ARVUTIVÕRGUD - KONSPEKT 2

1. MITMEKIHILINE ARHITEKTUUR 2
2. OSI MUDEL 2
3. TCP/IP MUDEL 3
4. AHELKOMMUTATSIOON. PAKETTKOMMUTATSIOON. SÕNUMIKOMMUTATSIOON 4
5. MULTIPLEKSIMINE 4
6. DATAGRAMM VÕRGUD, VIRTUAALAHELATEGA VÕRGUD 4
7. EDASTUSMEEDIA 4
8. AJALISED VIITED VÕRKUDES 5
9. MIDA ERINEVAD RAKENDUSED NÕUAVAD VÕRKUDELT 5
10. HTTP 5
11.FTP 6
12. ELEKTRONPOST. SMTP 6
13. DNS 7
14. USALDATAV ANDMEEDASTUS 7
15. GO-BACK-N, SELECTIVE-REPEAT 8
16. TCP 8
17. TCP VOO JUHTIMINE 9
18. TCP KOORMUSE JUHTIMINE 9
19. UDP 10
20. MARSRUUTIMINE 10
21. IPV4 JA IPV6 10
26. DATAGRAMMIDE EDASTUS LÄBI VÕRKUDE 11
27. VIGADE AVASTAMINE JA PARANDAMINE 11
28. LOKAALVÕRGUD. TOPOLOOGIAD 12

Arvutivõrgud - konspekt

1. Mitmekihiline arhitektuur

Rakenduskiht -> Transpordikiht -> Võrgukiht -> Transpordikiht -> Rakenduskiht.
Võimaldab lahutada arvutivõrgu ja riistvara konkreetsest rakendusest. Kõik komponendid on iseseisvad, neid saab sõltumatult asendada. Üks komponent (kiht) ei pea teadma, kuidas teine täpselt töötab. Olulised on ühe kihi poolt teisele pakutavad teenused. Alumine kiht pakub teenust ülemisele kihile (nt. transpordikiht rakenduskihile). Kõige madalam kiht on võrgukiht.

Andmevahetus kahe osapoole vahel:
Allikas - andmete genereerimine
Saatja – teisendab andmed transportimiseks sobivale kujule
Edastussüsteem – transpordib signaali ühest kohast teise
Vastuvõtja – võtab signaali ja teisendab arusaadavale kujule (ADM – analoog-digitaal muundur)
Adressaat – kasutab saadud andmeid

Saatja ja vastuvõtja peavad suhtlema samas keeles.
Protokoll – reeglistik, mida järgides on kaks osapoolt võimelised suhtlema. Koosneb süntaksist, semantikast ja ajastusest (kiiruste omavaheline kokkusobivus, time-outid jne.)

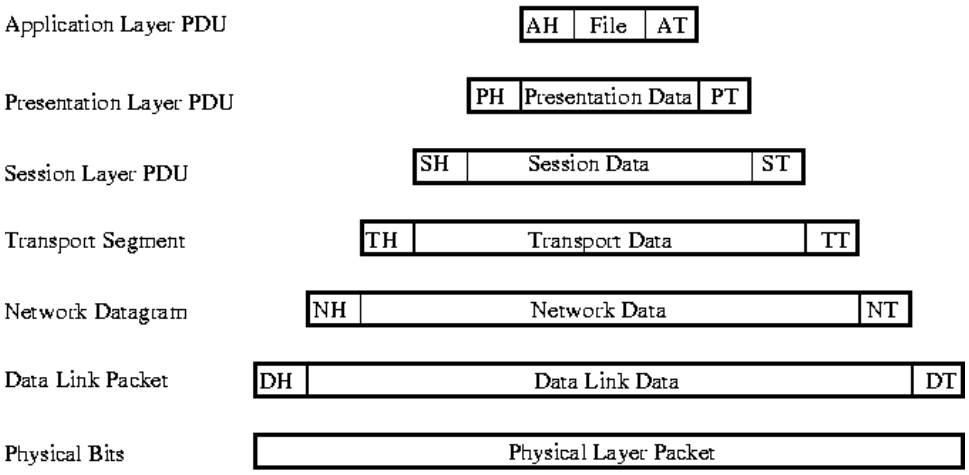
Saatja ja vastuvõtja samad kihid suhtlevad omavahel tinglikult (s.t. kasutades alumise kihi poolt temale osutatavaid teenuseid) ja eelnevalt kokku lepitud protokolli. Teenuseid osutatakse läbi liideste, s.t. läbi kindlaksmääratud funktsioonide.

