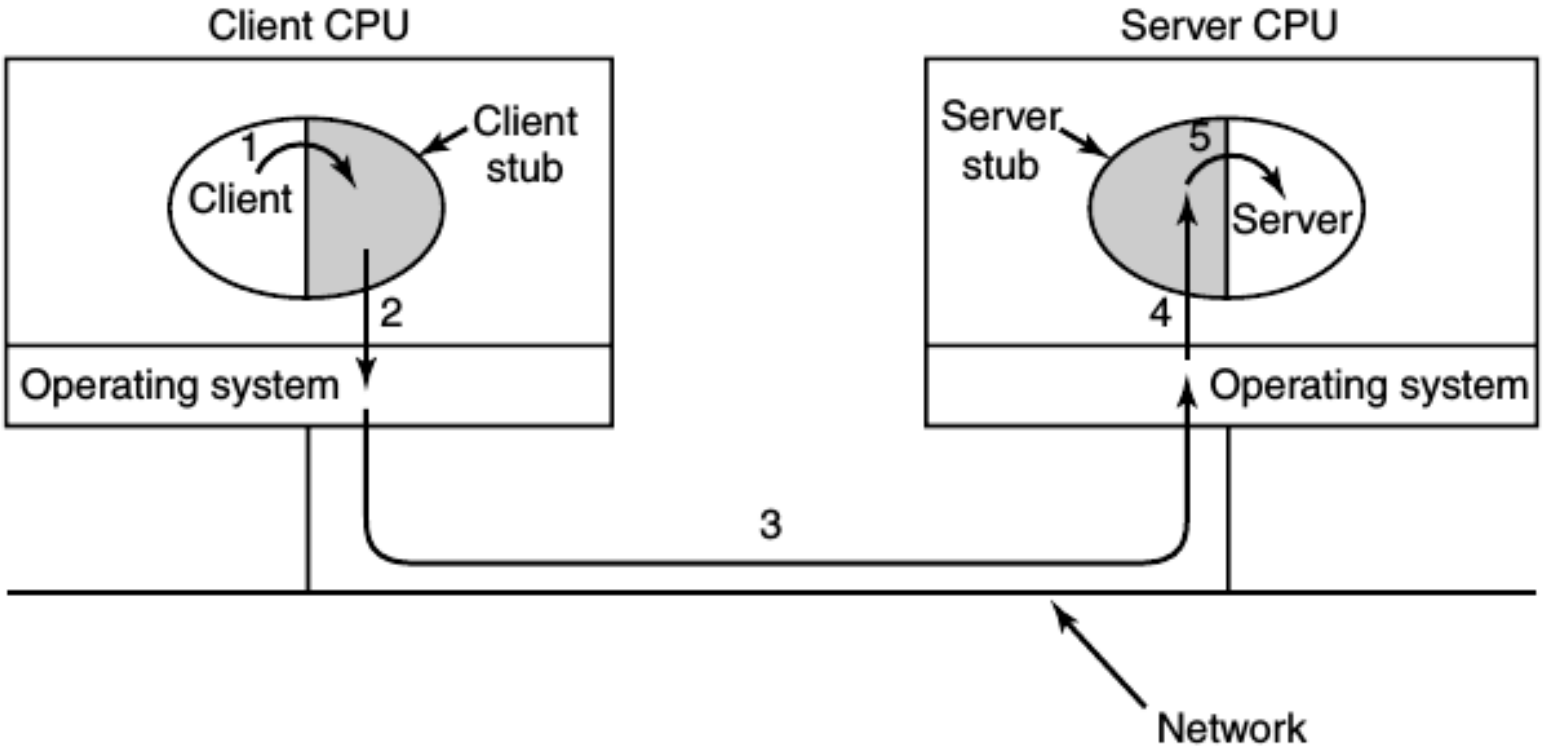


Figure 6-29. Steps in making a remote procedure call. The stubs are shaded.

Realaus laiko transporto protokolai

- UDP dažnai naudojamas realaus laiko multimedijos programų.
- Pavyzdžiai: interneto radijas, interneto telefonija, videokonferencijos ir t.t.
- **RTP (Real-time Transport Protocol)** yra vienas iš realaus laiko duomenų perdavimo protokolų, aprašytas RFC 3550.
- **RTP** tankina kelis įeinančius multimedijos srautus, formuoja **RTP** paketus ir siunčia duomenis naudodamas UDP.
- Gavėjas atlieka procesą atvirkščia tvarka.

