



# LTAT.06.001

# Operatsioonisüsteemid

Võrk II

Artjom Lind

[artjom.lind@ut.ee](mailto:artjom.lind@ut.ee)

14.11.2025

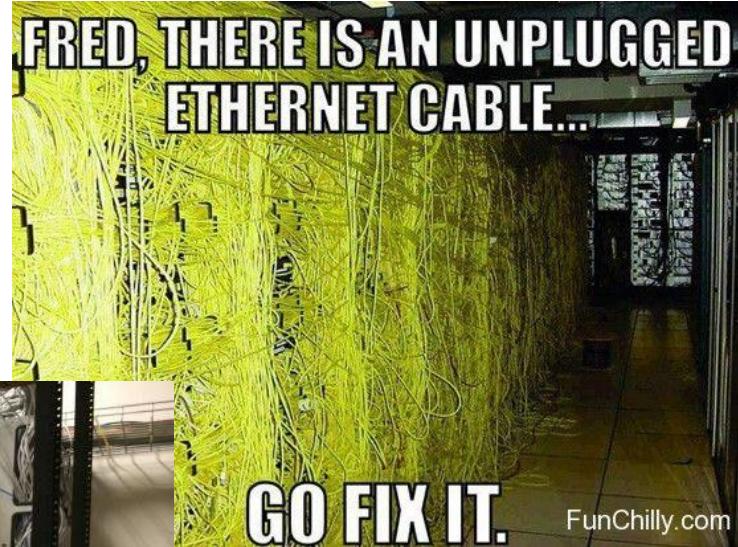
# Loengu plaan

- Võrguprotokoll
- OSI mudell
- Võrguliidesed
- Edastusviisid
- Ethernet
- IP
- TCP
- UDP
- NAT
- IPv6
- Sockets



**"IT SAYS NETWORK CABLE  
UNPLUGGED"**

[www.callcentermemes.com](http://www.callcentermemes.com)



# Võrguprotokoll

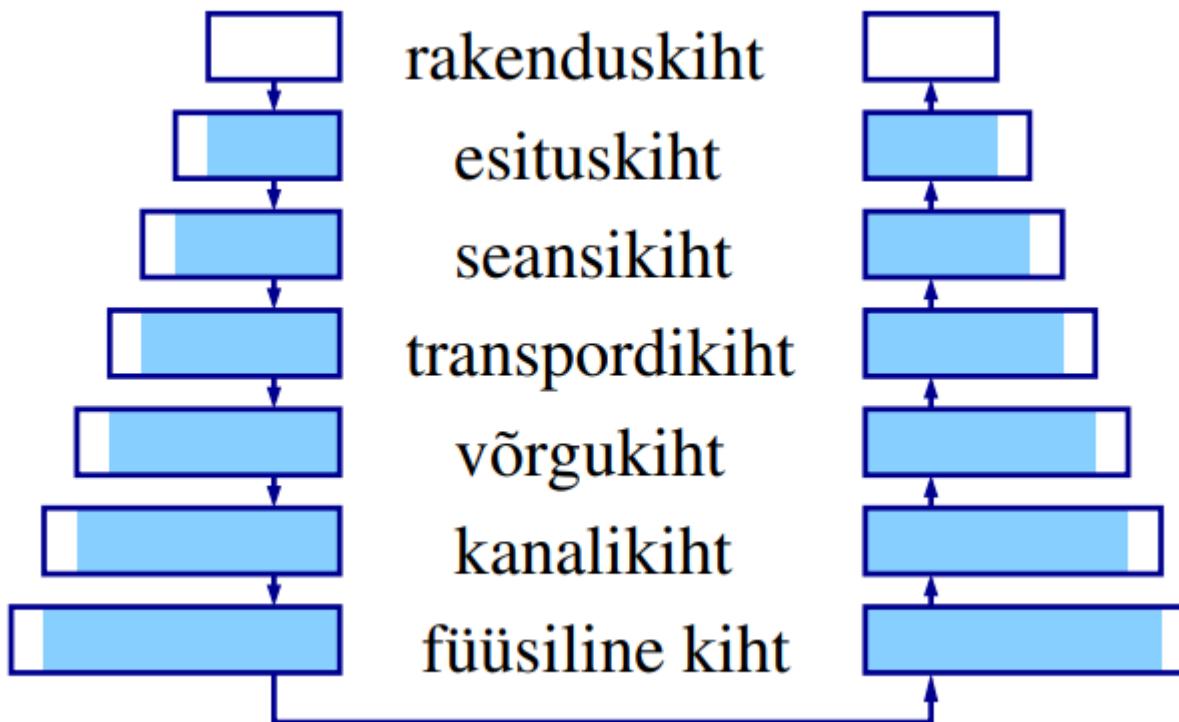
**Võrguprotokoll** on reeglite ja protseduuride kogum, mis määratlevad, **kuidas andmed liiguval läbi võrkude**. Need protokollid võimaldavad erinevatel võrguseadmetel, nagu arvutid ja serverid, omavahel suhelda ning andmeid edastada. **Protokollid määratlevad, kuidas andmeid pakendatakse, saadetakse, vastu võetakse ja dekodeeritakse.**

Näiteid võrguprotokollidest:

- TCP (Transmission Control Protocol) - Tagab andmete usaldusväärse edastamise, korrigeerides vigu ja haldades andmevoogu.
- IP (Internet Protocol) - Vastutab andmepakettide saatmise eest ühest arvutist teise, määrates siht- ja läheeadressid.
- HTTP (Hypertext Transfer Protocol) - Kasutatakse veebisaitide sisu edastamiseks veebiserveritest veebistrauseritesse.
- SMTP (Simple Mail Transfer Protocol) - Võimaldab e-kirjade saatmist ja vastuvõttu.
- FTP (File Transfer Protocol) - Kasutatakse failide ülekandmiseks arvutite ja serverite vahel.

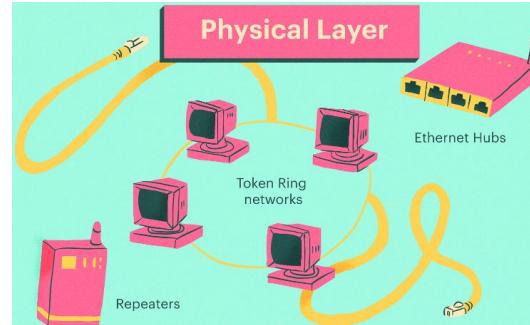
Iga protokoll täidab oma kindlat rolli võrgusuhtluses, tagades andmevahetuse sujuvuse ja efektiivsuse.

# ISO/OSI 7-kihiline mudel

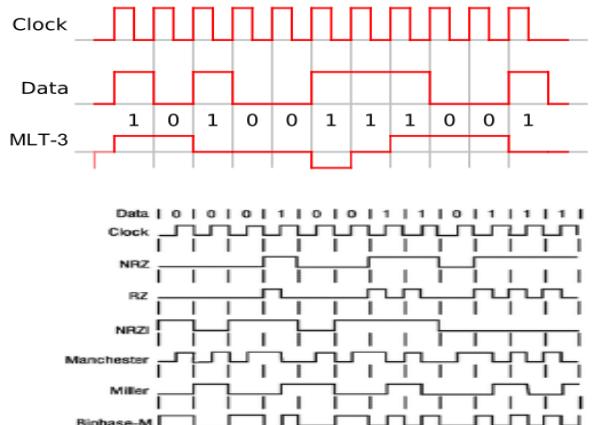


# Füüsiline kiht (physical layer)

- **Eesmärgid:** Füüsiline kiht võimaldab toorandmebitide edastamist füüsilise meedia kaudu.
  - a. Kahe seadme vaheline side, kandja kujundus, elektrisignaalide määratlemine
- **Minimaalne ühik:** 1 bitt
- **Riistvara Tugi:** See kiht kasutab kaableid, hube, retranslaatoreid, võrgukaarte ja modemeid andmeedastuseks.
  - a. Cat5e või Cat6 Etherneti kaableid või Wi-Fi ruuter, switch, hub, Ethernet adapter, ADSL modem, Wi-Fi adapter
- **Operatsioonisüsteemi Tugi:** Operatsioonisüsteemid sõltuvad füüsilise kihi riistvarast ja püsivarast andmeedastuseks.
  - a. Windows või Linux kasutab võrgukaardi draivereid füüsilise kihi funktsioneerideks.
- **Draiverite Tugi:** Füüsiline kihi seadmed vajavad nende tööks spetsiifilisi draivereid.
  - a. Intel Gigabit Etherneti võrgukaardi draiverid (e1000)
  - b. Broadcom, Realtek, Atheros, Qualcomm
- **API Tugi:** Füüsiline kihi API-d on tavaliselt madala taseme ja ei ole lõppkasutajale nähtavad.
  - a. Ethernet controller low level API
- **Protokollid:** Füüsiline kiht tegeleb andmete edastamise füüsiliste standardite ja spetsifikatsioneeridega, mitte kõrgema taseme protokollidega.
  - a. RZ, NRZ, Manchester, Miller, MLT-3

