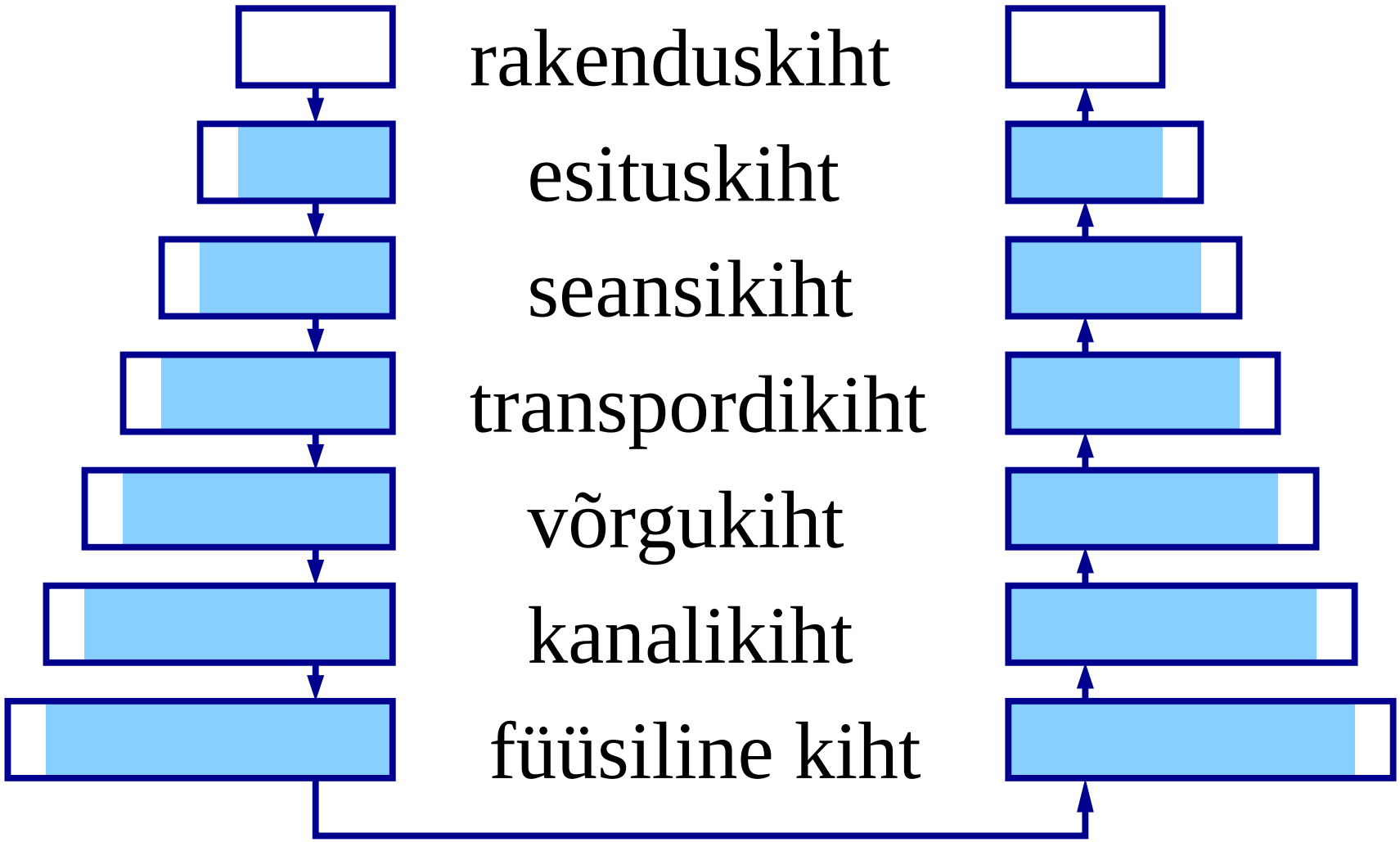
Kiirülevaade arvutivõrgust

- OSI kihiline mudel
- Võrguliidesed
- Ethernet ja MAC aadressid
- IP-aadressid ja ruutimine
- TCP, UDP ja pordid
- NAT
- IPv6
- Ühendused, soklid

ISO/OSI 7-kihiline mudel



Füüsiline kiht (1)

- Riistvara bitivoo ülekandmiseks
  - elektriimpulsid
  - valgus
  - raadiosignaalid
- Signaalinivood, sagedused
- Kaablid, pistikud

Kanalikiht (2)

- Andmepakettide kapseldamine bitivoo sisse
- Kaadrite sünkroniseerimine
- Vookontroll
- Veaparandus
- Kaadrite märgistamine tüüpidega