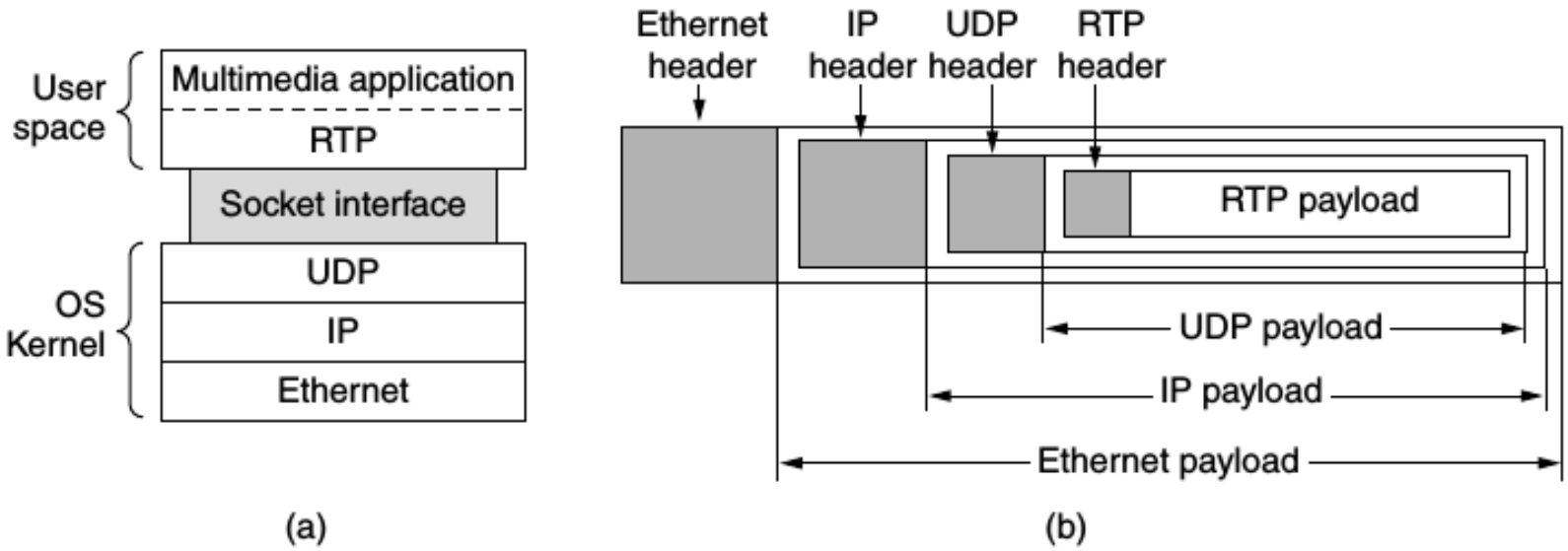


Figure 6-30. (a) The position of RTP in the protocol stack. (b) Packet nesting.

RTP

- RTP nenaudoja paketų patvirtinimo.
- Paketai numeruojami, pagal ką gavėjas mato ar trūksta paketų. Jeigu trūksta gali tiesiog praleisti video kadrą ar pan.
- RTP saugo informaciją apie tai koks signalas perduodamas ir koks kodavimas yra naudojamas.
- Saugoma informacija apie tai kur laike turi būti grojamas/rodomas to paketo turinys. Taip galima sinchronizuoti audio ir video perdavimą.

RTP antraštė

- P - ar paketas buvo papildytas iki reikiamo dydžio.
- X - naudojami antraštės praplėtimai.
- CC - kiek srautų yra perduodama (0 - 15).
- M - gali būti naudojamas programų žymėti, pavyzdžiui, video kadro pradžiai.
- Payload type - kokio tipo informacija perduodama (MP3 ir pan.)
- Sequence number - didinamas išsiuntus kiekvieną naują paketą.
- Synchronization source identifier - kuriam srautui priklauso paketas.