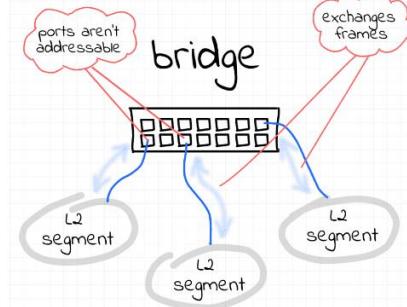
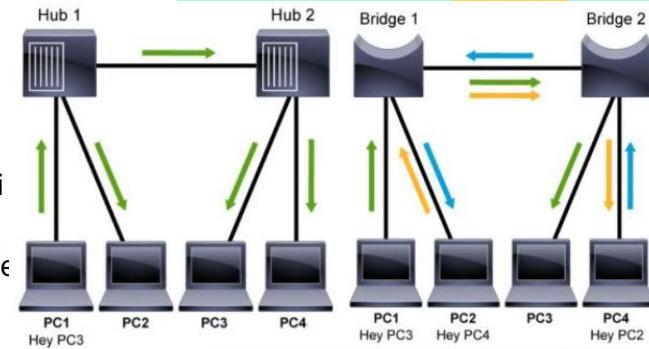
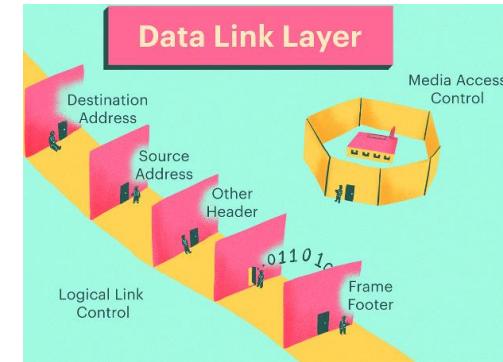


Cat5e Wire Diagram for T568B (Straight Through Cable)				
RJ45 Pin #	Wire Color (T568A)	Wire Diagram (T568A)	10Base-T Signal 100Base-TX Signal	1000Base-T Signal
1	White/Orange		Transmit+	Bi DA+
2	Orange		Transmit-	Bi DA-
3	White/Green		Receive+	Bi DB+
4	Blue		Unused	Bi DC+
5	White/Blue		Unused	Bi DC-
6	Green		Receive-	Bi DB-
7	White/Brown		Unused	Bi DD+
8	Brown		Unused	Bi DD-



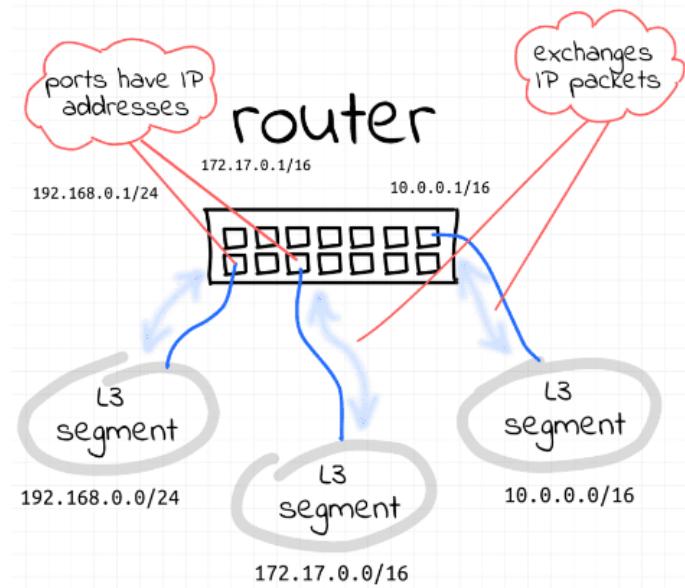
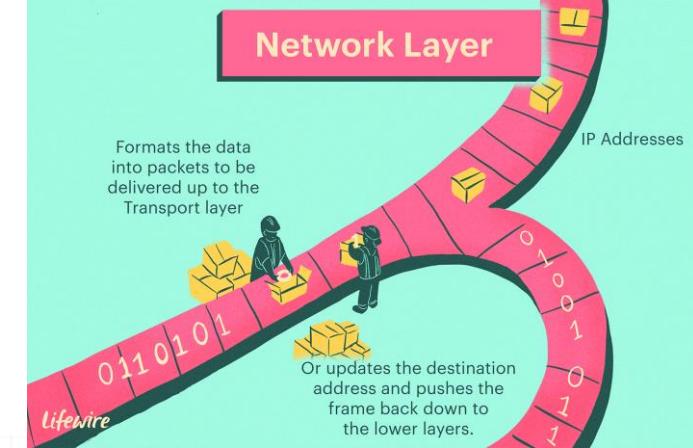
# Kanalikiht (Link layer)

- **Eesmärgid:** Andmesidelinkkiht tagab otseühendatud sõlmede vahelise usaldusväärse andmeedastuse ja veaparanduse.
  - a. Tüüpiline LAN over Ethernet: 1 switch ja mittu arvutit või 1 wi-fi AP ja mitu arvutit
- **Minimaalne ühik:** 1 kaader (frame)
- **Riistvara Tugi:** Kasutab võrgulüliteid, sildu ja võrgukaarte.
  - a. Switch, ethernet adapter, wi-fi APN, wi-fi adapter
- **Operatsioonisüsteemi Tugi:** Operatsioonisüsteemid kasutavad andmesidelinkkihi jaoks spetsiifilisi draivereid ja tarkvara.
  - a. Windows ja Linux kasutavad võrguadapteri draivereid linkkihi funktsioonide jaoks.
- **Draiverite Tugi:** Näiteks Etherneti lülitri draiverid.
  - a. Intel, Broadcom, Realtek, Atheros, Qualcomm
- **API Tugi:** Pakub API-sid võrgu seadmete juhtimiseks, nagu libpcap võrguliikluse jälgimiseks.
  - a. libpcap - võimaldab võrguliikluse jälgimist ja analüüsimist.
- **Protokollid:** Andmesidelinkkihi protokollid hõlmavad Etherneti, PPP ja MAC aadressimist.
  - a. Etherneti protokoll (IEEE 802.3) ja MAC (Media Access Control) aadressimine



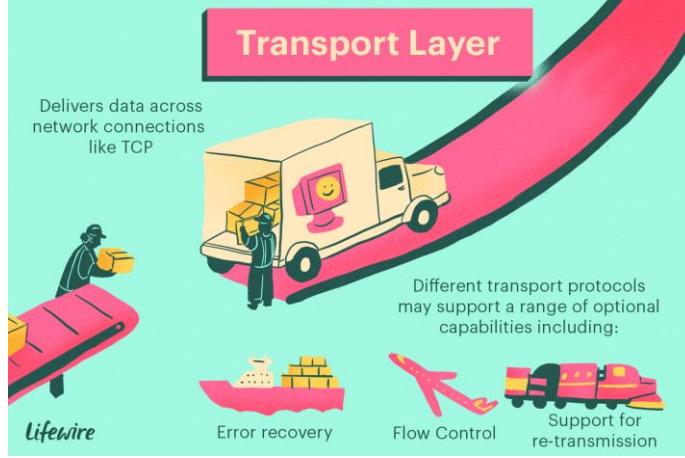
# Võrgukiht (Network layer)