Võrgukiht (3)

- Tee otsimine võrgus rohkem kui kahe seadme vahel
- Kommuteerimine ja marsruudi leidmine
  - Igale paketile eraldi
  - Ühe korra virtuaalse kanali loomisel
- Võrgusõlmede adresseerimine
- Pakettide edastamine erinevate võrkude vahel
- Ummistuste lahendamine (*congestion control*)

Transpordikiht (4)

- Andmete läbipaistev transport kahe rakenduse vahel
- Vajadusel garanteerib andmete järjestuse
- Vajadusel garanteerib andmete uuestisaatmise
- Tegeleb otspunktide vahelise vookontrolliga

Seansikiht (5)

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

Esituskiht (6)

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

Rakenduskiht (7)

- Rakendusprogrammide spetiifilised protokollid
- Iga rakendus saab defineerida oma protokolli
- Kõik ülejäänud aspketid 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 ruutimine 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 protokollis 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 võrku ühendatud arvutil peab olema igal liidesel aadress vastavast võrgust

Mis on rakenduse ja mis OS-i realiseerida?

- Tavalahendus on realiseerida protokollivirna kihid 2-3 opsüsteemis
  - Rakenduste isoleerimine üksteise eest
  - Pordinumbrite hõivamine
  - Soklite jagamine protsesside vahel
- Alternatiiv: transpordikihi kasutajarakendusse toomine
  - Vookontroll ja protokolli parsimine opsüsteemi seest protsessi sisse (*end-to-end* mudeli laiendamine parema skaleerimise huvides)
  - Opsüsteemi peab jääma minimaalne demultipleksimine ja jagatud ressursside haldus
- Rakenduskiht on pea alati kasutaja tasemel, Internetis 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

Ethernet

- Ethernet on tänapäeval protokollide pere, kus on sama kaadri formaat kuid erineva meedia ja kiirusega sidekanalid (kuni 40 Gbit/s seni standarditud)
- Lisaks alguse ja lõpu marketitele sisaldab üks Etherneti kaader:
  - Sihtseadme MAC aadress
  - Saatja MAC aadress
  - Mittekohustuslik VLAN tag (mitme loogilise võrgu tegemiseks samas kaablis)
  - Andmeosa pikkus
  - Andmeosa (42-1500 baiti)
  - Kaadri kontrollsumma (32-bitine CRC)

MAC-aadress

- Teise arvutit 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 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 baiti 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 väärtused 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 jääb 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:

sihtvõrk	**mask**	**kuhu**
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
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 võrku jagada alamvõrkudeks, näiteks /28 sealtsamast kust ISP-dki ja kasutada seda mitme ISP-ga suhtlemiseks
- 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

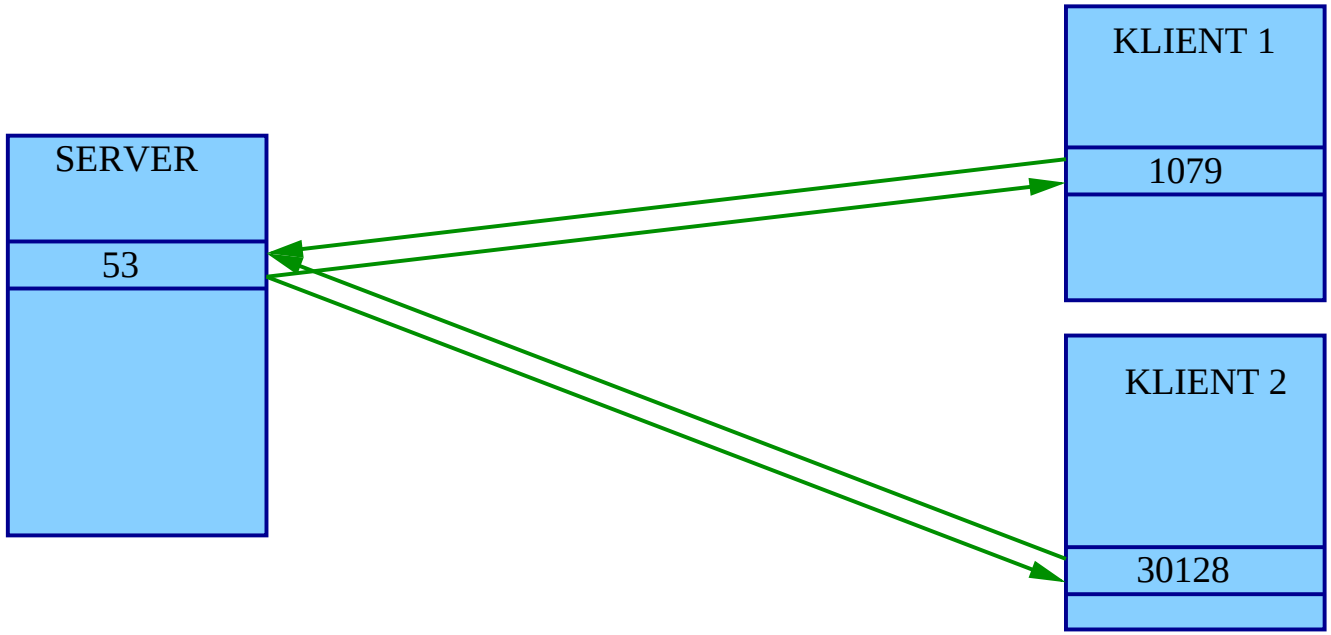
- *Internet Control Message Protocol*
- Ühelt arvutilt teisele saadetavate juhtsõnumite protokoll
- Näiteks:
  - *echo request* (ping)
  - *echo reply* (pong)
  - *redirect* (teine ruuter on otsem)
  - *time exceeded*
  - *destination unreachable, network unreachable*
  - *destination unreachable, host 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 taassaatmist ega järjestust)
- Kiire (ei vaja puhverdamist)
- Sobib ka ühesuunaliseks suhtluseks (leviedastuse ja multiedastuse puhul)
- UDP ise ei sisalda ummistuste vältimist (*congestion control*)
- Sobib nii lihtsate päring-vastus tüüpi protkollide 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