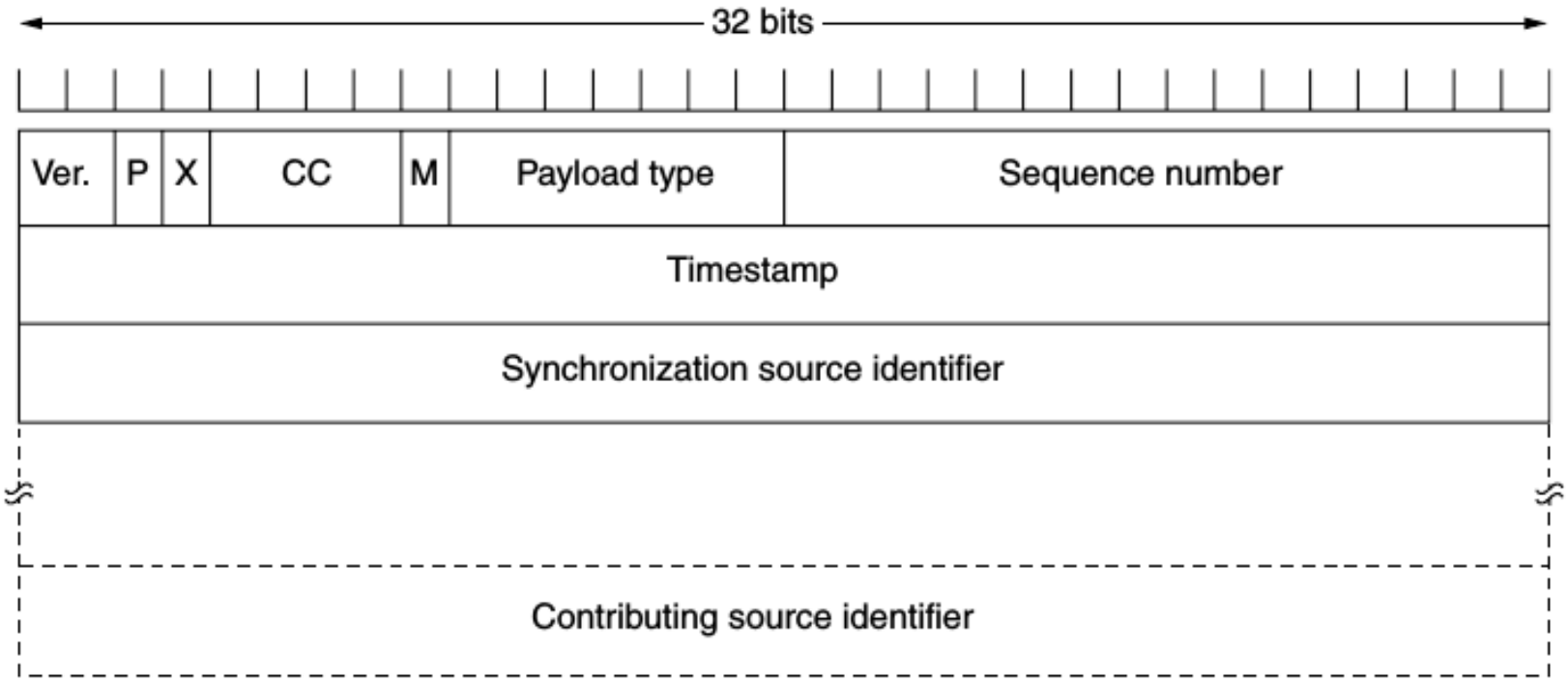


Figure 6-31. The RTP header.

RTCP - The Real-time Transport Control Protocol

- Naudojamas gryžtamajam ryšiui kurio reikia, kad audio ir video siuntėjas matytų kokiu greičiu gali perduoti signalus.
- Gavęs šią informaciją siuntėjas gali pakeisti duomenų suspaudimo nustatymus taip, kad informaciją būtų galima perduoti realiu laiku.
- RTCP taip pat valdo sinchronizavimą tarp duomenų srautų.
- RTCP leidžia priskirti srautam pavadinimus.

Duomenų perdavimas su buferizacija ir drebėjimo kontrole

- Gavus multimedijos duomenų srautą jo atvaizdavimas naudotojui bus vykdomas ne tuo pačiu metu kaip buvo gauti RTP paketai.
- Taip yra dėl to, kad paketai atvyks su skirtingais uždelsimais ir galimai ne ta tvarka kuria buvo išsiųsti.
- Paketai yra buferizuojami, kad būtų išvengta drebėjimo.
- Dėl buferizacijos yra vėlinamas visas audio/video duomenų srautas.
- Ilgas uždelsimas gali netikti gyvam vaizdo ir garso perdavimui, pavyzdžiui videokonfernecijoms.

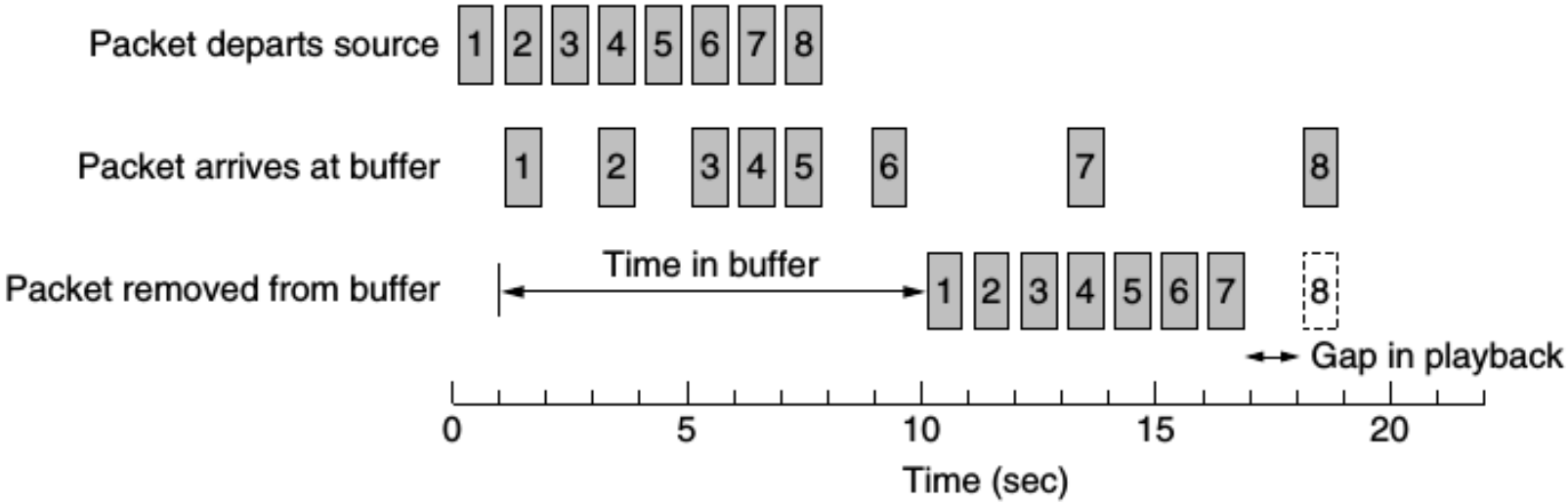


Figure 6-32. Smoothing the output stream by buffering packets.