- Tee otsimine võrgus rohkem kui kahe seadme vahel
- Kommuteerimine ja marsruudi leidmine
  - Igale paketile eraldi
  - Ühe korra virtuaalse kanali loomisel
- Võrgusõlmede addresseerimine
- Pakettide edastamine erinevate võrkude vahel
- Pakettide tükeldamine ja kokkupanek



# Transpordikiht (Transport layer)

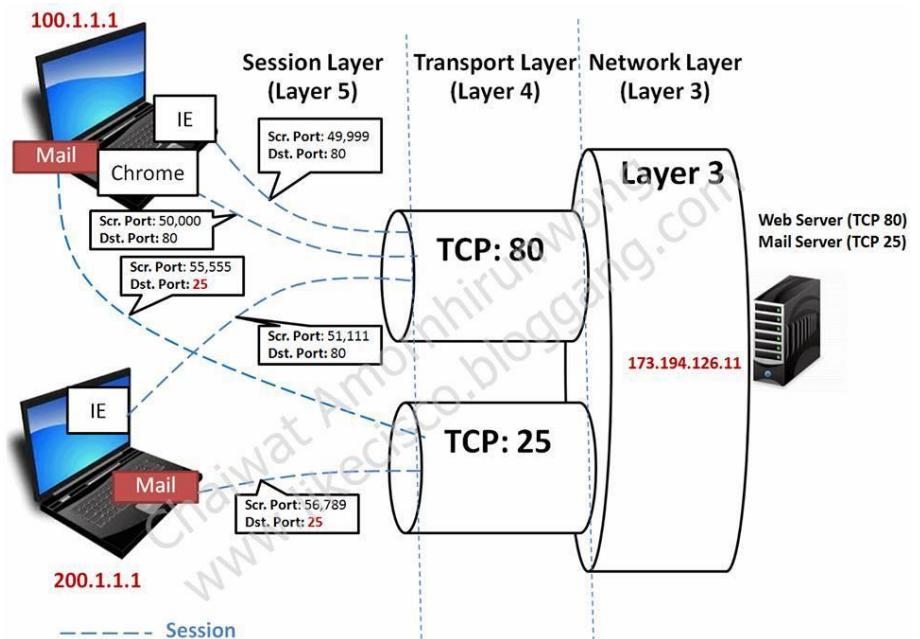
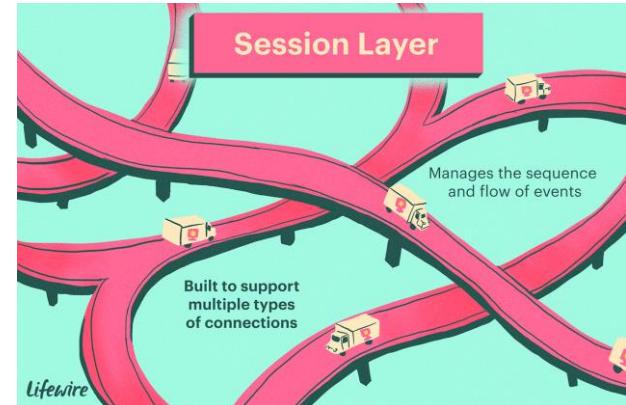
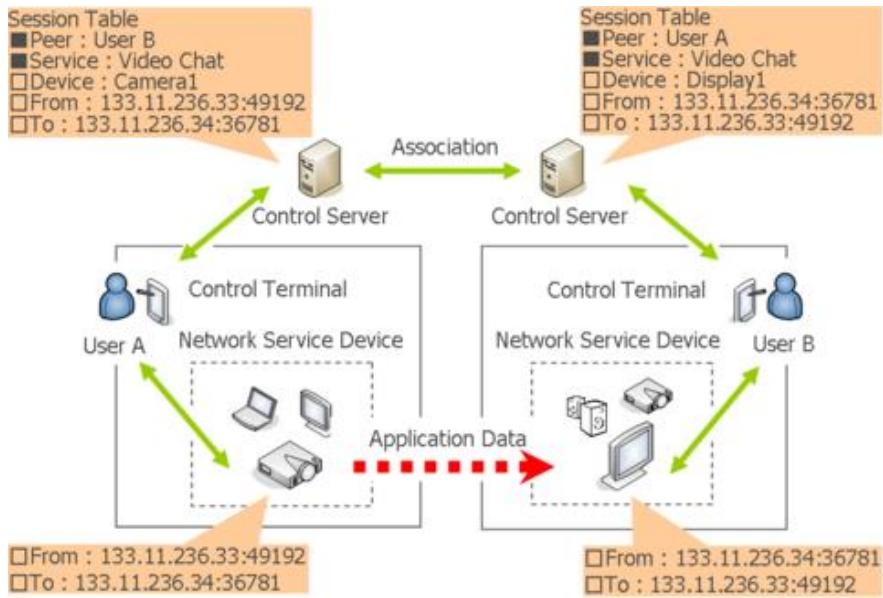
- Andmete läbipaistev transport kahe rakenduse vahel
- Vajadusel garanteerib andmete järjestuse
- Vajadusel garanteerib andmete uuestisaatmise
- Tegeleb otspunktide vahelise vookontrolliga
- Ummistuste lahendamine (congestion control)



TCP	UDP
Secure	Unsecure
Connection-Oriented	Connectionless
Slow	Fast
Guaranteed Transmission	No Guarantee
Used by Critical Applications	Used by Real-Time Applications
Packet Reorder Mechanism	No Reorder Mechanism
Flow Control	No Flow Control
Advanced Error Checking	Basic Error Checking (Checksum)
20 Bytes Header	8 Bytes Header
Acknowledgement Mechanism	No Acknowledgement
Three-Way Handshake	No Handshake Mechanism
DNS, HTTPS, FTP, SMTP etc.	DNS, DHCP, TFTP, SNMP etc.

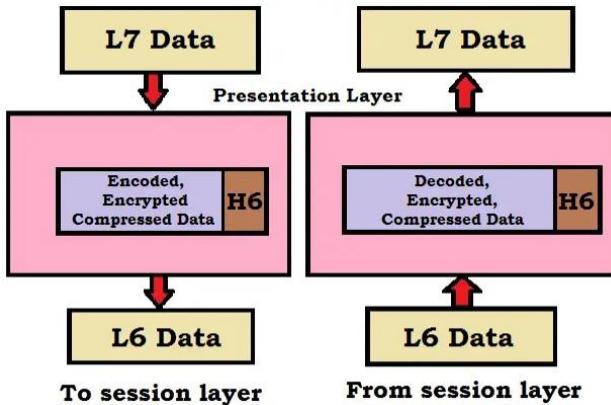
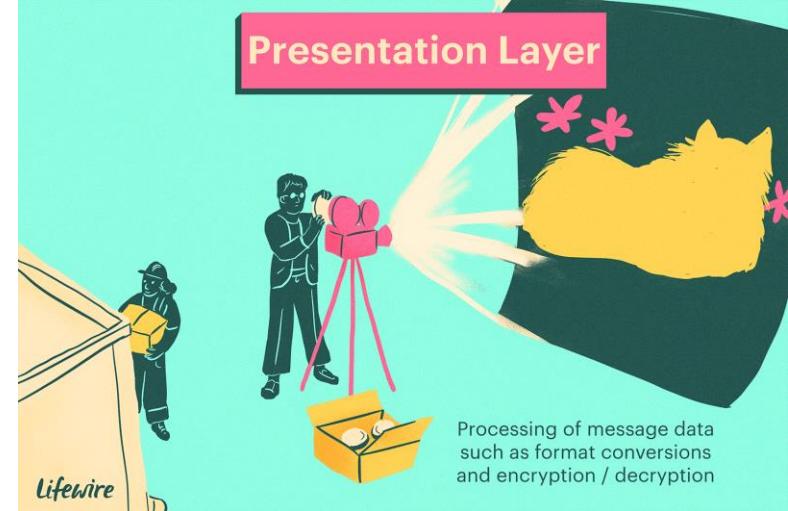
# Seansikiht (Session layer)