Iga kiht lisab saadud andmetele juurde kindla päise ja edastab tulemuse temast madalamal olevale kihile. Vastuvõtmisel võtab iga kiht temale määratud päise maha.

PDU – protocol data unit. Protokolli andmeüksus. Andmete hulk, mida üks kiht saadab teisele. Transpordikihi PDU sisaldab sihtaadressi, järjekorranumbrit ja veaparanduskoode. Transpordikiht annab oma PDU üle võrgukihile. Võrgukihis lisatakse arvuti aadress prioriteet. Toimub tegelik edastus.

SAP – service access point – rakenduskihi päis.
DSAP – destination service access point – transportkihi päis. Sisaldab siht-, rakenduse- ja pääsuaadressi.
DHOST – võrgukihi päis. Sisaldab sihtarvuti aadressi.

2. OSI mudel



7 kihti:
Rakenduskiht (application l.) – Võrguteenuste lõppkasutajale mugaval kujul esitlemine.

Esitluskiht (presentation l.) – Võrgust saabuvate andmete teisendamine üldkujult konkreetse rakenduse jaoks sobivale kujule ja vastupidi. Samuti tegeletakse siin failide pääsuõiguste ja lukustamise (s.t. kui kasutaja töötab konkreetse failiga) kontrollimisega.

Seansikiht (session l.) – Loob ühenduse tööjaamas töötava rakenduse ja võrgu vahel. Siin tehakse vahet juhtkäskudel ja andmetel. Toimub ühenduse loomine ja sulgemine, samuti autentimine. Määratakse, millisel kujul toimub info saatmine (krüpteerimine ?).

Transpordikiht (transport l.) – Realiseeritud lõppjaamades. Tegeleb lõppjaamade vahelise andmesidega. Siin toimub usaldusväärse andmeedastuse garanteerimine. Siin muudetakse rakenduselt saadud andmed segmentideks. Võrgu ülekandeks sobivateks segmentideks ja määratakse ning kontrollitakse nende järjekorda. Samuti määratakse ära, kas edastamisel kasutatakse TCP või UDP protokolli. Selles kihis luuakse ühendus masinate vahel. Siit allapoole võib ühendust lugeda punkt-punkt ühenduseks.

Võrgukiht (network l.) – Tegutsetakse IP aadresside tasemel. Andmeühikuks on datagramm. Kasutab võrguliidesena IP protokolli. Tegeleb marsruutimise ja erinevate võrkude vahelise andmeedastuse ning voo juhtimisega. Samuti tükeldatakse ja defragmenditakse ka suuremaid datagramme. Igal seadmel on 32-bitine IP-aadress. IP-pakette adresseeritakse IP-aadressi kaudu, kuid tegelikus edastuses kasutatakse MAC-i. IP aadress seotakse MAC-iga ARP protokolli abil.

Kanalikiht (data link l.) – Jagab datagrammid pakettideks. Muudab saabunud paketid datagrammideks. Töötab bititasemel ja lisab algus-lõpu lipukesi ja veakontrolli. Veakontroll on bititasemel. Vigaste pakettide korral nõutakse nende uuestisaatmist. Juhib füüsilist ja loogilist ühendust paketi sihtpunktiga, kasutades võrguliidest. Igale võrguseadmele on eraldatud unikaalne 48-bitine ainult antud seadmega seotud MAC (media access control) aadress. Kui kõik 48-bitti on 1-d, saavad paketi kätte kõik võrgus olevad seadmed. Siin toimub ka sissetuleva paketi MAC-aadressi kontroll (kas on pakett on mõeldud antud seadmele või mitte).