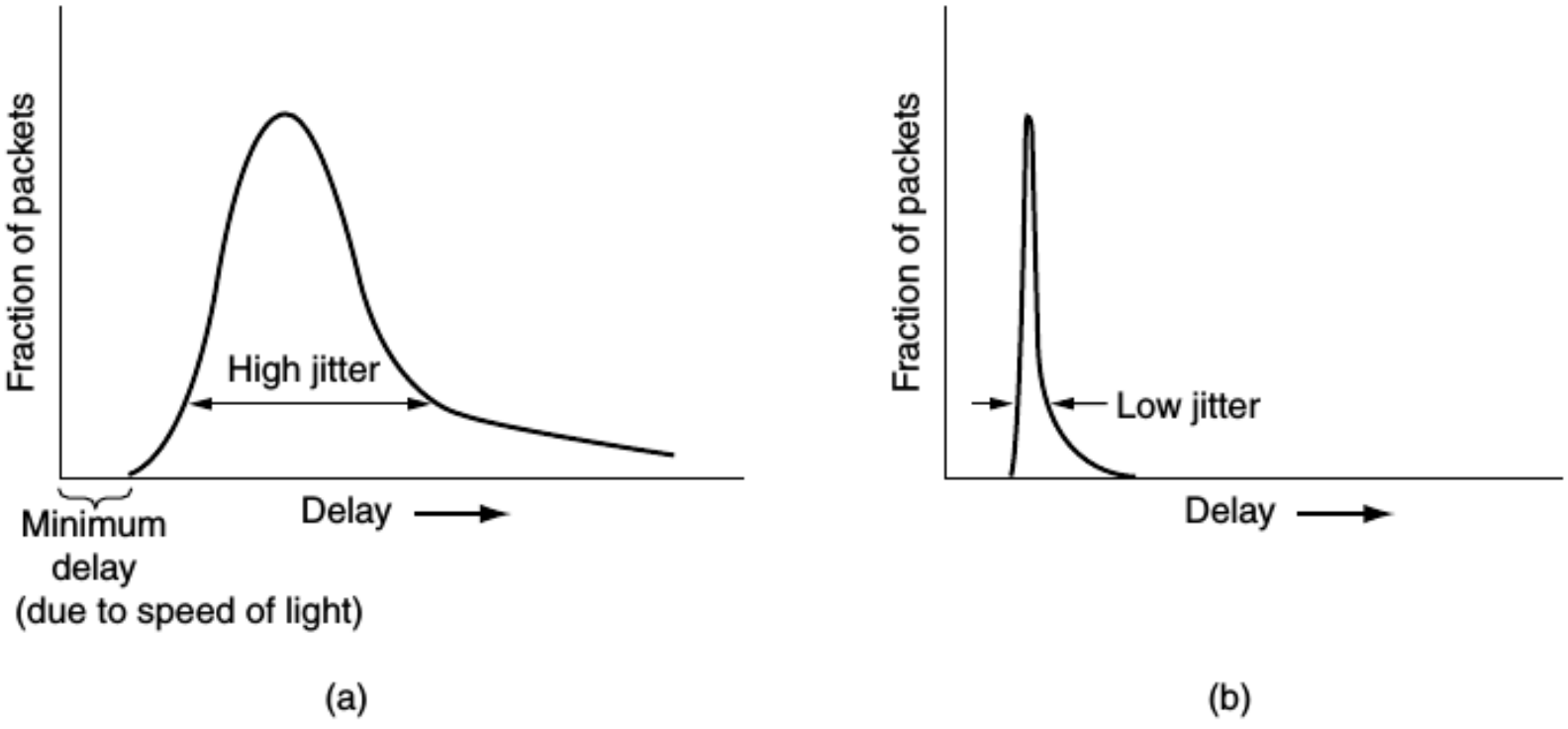


Figure 6-33. (a) High jitter. (b) Low jitter.

TCP

TCP (I)

- **TCP (Transmission Control Protocol)** skirtas kai duomenis reikia nugabenti be klaidų ir ta tvarka kuria jie buvo išsiųsti.
- Kadangi tinklo sluoksnis jokių garantijų paketų sėkmingam pristatymui neteikia.
- **TCP** darbas yra užtikrinti, kad paketai bus nugabenti ir gautus juos vėl surinkti į tokius duomenis kokie buvo išsiųsti.
- **TCP** naudoja jau aptartus socketus sudarytus iš IP adreso ir 16-bitų skaičiaus vadinamo portu.
- Sudarant **TCP** sujungimą turi būti užmegztas ryšys tarp socketų dviejuose kompiuteriuose. Vienas socketas gali būti naudojamas keliems sujungimams vienu metu.