- Seansihaldus kahe osapoole vahel:
  - Loob, haldab ja lõpetab loogilisi seansse
- Tegeleb ka seansside jätkamisega vea korral

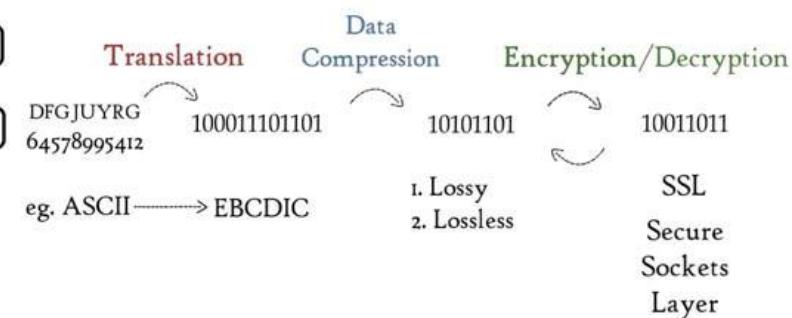


# Esitluskiht (Presentation Layer)

- Andmete esituskujust sõltumatu tõlkekiht
- Tegeleb andmete kodeeringuga, struktuurse esitusega
- Krüpteerimine • Nn. süntaksikiht

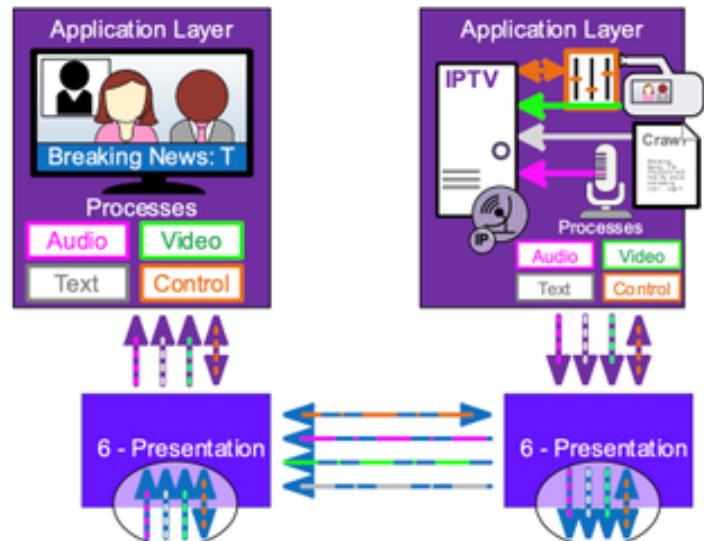
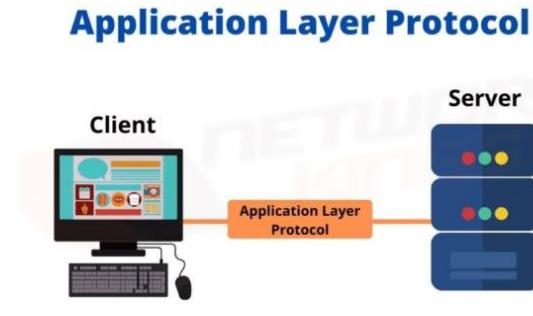
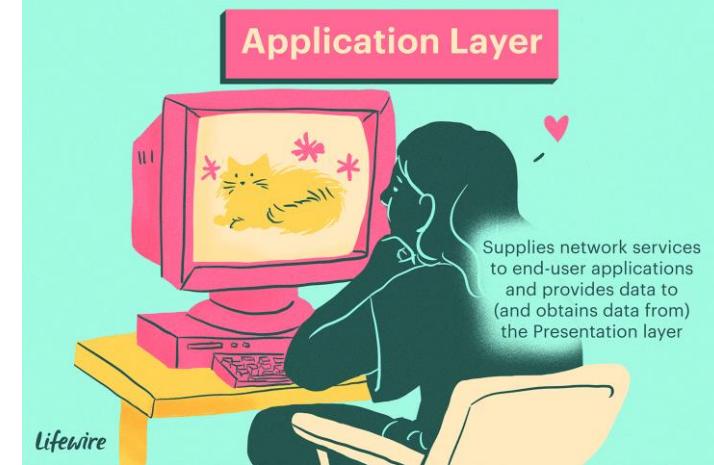


Presentation Layer



# Rakenduskiht (Application layer)

- Rakendusprogrammide spetsiifilised protokollid
- Iga rakendus saab defineerida oma protokolli
- Kõik ülejäävud aspektid on rakenduskihi hallata



# Internet ja OSI mudel

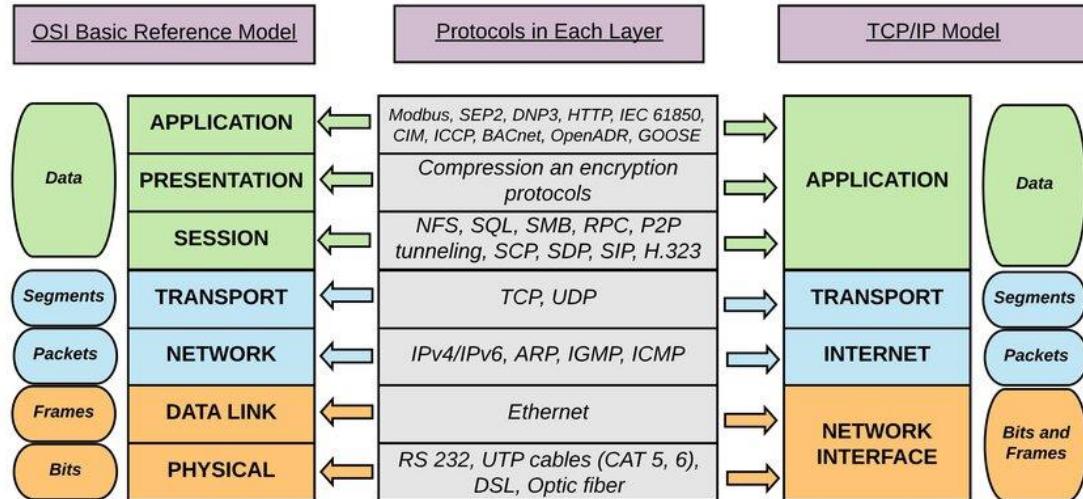
Füüsiline kiht — igasugused, näiteks Ethernet, WiFi

Kanalikiht — MAC aadressidega arvutite adresseerimine, Etherneti 802.3 kaadriformaat, ARP protokoll IP ja MAC vastavuse leidmiseks

Võrgukiht — IP: pakettide marsruutimine õigesse võrku

Transpordikiht: TCP (töökindel baidivoog), UDP (sõltumatute datagrammide saatmine)

Kolm ülemist kihti on kokku sulanud rakenduskihiks • Aegajalt on seansihaldust või esituskihi funktsionaalsust võimalik rakenduse protokolis eristada



# Võrguliidesed

Ühel arvutil võib olla üks või mitu erinevat füüsilist võrguliidest

Lisaks on tavaliselt kasutusel lokaalne arvutisisene võrguliides (loopback)

Enamasti on igal liidesel oma aadress

- Mitmesse L3 vörku ühendatud arvutil peab olema igal liidesel aadress vastavast võrgust

# Linux

```
me@myhost:~$ ip addr
```

...

```
me@myhost:~# ifconfig
```

...

# Windows

```
C:\Users\me> ipconfig
```

...

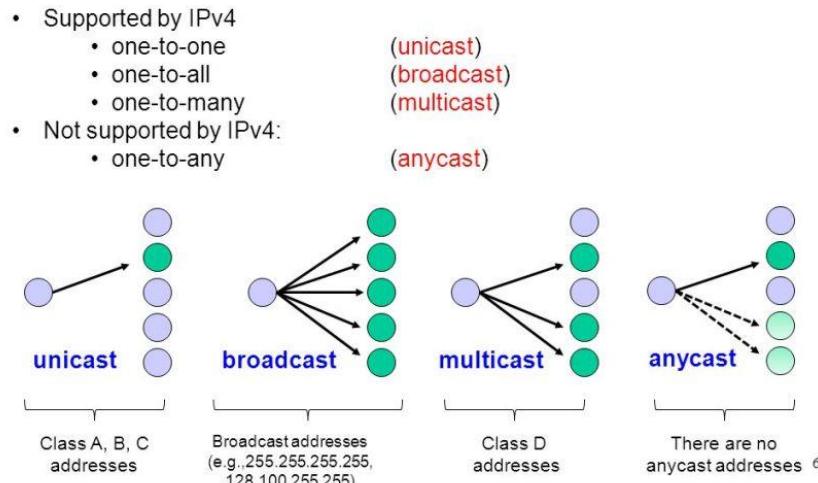
# Mis on rakenduse ja mis OS-i realiseerida?