Füüsiline kiht (physical l.) – Tegeleb bittide ülekandmisega. Juhib võrgu riistvara liideste tööd, s.h. kaabli tüüp (coax, twisted pair). Võrgu töösagedus, pinged, topograafia. (nt. 10BaseT, 10Base5, ArcNet)

3. TCP/IP mudel

Kirjeldatakse 3-5 tasemest koosneva mudelina, sõltuvalt implementatsioonist.

Rakenduskiht (application l.) – Sisaldab OSI rakendus-, esitlus- ja seansikihti. Rakendusena käsitletakse iga protsessi, mis toimub transpordikihist kõrgemal, sisaldades kõiki kasutajaga seotud toiminguid. Siin kontrollitakse andmete esitluskuju ja seansi juhtimist.
Rakendused kasutavad üle võrgu suhtlemiseks erinevaid protokolle, mis suhtlevad omavahel portide kui unikaalsete identifikaatorite kaudu. (POP, SMTP, FTP, HTTP).

Transpordikiht (transport l.) – Juhib programmide omavahelist suhtlemist võrgus, kasutades TCP või UDP protokolli.

Võrgukiht (internet l.) – Võimaldab andmeedastust masinate vahel, mis asuvad erinevates alamvõrkudes. Antud kihi teenuseid kasutavad lisaks lõppjaamadele ka marsruuterid. Toimub adresseerimine erinevate võrkude vahel. Kasutatakse IP ja ICMP protokolle.

Võrgupöörduskiht (link l.) – Seob endas OSI kanalikihi ja osaliselt ka füüsilise kihi. Toimub füüsiline adresseerimine ja füüsiliste parameetrite määramine.

Füüsiline kiht (physical l.) – Sellel tasemel toimub füüsiline andmeedastus.

4. Ahelkommutatsioon. Pakettkommutatsioon. Sõnumikommutatsioon

Ahelkommutatsiooni korral reserveeritakse kogu kanali ressurss ühenduse ajaks. Ühendus-orienteeritud. Vajalik on eelnev ühenduse loomine. Siin on tagatud kindel andmeedastuskiirus (oluline AV ja muu reaalajas edastatava info puhul). Suure kanali korral saab kasutada aja (erinevatel ajahetkedel kasutavad kanalit erinevad kliendid) või sageduse (erinevatel sagedustel saadetakse erinevat infot) järgi tihendamist.

Pakettkommutatsiooni puhul kasutatakse jagatud ressurssi. Iga pakett võib liikuda erinevat marsruuti pidi, mille tulemusena võib võrgusõlmedes esineda viivitusi. Efektiivsem, kui on lubatud teatav hilistumine, samuti paiskandmeedastuse korral. Pakettkommunikatsioon ei ole ühendus-orienteeritud, seda on võimalik muuta, kasutades kõrgemate kihtide protokolle (nt. TCP, mis muudab IP-võrgud ühendus-orienteerituks).

ATM seob kaks eelnevat, kasutades oma võrkudes nii kindlat andmeedastuskiirust kui ka jagatud ressurssi.

Sõnumiedastuse korral saadetakse edasi kõik ühe sõnumi paketid korraga. Võrgusõlmed peavad enne edastamist kõik sõnumi paketid kätte saama, seega võib viide olla suurem.

5. Multipleksimine

Ühes kanalis oleks mõistlik saata korraga mitmeid erinevaid pakette.

FDM (frequency division multiplexing) – Erinevad võrguseadmed kasutavad suhtlemiseks sidekanali erinevaid sagedusi.
TDM (time division multiplexing) – Igal seadmel on õigus oma infot edastada mingil kindlal ajahetkel. Vajalik on täpne sünkroniseerimine.

TCP protokolli korral realiseeritakse multipleksimine erinevate portide kasutuselevõtuga.