TCP (II)

- Visi **TCP** sujungimai yra full-duplex ir point-to-point. **TCP** nepalaiko broadcast ir multicast duomenų perdavimo.
- **TCP** sujungimas yra baitų srautas. Jeigu procesas rašo į socketą 4 kartus po 512 baitų jie gali būti nugabenti kaip du 1024 baitų gabalai arba vienas 2048.
- Gavėjas neturi galimybės nustatyti kokio ilgio duomenis rašė siuntėjas.
- Kitaip tariant socketai primena failus. TCP niekaip neinterpretuoja siunčiamų baitų srauto.

Port	Protocol	Use
20, 21	FTP	File transfer
22	SSH	Remote login, replacement for Telnet
25	SMTP	Email
80	HTTP	World Wide Web
110	POP-3	Remote email access
143	IMAP	Remote email access
443	HTTPS	Secure Web (HTTP over SSL/TLS)
543	RTSP	Media player control
631	IPP	Printer sharing

Figure 6-34. Some assigned ports.

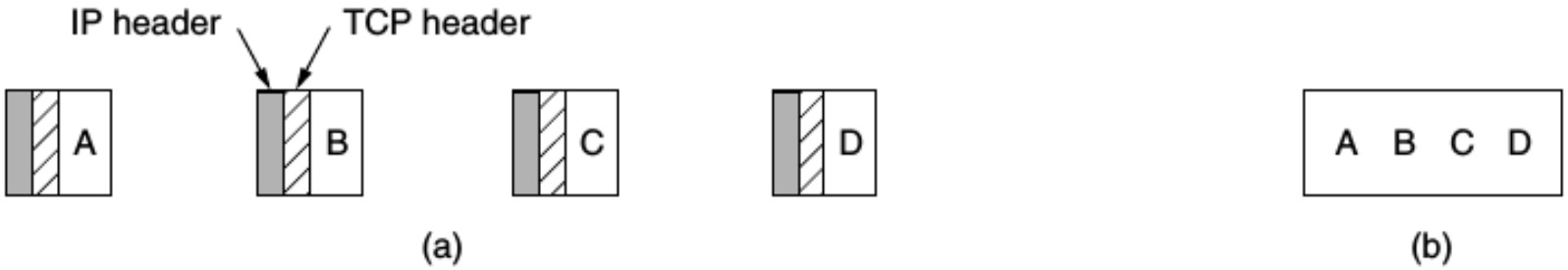


Figure 6-35. (a) Four 512-byte segments sent as separate IP datagrams. (b) The 2048 bytes of data delivered to the application in a single READ call.

TCP protokolas

- Kiekvienas baitas turi savo eilės numerį - 32 bitų skaičiaus pavidalu.
- TCP segmentas sudarytas iš 20 baitų antraštės po kurios seka nulis arba daugiau duomenų baitų.
- Segmento dydis ribojamas IP paketo dydžio bei duomenų perdavimo kanalo naudojamo kadro dydžio.
- IP paketai gali būti fragmentuojami, tačiau dėl to nukenčia duomenų perdavimo efektyvumas ir to stengiamasi išvengti.
- Naudojamas slenkančio lango protokolas segmentų patvirtinimams, juos apdarėme paskaitoje apie duomenų kanalo sluoksnį.