- Tavalahendus on realiseerida protokollivirna kihid 2-4 opsüsteemis
  - Rakenduste isoleerimine üksteise eest
  - Pordinumbrite hõivamine
  - Soklite jagamine protsesside vahel
- Alternatiiv: transpordikihi (4) kasutajarakendusse toomine
  - Vookontroll ja protokolli parsimine opsüsteemi seest protsessi sisse (end-to-end mudeli laiendamine parema skaaleerimise huvides)
  - Opsüsteemi peab jäama minimaalne demultipleksimine ja jagatud ressursside haldus •
- Rakenduskiht on pea alati kasutaja tasemel, Internet mudelis koos sellega ka seansi- ja esituskiht

# Edastusviisid

- Ainuedastus (unicast) — andmepaketi saatmine üle võrgu ühelt saatjalt ainult ühele vastuvõtjale
- Leviedastus (broadcast) — andmepaketi saatmine üle võrgu ühelt saatjalt kõigile võrgusõlmedele mingis piirkonnas
- Multiedastus (multicast) — andmepaketi saatmine üle võrgu ühelt saatjalt valitud vastuvõtjate rühmale
- Suvaedastus (anycast) — andmepaketi saatmine üle võrgu ühelt saatjalt rühma lähimale vastuvõtjale

## Delivery modes



# Ethernet

Ethernet on tänapäeval protokollide pere, kus on sama kaadri formaat kuid erineva meedia ja kiirusega sidekanalid (kuni 100 Gbit/s seni standarditud)

- Etherneti kaader:
  - Preamble (fikseeritud baitide jada)
    - andmete edastuse alguses sünkroonida saatja ja vastuvõtja kellasid
  - Start frame delimiter
    - Kaadri alguse eristaja
    - selle väärthus on alati "10101011"
    - aitab vastuvõtjal eristada Preamble'it
  - Saaja MAC aadress
  - Saatja MAC aadress
  - Mittekohustuslik VLAN tag
    - (mitme loogilise võrgu tegemiseks)
    - samas kaabilis)
  - Andmeosa pikkus
  - Andmeosa (42-1500 baiti)
  - Kaadri kontrollsumma (32-bitine CRC)

Ethernet Frame

Preamble- 56 bits of alternating 0 and 1  
SFD-Start Frame Delimiter (10101011)

Bytes	Preamble	SFD	Destination MAC	Source MAC	Length or type	Data & Padding	CRC
	7	1	6	6	2	>=46,<=1500	4

Physical Layer Header

# MAC-aadress

- Teise arvuti adresseerimiseks 2. kihi (L2) võrgus peab tema aadressi teadma (näiteks MAC-aadress Etherneti-laadsete protokollide puhul)
- MAC — Medium Access Control
- 48-bitine idee poolest unikaalne seadme identifikaator
- Näiteks 00:0a:e4:7e:a5:e0
- Igal võrguliidesel on aadress tootja poolt sisse programmeeritud
- Koosneb tootja prefiksist ja unikaalsetest baitidest tootja piires
- MAC-aadress ei paista kohtvõrgust (LAN) kaugemale!

# IP

- Internet Protocol
- Palju kohtvõrke on kokku ühendatud, IP-aadresside järgi leitakse tee läbi mitme võrgu õige seadmeni
- IPv4 kasutab 32-bitiseid aadresse
  - $127.0.0.1 = 01111111\ 00000000\ 00000000\ 00000001$
  - $192.0.2.1 = 11000000\ 00000000\ 00000010\ 00000001$
- IP-aadress jaguneb kaheks: võrguosa ja hostiosa
  - Võrguosa bitis on sama L2 võrgu piires kõigil hostidel samad
  - Hostiosa bitid tagavad sama L2 võrgu piires unikaalsuse

# IP võrgumask

- Võrgumask näitabki, missugused bitid antud võrgus fikseeritud on
- Näiteks levinuim  $255.255.255.0 = 3$  bai ti fikseeritud (/24)
- $255.255.254.0 = 11111111\ 11111111\ 11111110\ 00000000 = /23$
- $255.255.255.240 = 11111111\ 11111111\ 11111111\ 11110000 = /28$
- Igas võrgus on eritähendusega aadressid „kõik nullid“ (vanasti kasutusel leviaadressina) ja „kõik ühed“ (leviaadress tänapäeval)

# Võrgumaski lihtne peast arvutamine

- Olgu meil võrgumask 255.255.255.224 kujul, tahame teada bittide arvu ja seda, mitu hosti on antud võrgus (koos leviaadressidega)
- Arvutus on lihtsalt peast tehtav:
  - 255 ja 0 väärused on triviaalselt kõik 1-d või kõik 0-d, need saab baidi kaupa kokku arvutada
  - Neist erinev maski bait  $224 = 256 - 32$
  - Seega on antud võrgus  $32$  eri IP-d –  $32 = 2^5$ , seega on hostiosa pikkus  $5$  bitti
  - Maskis jäab võrguosale seega sellest baidist  $8-5=3$  bitti
  - Seega on numbriliseks maskiks  $/27$  (sest  $8+8+8+3=27$ ).

# ARP

- Kui L2 võrgus tahab arvuti IP-aadressiga Y arvutile IP-aadressiga X paketti saata, on vaja leida sihtarvuti MAC-aadress, et ainult talle saata teisi segamata
- Selleks on IPv4 juures ARP (Address Resolution Protocol)
- ARP päringu puhul kisab klient L2 leviaadressile päringu, et „Kes teist on X? Öelge seda palun Y-le.“
- Kui X enda kohta päringut kuuleb, vastab ta Y-le.
- Y saab nüüd X-le üle L2 IP-paketti saata.
- Lisaks hoiab Y seda kirjet mõne aja meeles oma ARP tabelis.

# IP (mars)ruutimine

- (Mars)ruuterid edastavad liiklust mitme võrgu vahel
- Paketi teekonna igal sammul otsustab selle sammu ruuter, kuhu pakett edasi saata
- Selleks on ruutingutabelid sihtvõrkudega, kust valitakse iga paketi puhul sobivate hulgast kõige pikema maskiga kirje, näiteks:

<b>sihtvõrk</b>	<b>mask</b>	<b>kuhu</b>
192.168.0.0	255.255.255.0	eth0
192.168.1.0	255.255.255.0	10.0.0.2
192.168.0.0	255.255.0.0	10.0.0.5
10.0.0.0	255.255.255.0	eth1
0.0.0.0	0.0.0.0	10.0.0.1

# IP-vahemike jaotamine

- Hierarhiline:
  - ISP saab suure ploki, näiteks /16 kuni /19
  - ISP jagab igale kliendile väiksema ploki, näiteks /24
  - Iga klient võib oma vörku jagada alamvõrkudeks, näiteks /28
- Klient võib osta otse teenusepakkujast sõltumatu IP ploki ja selle maailmale mitme ISP kaudu kättesaadavaks teha
  - Kõrgema käideldavuse jaoks
  - AS (Autonomous System)
  - osapool, kellel on otse oma teenusepakkujast sõltumatu IP-vahemik ja kes korraldab selle ühendamist teiste AS-ide kaudu