6. Datagramm võrgud, virtuaalahelatega võrgud

Datagramm-võrkudes toimub marsruutimine sihtpunkti aadressi järgi. Iga paketi puhul otsustatakse eraldi, milline marsruut oleks kõige õigem valida.
Virtuaalahelatega võrgud – Enne andmete saatmist pannakse marsruut paika. Luuakse virtuaalne ahel, mille kaudu saates ei pea igale paketile eraldi marsruuti otsima. Paketid on sel juhul alati õiges järjekorras. Ahelate loomiseks kasutatakse identifikaatorit, mis ei ole unikaalsed globaalses mõttes, vaid igas ruuteris hoitakse vastavuste tabelit, mille järgi saab teada, kuhu antud identifikaatoriga pakett on vaja edasi saata.

7. Edastusmeedia

Eristatakse juhitavaid keskkondi ja vabu keskkondi. Vabades keskkondades signaale ei juhita, need kulgevad vabalt.

Juhitava keskkonna edastusmeediad:

TP	CAT5	100 Mbps	
	CAT3	10 Mbps	
CX		10 Mbps	
Fiiber	Ethernet	100 Mbps	
	Point-to-point	5 Gbps	

Vaba keskkonna edastusmeediad:

Mikrolained	45 Mbps	
WLAN	2 Mbps	
	11 Mbps	
SAT	50 Mbps	270 msek viide
WAN (mobiilside)		

8. Ajalised viited võrkudes

Seotud andmete töötlemise ja järjekordadega; saatmisega liini ja liikumisega mööda seda.

Töötlemise viide: iga pakett võetakse vastu, päise järgi analüüsitakse, kuhu see edasi saata – selleks kulub aega.

Järjekordade viide: vaja oodata, kuni protsessor vabaneb paketi töötlemiseks, samuti on määrav võrgu koormus (kui kiiresti saab paketti edasi saata).

Edastusviide: aeg, mis kulub paketi liinile toimetamiseks.

Meediumi viide: aeg, mis kulub paketi liikumiseks mööda sidekanalit.

$t = R/l$	t – aeg, mis kulub bittide saatmiseks liini, R – ribalaius, l – liini pikkus
$i = l*a/R$	i – liikluse intensiivsus, a – keskmine pakettide saabumise aeg

Igas võrguseadmes on puhver (stack), kuhu salvestatakse kõik töötlemist ootavad paketid. Kui puhver on täis, hakatakse sissetulevaid pakette ignoreerima, s.t. $i < 1$.

9. Mida erinevad rakendused nõuavad võrkudelt

Kui kaks rakendust asuvad ühes arvutis, kasutatakse omavaheliseks suhtlemiseks operatsioonisüsteemi. Kui andmevahetus toimub üle võrgu, siis vajatakse rakenduskihi protokolle.
Rakendused nõuavad kahetasemelist adresseerimist: IP-aadressi ja pordi kaudu.

Rakenduse jaoks võrku iseloomustavad parameetrid:
Andmete kadu – sõltuvalt rakendusest võib andmete kadu olla suurem või väiksem, häirimata seejuures rakenduse tööd. Mõni rakendus on andmete kao suhtes tolerantsem kui teine. (nt. live video vs. FTP)

Ajalised viited – mõne rakenduse puhul pole viide nii määrav (n.t. e-mail). Reaalajarakendustes see nii ei ole (AV-ülekanne).

Edastuskiirus – /mõtle ise edasi!/

Vastavalt rakenduse vajadustele kasutatakse erinevaid protokolle. TCP on veakindel, paketid pannakse alati õigesse järjekorda (see võtab aega). UDP-s ei ole veakontrolli, samuti ei garanteerita pakettide kohalejõudmist ega nende õiget järjekorda. Oluline on ühenduse hoidmine, mitte see, kas andmed lähevad kaduma või mitte (nt. real audio).

10. HTTP

Hyper Text Transfer Protocol
Veebiserveri ja brauseri omavahelise suhtlemise protokoll. Kasutab alusena TCP-d. Olekuta protokoll, s.t. veebiserver ei mäleta kliendi eelmisi päringuid.
HTTP 1.0 korral algatatakse iga päringu jaoks uus TCP-ühendus, HTTP 1.1 korral võib ühe ühenduse raames teostada mitu päringut. Ühenduse kestvus piiratakse ajalimiidiga.