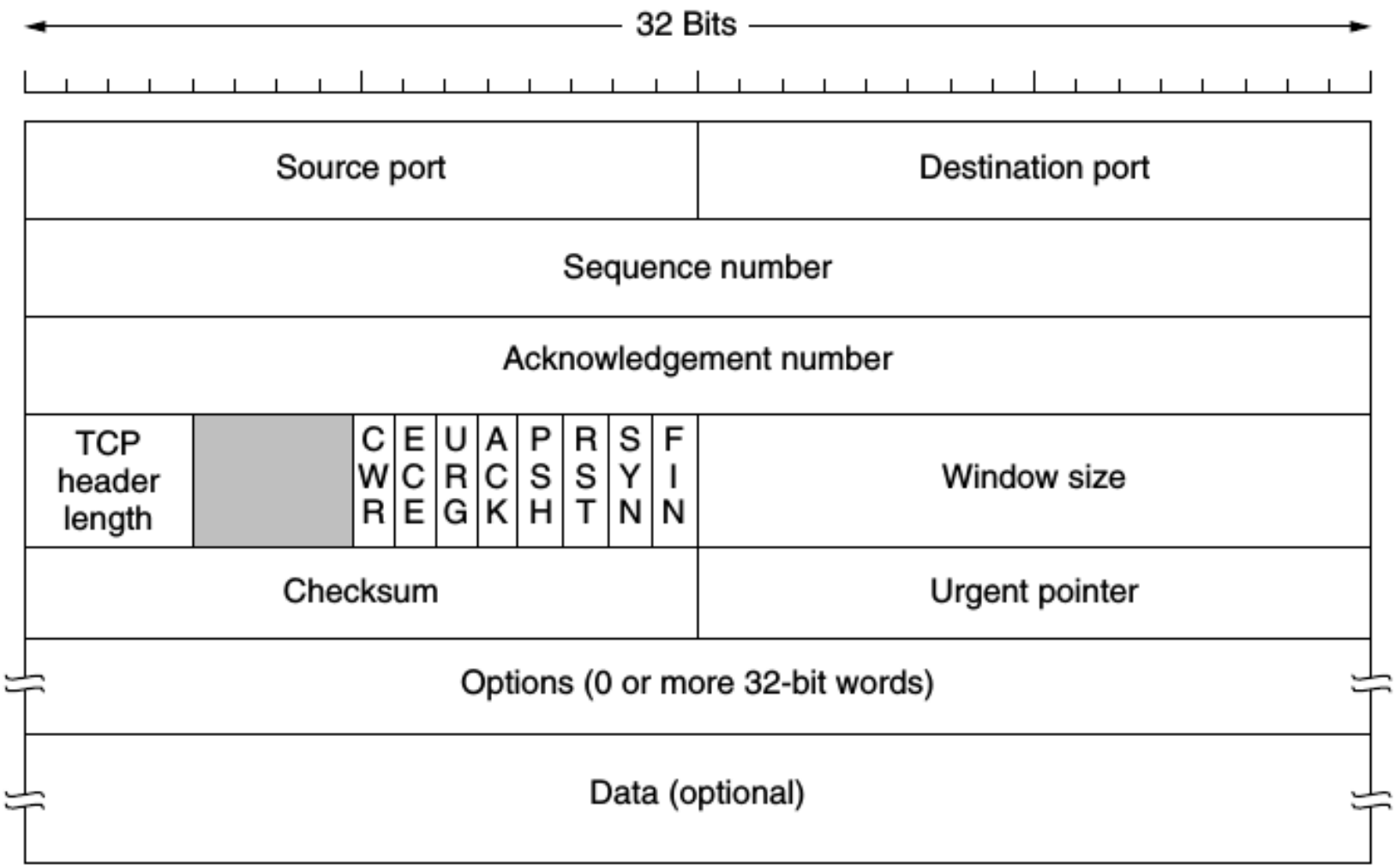


Figure 6-36. The TCP header.

TCP antraštė (I)

- Source port, destination port - siuntėjo ir gavėjo portai.
- Sujungimą sudaro penki dalykai - siuntėjo ir gavėjo portai, siuntėjo ir gavėjo IP adresai ir protokolas (TCP).
- Sequence number - pirmo segmento baito eilės numeris.
- Acknowledgement number - kokio baito numerio toliau tikisi gavėjas.
- TCP header length - kiek 32 bitų žodžių yra antraštėje.
- CWR ir ECE - naudojami pasakyti duomenų siuntėjui kad reikia sulėtinti duomenų siuntimą.
- URG - Naudojamas Urgent pointer.
- ACK - Jei ACK yra nulis segmente nėra patvirtinimo.

TCP antraštė (II)

- PSH - signalas gavėjui, kad reikia duomenis perduoti iš karto gavus o ne buferizuoti.
- RST - kažkas negerai su sujungimu.
- SYN - naudojamas kurti sujungimams. SYN $= 1$, ACK $= 0$ - CONNECTION REQUEST, SYN $= 1$, ACK $= 1$ - CONNECTION ACCEPTED.
- FIN - nutraukti sujungimą.
- Window size - langa dydis naudojamas slenkančio lango protokole.

TCP antraštės papildomi nustatymai

- Naudojamas tipas-ilgis-reikšmė formatas.
- **MSS (Maximum Segment Size)** - kokio dydžio segmentus gali priimti hostas.
- Window scale - leidžia siuntėjui ir gavėjui susitarti dėl lango dydžio.
- Timestamp - pridedama laiko informacija prie segmento, naudojama matuoti vėlinimui, kad būtu lengviau nustatyti ar paketas pamestas.

TCP sujungimo sukūrimas

1. Viena pusė pasyviai laukia sujungimo atlikdama LISTEN ir ACCEPT tokia tvarka.
2. Kita pusė naudoja CONNECT primityvą, nurodydama IP adresą ir portą prie kurio nori jungtis. CONNECT išsiunčia TCP segmentą su įjungtu SYN bitu ir išjungtu ACK bitu ir laukia atsako.
3. Gavusi tokį segmentą viena pusė patikrina ar tokiu porto užsiregistravęs procesas atliko LISTEN primityvą, jei ne išsiunčia segmentą su įjungtu RST bitu.
4. Jei kažkoks procesas priima sujungimą jis išsiunčia TCP segmentą su įjungtu SYN bitu ir įjungtu ACK bitu.