# Ruutinguinfo levitamine

- Staatiline ruuting — seadmesse konfigureeritakse ruutingutabel administraatori poolt
- Dünaamiline ruuting — ruuterid vahetavad omavahel infot kättesaadavate võrkude kohta ja arvutavad ise, kust kaudu mingi võrk kõige otsem kättesaadav on
- „Core routers“ — Interneti tuumik, mis koosneb ruuteritest, mis teavad kõigi võrkude asukohti ilma vaikeruutingut kasutamata
- Sisemised ruutinguprotokollid (IGP — interior gateway protocol) — kasutamiseks organisatsiooni sees optimaalse tee leidmiseks, näiteks RIP ja OSPF
- Välised ruutinguprotokollid (EGP — exterior gateway protocol) — kasutamiseks organisatsioonide vahel suuremate plokkide granulaarsusega, näiteks BGP

# IP-aadresside jagamine kohtvõrgus

- Staatiline — igale seadmele eraldatakse IP käsitsi ning konfigureeritakse seade seda kasutama
- Dünaamiline — seade saab käivitamisel omale ajutise aadressi automaatselt
- DHCP (Dynamic Host Configuration Protocol) — protokoll aadressi, domeeninime, staatiliste ruutingute ja muude parameetrite küsimiseks
- DHCP server teab, mis IP-aadress missugusele MAC-aadressile vastavuses on (seos võib olla ajutine või permanentne)

# ICMP

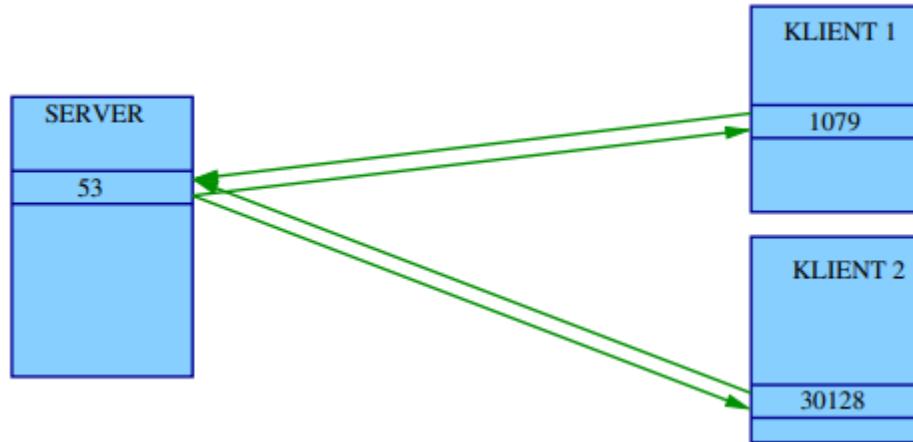
- Internet Control Message Protocol
- Ühelt arvutilt teisele saadetavate juhtsõnumite protokoll
- Näiteks:
  - echo request (ping)
  - echo reply (pong)
  - redirect (teine ruuter on otsem)
  - time exceeded
  - destination unreachable, network unreachable
  - destination unreachable, host unreachable
  - destination unreachable, administratively prohibited
  - destination unreachable, fragmentation needed but DF set (seda ei tohi filtreerida!)

# UDP

- User Datagram Protocol
- Lähtearvuti mingilt protsessilt sihtarvuti mingile protsessile teate saatmine • Lihtne (ainult fragmenteerimine)
- Pole töökindel (iga pakett võib kaotsi minna)
- Olekuvaba (ei mingit korduvsaatmist ega järjestust) • Kiire (ei vaja puhverdamist)
- Sobib ka ühesuunaliseks suhtluseks (leviedastuse ja multiedastuse puhul) • UDP ise ei sisalda ummistuste välimist (congestion control)
- Sobib nii lihtsate päring-vastus tüüpi protokollide kui reaalajavoogude jaoks

# UDP port

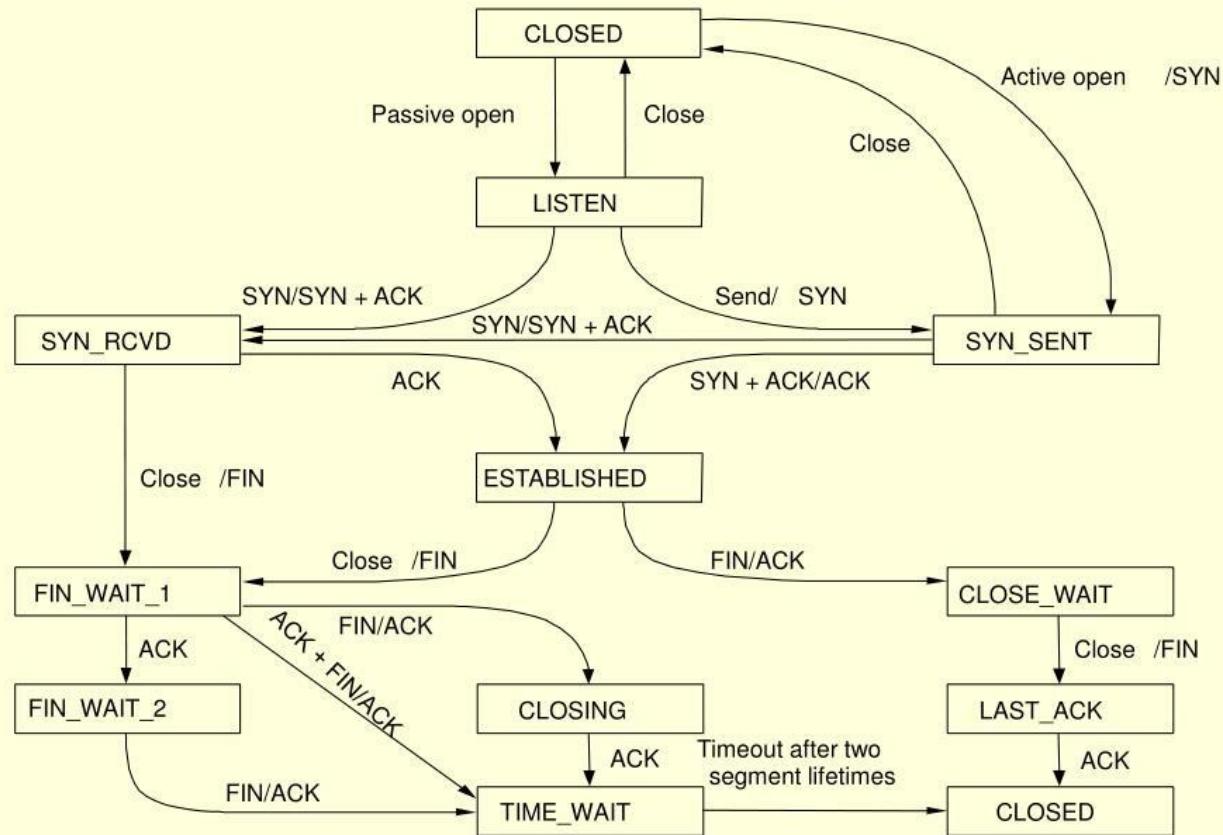
- Eristamaks arvutis mitut protsessi, on side kummaski otspunktis lisaks IP-aadressile kasutusel 16-bitised pordinumbrid
- Enamus UDP-d kasutavates rakendusprotokollides on teenuse pordinumber fikseeritud, kliendi pordinumber dünaamiline ning server vastab sellele kliendi pordile, kust päring tuli



# TCP

- Transmission Control Protocol
- Lähtearvuti mingilt protsessilt sihtarvuti mingile protsessile baidivoo kahesuunaline edastamine
- Pakettide piirid pole fikseeritud, tükeldus võib teel muutuda
- Töökindel (iga andmetükki kviteeritakse, kaotsi minekul saadetakse mingi aja pärast uuesti)
- Järjestatud (andmed jõuavad rakendusele kohale samas järjekorras nagu teisest rakendusest saadeti, TCP puhverdab vajadusel kuni vahepealsed andmed kah kohal on)
- Raskekaalulisem kui UDP — vajab ühenduse loomist andmete saatmiseks, tegeleb ummistuste välimisega
- Tegeleb vookontrolliga (et üks rakendus ei saadaks rohkem kui teine jõuab vastu võtta)