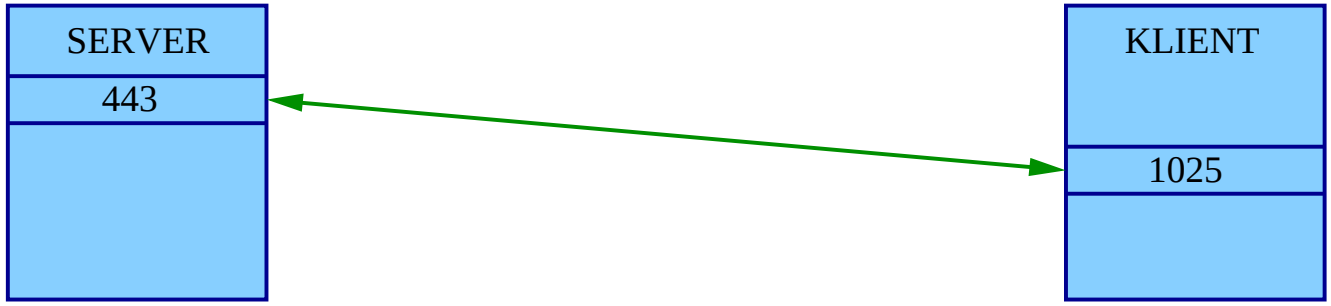
- *Transmission Control Protocol*
- Lähtearvuti mingilt protsessilt sihtarvuti mingile protsessile baidivoo kahesuunaline edastamine
- Pakettide piirid pole fikseeritud, tükeldus võib teel muutuda
- Töökindel (iga andmetükki kviteeritakse, kaotsiminekul saadetakse mingi aja pärast uuesti)
- Järjestatud (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 vältimisega
- Tegeleb vookontrolliga (et üks rakendus ei saadaks rohkem kui teine jõuab vastu võtta)

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 oma loenduritega
- *Slow start* — ühenduse 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