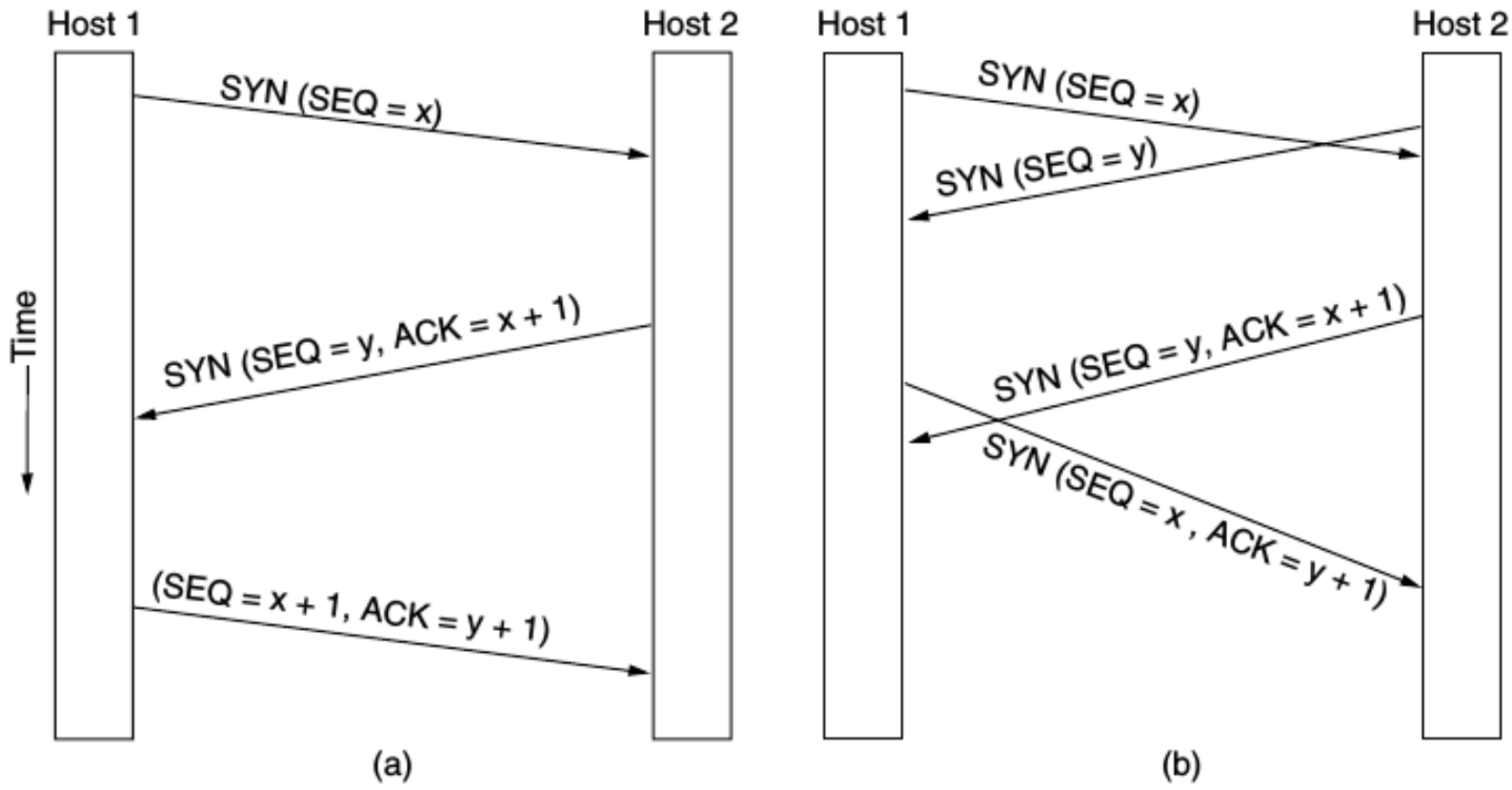


Figure 6-37. (a) TCP connection establishment in the normal case. (b) Simultaneous connection establishment on both sides.

TCP sujungimų valdymo modelis (I)

- Kiekvienas sujungimas iš pradžių yra CLOSED būsenos. Iš tos būsenos išeinama arba atidarant sujungimą pasyviai (LISTEN) arba aktyviai (CONNECT). Kai kita pusė atlieka priešingo tipo atidarymą sujungimas laikomas ESTABLISHED. Kai baigiama sujungimas yra CLOSED.
- Diagramoje kliento keliai pažymėti stora vientisa linija, serverio trūkia o neįprasti keliai plona linija.
- Iš kliento pusės:
  1. Kai programa kliento pusėje įvykdo CONNECT užklausą, lokali TCP esybė sukuria sujungimo įrašą, pažymi, kad jis yra SYN SENT būsenos ir išsiunčia SYN segmentą.
  2. Sulaukus SYN+ACK, TCP išsiunčia galutinį ACK ir pereina į ESTABLISHED būseną.
  3. Kai programa baigia darbą įvykdo CLOSE. TCP išsiunčia FIN segmentą ir laukia atitinkamo ACK. Gavus ACK pereina į FIN WAIT 2 kuris reiškia, kad viena sujungimo kryptis uždaryta. Užsidarius kitai puse ateina FIN kuris irgi patvirtinamas. TCP palaukia du kartus tiek koks yra segmento gyvavimo laikas.