# TCP State-Transition Diagram

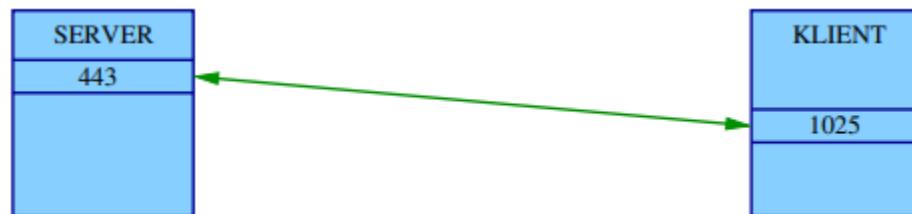


# TCP detailid

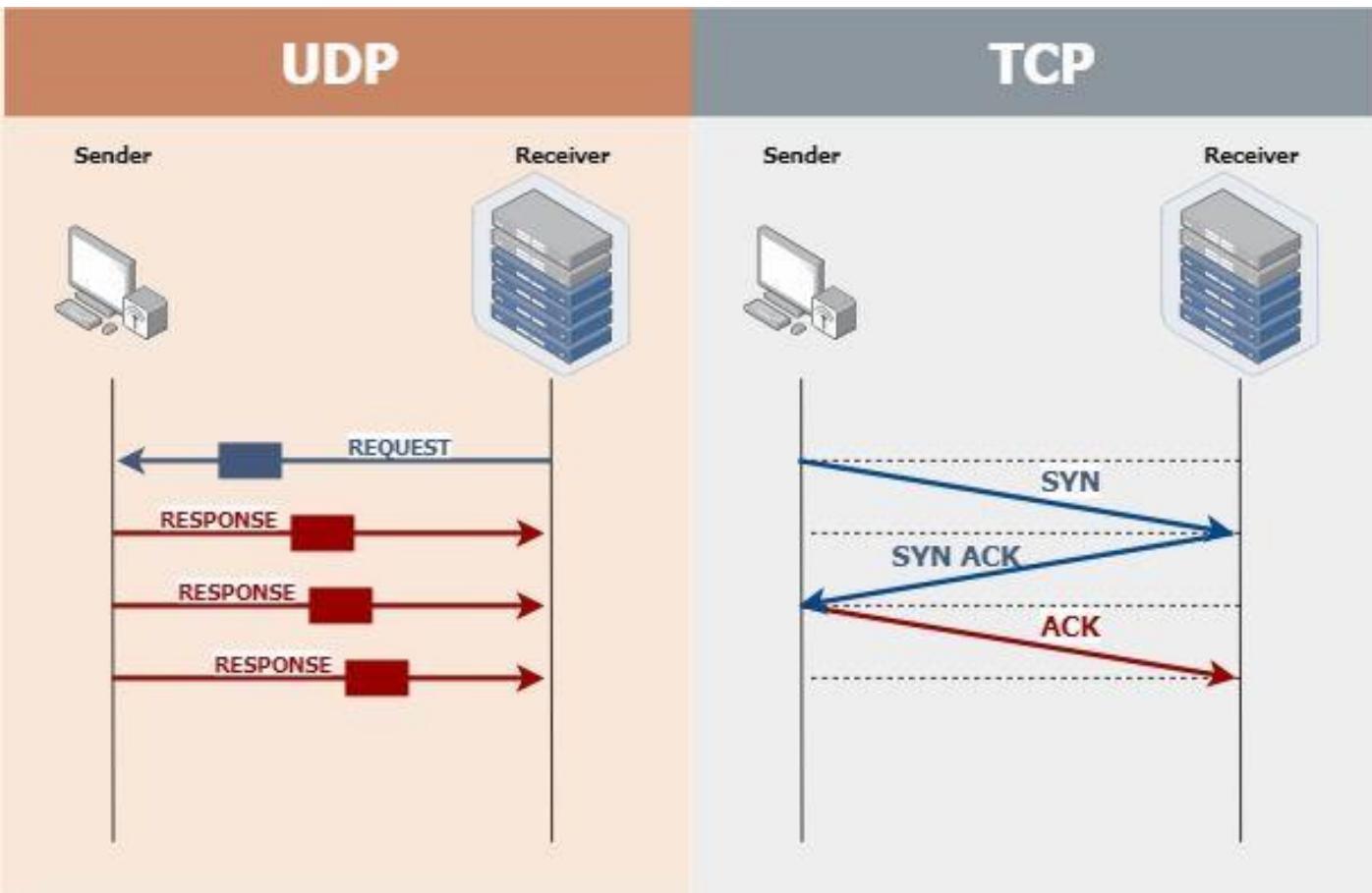
- Selgelt ainult kahele osapoolele (ainuedastus)
- Ühenduse loomine: SYN, ACK, SYN+ACK
- Aknaga protokoll (kui palju andmeid on korraga teel?) vookontrolliks
- Kummaski suunas sõltumatute loenduritega baidivoog
- Slow start — ühenduse või iga järgmise suurema mahu saatmise algul hakatakse kiirust järjest kasvatama, kuni mõõdetakse ära teekonna läbilaskevõime

# TCP pordid

- Side kummaski otspunktis on protsesside eristamiseks lisaks IP-aadressile kasutusel 16-bitised pordinumbrid
- Kahe protsessi vahel võib olla mitu sõltumatut TCP ühendust erinevate kliendiportidega
- Igal teenusel on enamasti fikseeritud serveri pordinumber • Kliendi pordinumber on tavaliselt juhuslik



# TCP vs. UDP sequence diagrams



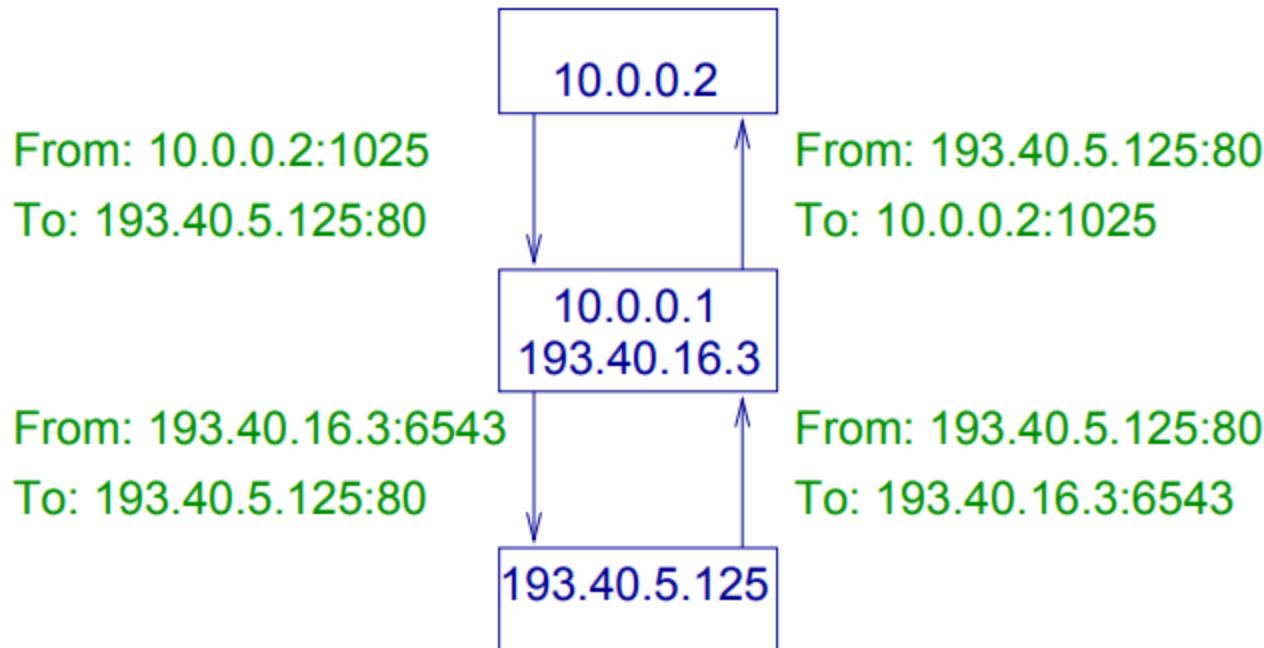
# Võrguaadresside tõlkimine — NAT

- IPv4 aadressidega on kitsas käes, vaja on aadresside kasutamist optimeerida
- Tahame sisevõrgu struktuuri teiste eest ära peita
- Tahame, et sisevõrgu masinad ei oleks väljast otse nähtavad
- Lahendus(?): kasutame sisevõrgus privaataadresse, mis Internetis ei esine •  
Vahel on siiski vaja pakette sise- ja välisvõrgu vahet liigutada
- Lahenduseks on aadresside tõlkimine ruuteris. Tõlkimist on kolme moodi:
  - Staatiline:  $n - n$  — tõlgitakse terve aadressiplokk
  - Dünaamiline:  $n - m$ ,  $m < n$  — avalikke aadresse on vähem
  - Tõlkimine porte kasutades:  $n - 1$  — kõik siseadressid 1 välisaadressiks, varieeritakse lähepordi numbrit