Esineb kolme tüüpi päringuid:
GET – küsib infot;
POST – klient saadab veebiserverile infot
HEAD – päring, millele ei nõuta serveri-poolset vastust.

Kuna veebiserver ei mäleta eelmisi päringuid, peab näiteks alati autentimist nõudva lehe puhul iga päringu algusesse lisama „authorization:"-rea. Kui seda rida ei ole, siis nõutakse kasutajanime ja parooli uuestisisestamist.

HTTP olekuta olemust püütakse korvata küpsiste abil. Küpsistesse salvestatakse info, mida järgnevatel päringutel vaja võib minna. Küpsiseid eristab nende identifikaator, mis on serveri poolt genereeritud ja salvestatud. Klient peab iga päringu alguses selle identifikaatori serverile edastama.

Kiiruse suurendamiseks (andmemahtude vähendamiseks) kasutatakse nn. tingimuslikku GET-i. Sel juhul ei saadeta objekti brauserile, kui viimasel on juba olemas piisavalt värske koopia sellest.

Vahemälu kasutamine. Kõik külastatud leheküljed salvestatakse vahemällu (cache), et nende hilisemal vaatamisel oleksid leheküljed kättesaadavad kohalikust arvutist.

Proxy serveri kasutamisel tõmmatakse kõik leheküljed proxy serverist. Kui proxys lehekülge ei ole, tõmbab proxy server selle ise originaalasukohast, et hiljem saaksid kasutajad selle juba kohaliku võrgu proxyst.

Cache ja proxy vähendavad ajakulu ja võrgu koormust.

HTTP päringu vastuses sisaldub vastuse kood ja tekst (nt. 404 – Page not found). Samuti on ära näidatud serveri tüüp, viimane muutmise kuupäev, paketi pikkus ja andmete tüüp.

11.FTP

File Transfer Protocol, transpordikiht, port nr. 21
Kasutatakse failide transportimiseks.
Juhtkäskude ja andmete vahetamiseks kasutatakse tavaliselt erinevaid porte. FTP on olekut säilitav protokoll, kasutajainfo ja aktiivse kataloogi info säilitatakse. Seega ei ole vaja iga päringu algul edastada kasutajanime ja parooli, samuti pole vaja öelda oma asukohta kataloogipuus. Vastustena FTP päringutele saadetakse vastuse kood ja selle tähendus (n.t. 331 Username OK).

12. Elektronpost. SMTP

Simple Mail Transfer Protocol, transporikiht, port nr. 25
Meilisaatmiseks on vajalikud kolm komponenti: meiliserver, meiliklient ja neid siduv SMTP protokoll. Meiliklienti kasutatakse kirjade saatmiseks ja lugemiseks (kopeerides need eelnevalt meiliserverist).
Meiliserveris hoitakse kõiki sissetulnud kirju, seal asuvad kasutajate postkastid ja saatmisel olevad kirjad (ühtne järjekord, sõltumata kasutajast).

Enne saatmist luuakse TCP-ühendus kahe meiliserveri vahel. Kasutatakse 7-bitist ASCII kodeeringut.
Saatmise kolm faasi: Ühenduse loomine, teadete saatmine, ühenduse lõpetamine.
Teate saatmisel ei ole kirja sees lubatud mõned märgikombinatsioonid CR/LF.CR/LF, mis tähendab kirja lõppu.

SMTP on push-protokoll, s.t. toimub andmete saatmine kliendi poolt serverisse (vs. HTTP, mis on ainult tõmbamiseks – pull-protokoll). HTTP puhul saadetakse kõik objektid eraldi vastustena. SMTP puhul on kõik objektid kapseldatud ühte vastusesse (MIME).