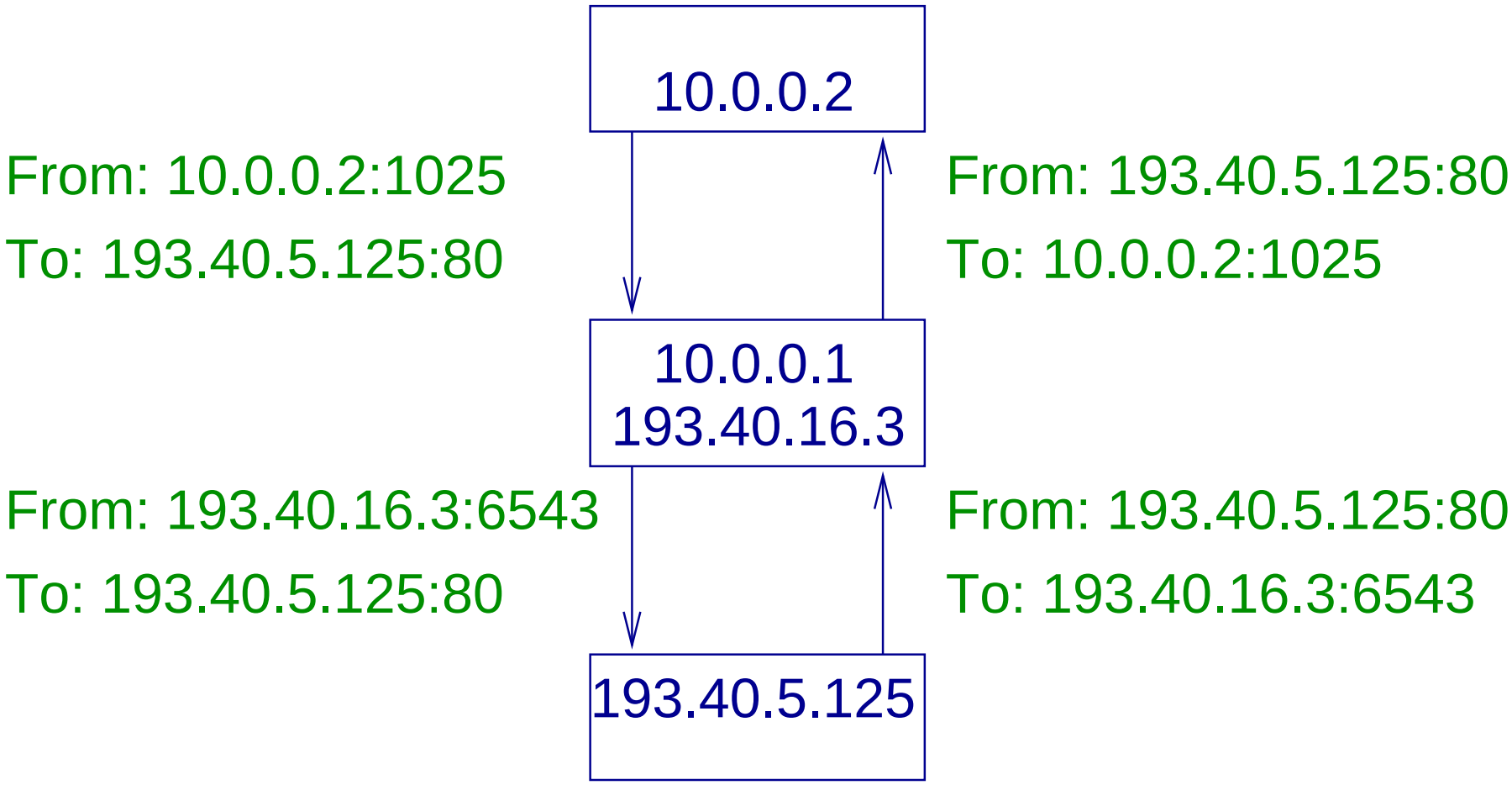
Võrguaadresside tõlkimine — NAT

- 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 siseaadressid 1 välisaadressiks, varieeritakse lähtepordi numbrit