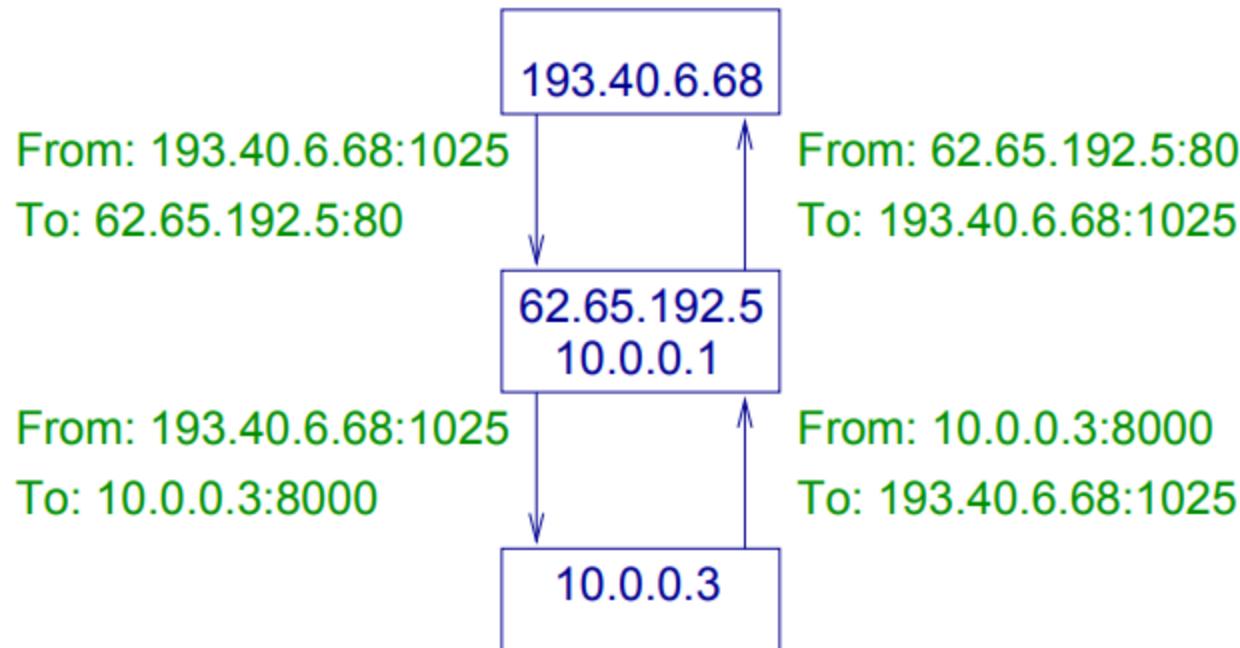
# NAT tehnoloogia

- Standarditega on paika pandud aadressivahemikud, mida võib vabalt oma sisevõrkudes kasutada:
  - 10.\*.\*
  - 172.16.\*.\*
  - 172.31.\*.\*
  - 192.168.\*.\*
- Neid aadresse Internetis ei ruudita
- Aadresse tõlkiv ruuter modifitseerib ühe osapoole IP aadressi (tõlgib ühe suuna andmed ning tõlgib tagasi vastused)
- Lähteaadressi maskeerimise abil saame varjata klientarvutit (algatajat) → SNAT
- Sihtaadressi maskeerimise abil saame varjata serverarvutit → DNAT
- SNAT ja DNAT võib samas ruuteris ka järjest rakendada

# SNAT näide



## DNAT näide



# NAT probleemid

- Teeb katki TCP/IP mudeli, kus ainult ühenduse otspunktid teavad detaile
- Sunnib peale mingi osaliselt fikseeritud marsruudi otspunktide vahel
- Toob sisse ühe katki mineku punkti
- Toob sisse ühildumatuse paljude protokollidega
- Ei lahenda IPv4 aadresside kitsikust
- AGA:
  - Leevendab IPv4 aadresside kitsikust
  - Aitab lihtsalt ja praktiliselt võrku turvalisemaks teha

# IPv6

- Uus võrgukihi protokoll 128-bitiste aadressidega:
  - 2001:bb8:2002:2400:209:3dff:fe11:e8c5/64
  - 2002:5abf:a1ac:2::1/64 – ::1/128
- ARP asemel multiedastusel põhinev neighbour discovery
- Samad TCP, UDP ja enamus rakendustaseme protokolle
- Uued ICMPv6, DHCPv6
- Lisaks DHCPv6-le võimalus olekuvabalt aadresse konfigureerida
- Igale kohtvõrgule /64, ruuter reklamib prefiksit
- Host paneb oma aadressi kokku prefiksist ja 64-bit kohalikust osast (MAC või juhuslik)

# IPv6 üleminek

- Ülemineku ajaks võetakse enamus võrkudes kasutusele nii IPv4 kui IPv6 aadressid
- Protokollid IPv6 tunneldamiseks üle IPv4 (automaatne tunneldamine: 6to4, Teredo, ISATAP)
- API tasemel suudab IPv6 sokkel teenindada ka IPv4 ühendusi
  - IPv4-mapped aadressid ::193.40.36.2 kujul IPv4-aadresside tähistamiseks API-s (aga mitte „traadil“)
  - Vastupidi ei saa
- IPv6 puhul on NAT tugevalt vastunäidustatud — aadresse jätkub, väldime NAT probleeme kui võimalik

# Soklid

- Sokkel (socket) — programmeerimisliides (API) võrguga suhtlemiseks
- Sokkel on sidekanali otspunkt (näiteks TCP ühenduse kummaski otsas on vastaval rakendusel sokkel)
- Sokli-API on protokollist sõltumatu, disainitud OSI mudeli järgi
- Pärit BSD Unixist, tänapäeval üldlevinud (+kohalikud täiendused eri OS-ides)
- Toetab palju protokolliperekondi
- Toetab voosokleid (stream socket) ja paketisokleid (datagram socket)
- Blokeeruvad ja mitteblokeeruvad soklid
- Lisaks nimelahendus (gethostaddr() jt, Interneti puhul kasutab DNS)

# Soklite näited

- Kombineerime protokolliperekonna ja sokli tüübi:
  - PF\_INET, SOCK\_STREAM — TCPv4
  - PF\_INET6, SOCK\_STREAM — TCPv6
  - PF\_INET, SOCK\_DGRAM — UDPv4
  - PF\_UNIX, SOCK\_STREAM — Unixi sisene stream-sokkel protsesside vahel
  - PF\_RAW, SOCK\_DGRAM — Etherneti kaadrid toorkujul

# Soklite API

- socket()
- connect()
- send(), recv()
- sendto(), recvfrom()
- shutdown(), close()
- bind()
- listen()
- accept()
- ioctl()

# Soklinäide: TCP

<b>server</b>	<b>klient</b>
socket()	
bind()	
listen()	
	socket()
	connect()
accept()	
send()	recv()
recv()	send()
...	...
close()	close()

# Soklinäide: tavaline UDP

<b>server</b>	<b>klient</b>
socket()	
bind()	socket()
	bind()
recvfrom()	sendto()
sendto()	recvfrom()
...	...
	close()

# Soklinäide: UDP ja sokli ühendamine

<b>server</b>	<b>klient</b>
socket()	
bind()	
	socket()
	connect()
recvfrom()	send()
sendto()	recv()
...	...
	close()