MIME (Multipart Internet Mail Extensions) – SMTP teadete kodeerimise viis, mis võimaldab edastada infot, mis ei ole 7-bitilises ASCII-s (graafika, AV). MIME toetab ka teisi kooditabeleid (KOI8-R, Unicode jne.) On ka eelnevalt defineeritud MIME-tüübid (gif, html, postscript, jne.).

SMTP sõnumi formaat

Päis – sisaldab infot kirja saatja, saaja, teema ja kuupäeva jms. kohta. MIME-kirja korral on lisatud read kasutatud MIME versiooni kohta, samuti kirja sisu kodeeringu tüüp. Lisaks määratakse ära, millist MIME-i alamtüüpi on kirja sisu (html, gif).
Sisu – kodeeritud vastavalt päises määratud kodeeringu tüübile.

Kui kasutatakse mitmeosalist MIME-i, on alamtüübiks multipart/mixed. Lisaks võetakse kasutusele eraldaja (boundary), mille abil tehakse vahet kirja erinevate osade vahel. Iga osa võib olla seejuures kodeeritud erinevalt ja kasutada isesugust MIME-i alamtüüpi.

Kirjade lugemiseks kasutatakse POP3 protokolli. POP3 võimaldab vaid näidata postkastis olevate kirjade arvu, lugeda ja kustutada suvalist kirja.

IMAP – meililugemisprotokoll, mis on suuremate võimalustega kui POP3 (kirjade „prügikasti" saatmine, lugemata ja loetud kirjade eristamine).

13. DNS

Domain Name System
Kasutab UDP-d (Saadetakse üksikuid pakette, ei kulu aega ühenduse loomiseks – kiire). Tegeleb domeeninimede teisendamisega IP-aadressideks. Töötab hajusandmebaasi põhimõttel (kogu info ei ole kunagi ühes serveris). Iga nimeserver haldab Internetis teatud piirkonda (domeeni). Andmebaas on mitmes serveris dubleeritud.

Dubleerimise põhjused:
Vähendada koormatust
Vähendada tõenäousust, et nimelahendus ei tööta.
Vahemaadest tingitud viivituste vähendamine.

Lokaalne (puhverdav) nimeserver – puhverdab nimeinfot, et parandada päringute kiirust korduvate päringute puhul.
Juurserverid – sisaldavad infot kõigi tippdomeenide (com, edu, ee jne.) kohta.

Autoritatiivne (authoritative) nimeserver on see server, mille *andmebaasis* on info domeeninime ja sellele vastava IP-aadressi kohta. Teised nimeserverid ainult puhverdavad antud andmeid (non-authoritative). Autoritatiivsest serverist saab alati vastuse nimepäringule.

Rekursiivne päring – kui nimeserver ei oma infot antud domeeni kohta, küsib ta järgmise serveri käest edasi jne., kuni vastus on käes. (See koormab serverit, võtab aega). Vastus tuleb alati sama teed mööda tagasi.

Iteratiivne (mitterekursiivne) päring – kui nimeserver ei tea antud domeeni IP-aadressi, siis saadetakse kliendile selle nimeserveri IP, kust edasi küsida.

Päringu saabumisel kontrollitakse alati kohaliku nimeserveri puhvrit. Kui seal vastust ei ole, käivitub tavaline päringute protseduur.

14. Usaldatav andmeedastus

Süsteem peab olema võimeline töötama ka juhul, kui osa pakette läheb kaduma või andmete ülekandmisel tekivad bitivead. Mitteusaldatava kanali karakteristikud määravad usaldusväärse protokolli (rdt) keerukuse.
rdt mudel: Aste-astmelt luuakse saatja aja vastuvõtja vahel turvaline andmeedastussüsteem. Selle loomisel arvestatakse ainult ühesuunaliste ühendustega ja selle graafiliseks kujutamiseks kasutatakse lõplikke automaate (finite-state machines – FSM).

rdt 1.0 – Töökindel kanal, kus ei ole bitivigu ja pakett ei lähe kaduma. Saatja saadab paketi kanalisse ja vastuvõtja saab selle kätte.