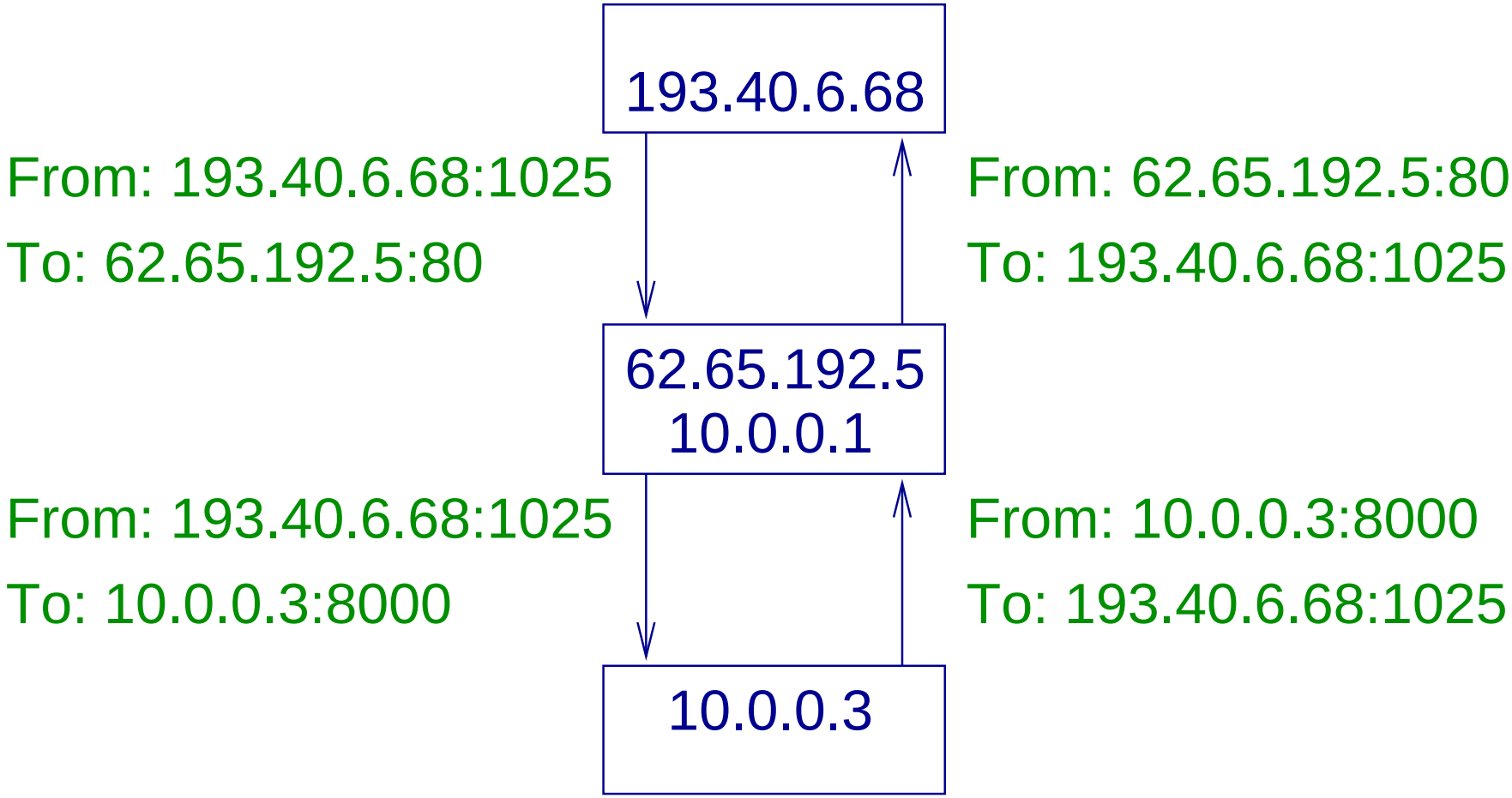
NAT tehnoloogia

- 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) $\rightarrow$ SNAT
- Sihtaadressi maskeerimise abil saame varjata serverarvutit $\rightarrow$ DNAT
- SNAT ja DNAT võib samas ruuteris ka järjest rakendada

SNAT näide


10.0.0.2
From: 10.0.0.2:1025
To: 193.40.5.125:80
From: 193.40.5.125:80
To: 10.0.0.2:1025
10.0.0.1
193.40.16.3
From: 193.40.16.3:6543
To: 193.40.5.125:80
From: 193.40.5.125:80
To: 193.40.16.3:6543
193.40.5.125

DNAT näide


193.40.6.68
From: 193.40.6.68:1025
To: 62.65.192.5:80
From: 62.65.192.5:80
To: 193.40.6.68:1025
62.65.192.5
10.0.0.1
From: 193.40.6.68:1025
To: 10.0.0.3:8000
From: 10.0.0.3:8000
To: 193.40.6.68:1025
10.0.0.3

NAT probleemid

- Teeb katki TCP/IP mudeli, kus ainult ühenduse otspunktid teavad detaile
- Sunnib peale mingi osaliselt fikseeritud marsruudi otspunktide vahel
- Toob sisse ühe katkimineku 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 töötav *neighbour discovery*
- Samad TCP, UDP ja enamus rakendustaseme protokolle
- Uued ICMPv6, DHCPv6
- Lisaks DHCPv6-le võimalus olekuvabalt aadresse konfigureerida
  - Igale kohtvõrgule /64, ruuter reklaamib 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 soocket*) 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

server	klient
socket()	
bind()	
listen()	
	socket()
	connect()
accept()	
send()	recv()
recv()	send()
...	...
close()	close()

Soklinäide: tavaline UDP

server	klient
socket()	
bind()	
	socket()
	bind()
recvfrom()	sendto()
sendto()	recvfrom()
...	...
	close()

Soklinäide: UDP ja sokli ühendamine

server	klient
socket()	
bind()	
	socket()
	connect()
recvfrom()	send()
sendto()	recv()
...	...
	close()