TCP sujungimų valdymo modelis (II)

Iš serverio pusės sujungimas:

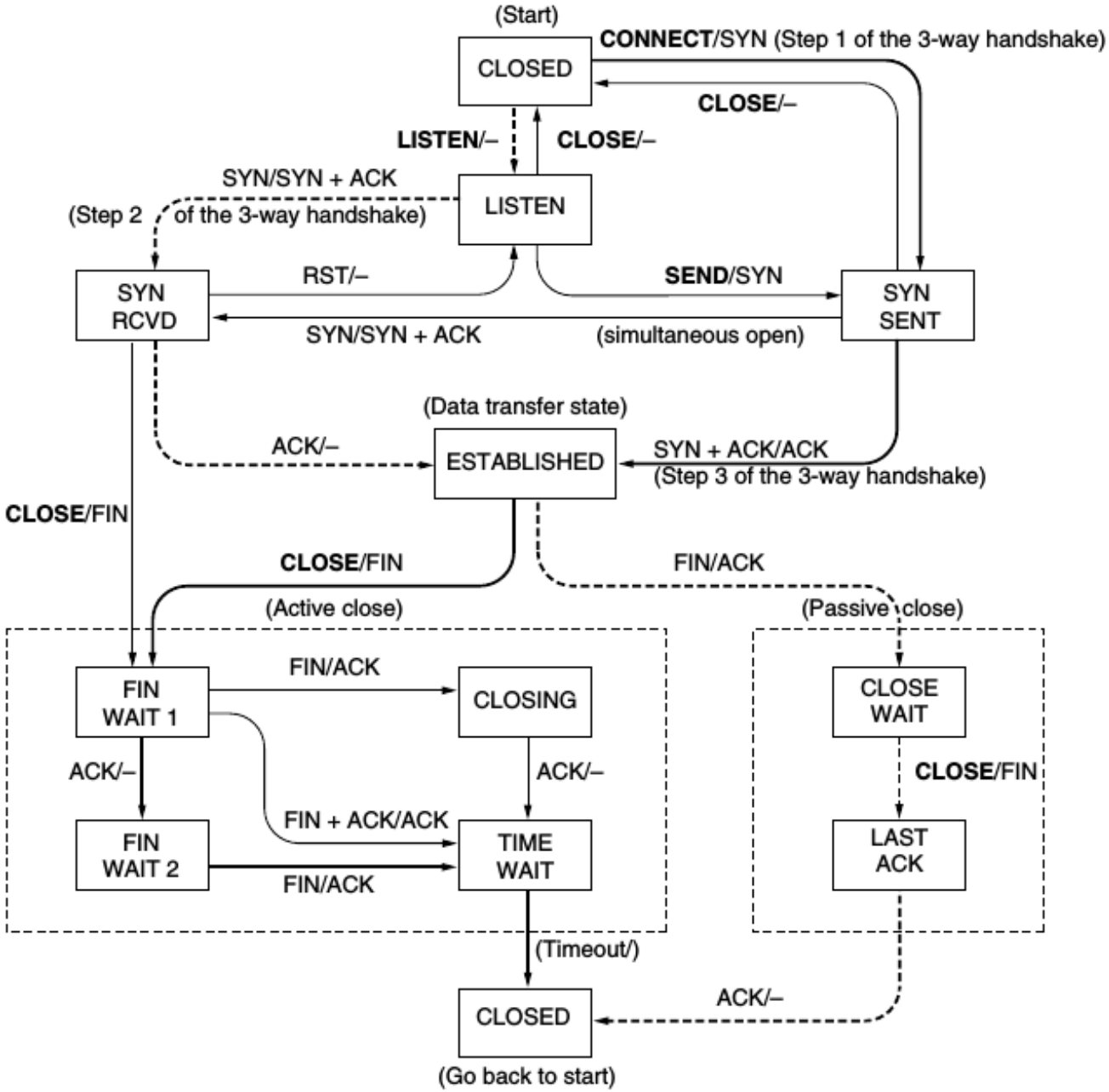
1. Serveris atlieka LISTEN ir laukia norinčių prisijungti.
2. Kai atvyksta SYN jis yra patvirtinamas ir pereinama į SYN RCVD būseną.
3. Kai serverio SYN patvirtinamas jis pereina į ESTABLISHED būseną.

Iš serverio pusės sujungimo nutraukimas:

1. Kai klientas baigė perduoti duomenis jis įvykdo CLOSE, dėl ko serverį pasiekia FIN segmentas. Serveryje dėl to įvyksta signalas.
2. Kai jis taip pat atlieka CLOSE, FIN išsiunčiamas klientui.
3. Kai ateina patvirtinimas, kad klientas gavo FIN, sujungimas sunaikinamas, atmintis atlaisvinama.

State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIME WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

Figure 6-38. The states used in the TCP connection management finite state machine.


(Start)
CLOSED
CONNECT/SYN (Step 1 of the 3-way handshake)
CLOSE/–
LISTEN/–
CLOSE/–
SYN/SYN + ACK
(Step 2 of the 3-way handshake)
LISTEN
RST/–
SEND/SYN
SYN RCVD
SYN SENT
SYN/SYN + ACK
(simultaneous open)
(Data transfer state)
ACK/–
ESTABLISHED
SYN + ACK/ACK
(Step 3 of the 3-way handshake)
CLOSE/FIN
CLOSE/FIN
FIN/ACK
(Active close)
(Passive close)
FIN WAIT 1
FIN/ACK
CLOSING
CLOSE WAIT
ACK/–
ACK/–
CLOSE/FIN
FIN + ACK/ACK
FIN WAIT 2
TIME WAIT
LAST ACK
FIN/ACK
(Timeout/)
ACK/–
CLOSED
(Go back to start)