rdt 2.0 – Kanal, kus esinevad bitivead. Võivad esineda muutused bittides. UDP protokolli puhul kasutatakse kontrollsummat, et kindlaks teha moondunud bitte. Vastuvõtja peab saatma saajale kinnituse, kui pakett on vigadeta kohale tulnud (ACK) või kui pakett on vigane (NACK). Kui ACK-i antud paketi kohta ei tulnud või tuli NACK, tuleb paketti korrata. Kui moondub kviitung, on oht, et osad paketid saadetakse teistkordselt. Selle vältimiseks kasut. pakettide nummerdamist. Kadudeta süsteemis piisab pakettide eristamiseks vaid nullist ja ühest.

rdt 2.2 – Kasutatakse ainult ACK-kviitungeid. Iga kviitungiga pannakse kaasa paketi number, mille kohta antud kviitung käib. Kui ühte paketti kätte ei saadud, saadetakse välja teistkordne ACK juba varem kättesaadud paketi kohta. See on samaväärne rdt 2.0 NACK kviitungile. Saatja saab sel juhul teada, et üks pakett on moondunud ja seda tuleb korrata.

rdt 3.0 – Kanal, kus esinevad bitivead ja paketikadu. Kuna siin võivad kaduma minna nii andmed, kui paketi kviitungid, võetakse kasutusele taimer. Kui selle aja jooksul ei ole kinnitust tulnud, tuleb paketti korrata. Ka siin tuleb iga kviitungiga kaasa panna paketi järjekorranumber, mis välistab duplikaadid. rdt 3.0 raiskab ressurssi, kuna ooteajad on liiga pikad.

15. Go-back-n, selective-repeat

Vigaste pakettide korrigeerimine.

Go back-n: Kui paketi saatmine ei õnnestunud, minnakse tagasi n-paketi võrra ja korratakse kõike, mis juba saadetud. Paketi päises on ette nähtud väli identifikaatori jaoks. Kui väli saab täis, alustatakse otsast peale.

Aken – mitu paketti võib saata enne esimese kinnituse saabumist. Aken võib olla muutuva suurusega, mis sõltub saatja, vastuvõtja ja võrgu parameetritest.

Voo juhtimine – Määrtakse kindlaks, kui palju saatja võib saata ja kui palju vastuvõtja suudab vastu võtta.

Kui kasutatakse kumulatiivset ACK-i, siis sellise kviitungi saamine mõne paketi kohta kviteerib automaatselt ära ka kõik varasemad saadetud paketid.

Vastuvõtja jälgib saabunud pakettide järjekorranumbreid. Kui saabunud paketi järjekorranumber näitab, et eelnev pakett pole kohale jõudnud, ei saada vastuvõtja ACK teadet ja saabunud paketti ignoreeritakse. Akent nihutatakse ainult siis, kui saabub ACK teade ühele aknas olevale saadetud paketile, tõendades ka, et eelnevad paketid on kohale jõudnud. Kui teatud aja jooksul ei toimu akna nihutamist, st. akna esimestele pakettidele pole kinnitust tulnud, saadetakse kõik paketid uuesti. Paketid peavad olema saabunud vastuvõtjasse õiges järjekorras, vastasel juhul toimub pakettide uuesti saatmine alates paketist, kus viga ilmnes tänu akna kellale.

Selective Repeat: Korratakse ainult seda paketti, mida teine osapool kätte ei saanud. Puhverdamine keerulisem, kuna peab meeles pidama, millised paketid on käes ja millised ei ole. Saatja saadab uuesti ainult need paketid, millele ei saadud kättesaamise kinnitust. Iga paketi jaoks on eraldi kell.

Kui saabunud paketid on vales järjekorras, puhverdatakse need. Kui paketid on õiges järjekorras, nihutatakse akent edasi ja kviteeritakse need. Kui pakette ei kviteerita, ei saa saatja akent edasi nihutada. Akna pikkus on alati pool identifikaatorite arvust.

Akent nihutatakse alati siis, kui akna kõige esimene saadetud pakett on saanud ACK teate.

16. TCP

Transpordikihi protokoll, asub ainult lõppsõlmedes. Usaldusväärne ja töökindel. Kasutab punkt-punkt ühendust (üks saatja, üks vastuvõtja). Mõlemal poolel on omad puhvrid.

Kasutatakse duplekssidet. TCP on ühendusele orienteeritud (handshake). Nummerdatakse baite, mitte segmente, kasutatakse kumulatiivset kviteerimist. TCP-l ei ole eraldi ACK-segmenti.

Ühenduse loomisel valivad mõlemad osapooled endale ühe identifikaatori juhuslikest. Vastuvõtja informeerib saatjat, palju tal puhvris vaba ruumi on. Saatja püüab hoida kviteerimata andmehulka väiksemana sellest vabast ruumist.

Kui kviteerimata paketile saabub timeout, tuleb paketti korrata. Kui timeout on liiga lühike, koormatakse tipptunnil ilmaasjata võrku, kui on liiga pikk, siis muutub viivitus liiga suureks.

Ühenduse loomise protsess
- Klient saadab segmendi SYN (ident) ja valib esimese järjekorranumbri;
- Vastuvõtja saab SYN-i kätte, vastab SYNACK ja saadab oma järjekorranumbri ning eraldab vajaliku stacki;
- Klient saadab uue segmendi, mis kviteerib serveri vastuse ja eraldab oma mälus vajalikud puhvrid;

Ühenduse sulgemise saavad algatada mõlemad pooled:
- Klient saadab TCP FIN segmendi serverile;
- Server vastab ACK, sulgeb ühenduse ja saadab FIN-i;
- Klient vastab ACK, ja läheb „timed wait" olekusse – vastab ACK kõikidele FIN-idele;
- Server saab vastuse kätte ja lõpetab ühenduse.

17. TCP voo juhtimine.

Vastuvõtja informeerib saatjat, palju tal puhvris vaba ruumi on. Saatja püüab hoida kviteerimata andmehulka väiksemana sellest vabast ruumist. Oluline on optimaalne timeout. Kui see on liiga lühike, koormatakse võrku, kui on liiga pikk, muutub viide suureks. Iga paketi saatmisel võetakse aega: saatmine+kinnitus. Tehakse statistikat – arvutatakse kaalutud keskmine. Usaldatavuse tagamiseks lisatakse sellele mingi konstant.

18. TCP koormuse juhtimine

Erineb voo juhtimisest. Koormuse juhtimisega hajutatakse võrgu koormust, mitte konkreetsetes masinates olevat pakettide hulka.
Voo juhtimine – „garaažid täis", koormuse juhtimine – „ristmikud täis". Liiga palju allikaid saadavad rohkem andmeid, kui võrk välja kannatab.
Ooteajad hakkavad kasvama, puhvrid saavad täis, hakatakse andmeid ignoreerima. Võrk läheb umbe eksponentsiaalse kiirusega, sest time-outide tõttu hakatakse pakettide saatmisi kordama.

Kui ruuteri puhvrid on täis, siis kõik saabunud paketid lähevad kaduma. Seega tuleb saatmist korrata. Tegelikkuses kasutatakse efektiivselt 2/3 või veel vähem maksimaalsest võimsusest.

Reguleerimine
Punkt-punkt – transpordikiht ei saa teada, kui suur on tegelik koormus. Seda hinnatakse kaudselt pakettide kadumise ja viidete järgi. (TCP-s). Võrgukiht võib anda ka tagasisidet (nt. ruuterites. Kasutatakse nii ATM-s kui TCP-s).
ATM-s kasutatakse available bit rate'i. See on kättesaadav edastuskiirus. Lisaks andmetele saadetakse ka halduspakette.

Pakettide saatmisel proovitakse koormust suurendada (nihutades akent suuremaks). Kui tekib ülekoormus, muudetakse aken jälle väikseks tagasi ja proovitakse uuesti. Ülekoormuse vältimiseks hakatakse pärast teist piirile jõudmist akent suurendama lineaarselt.

19. UDP

Transportkihi protokoll. UDP puhul võivad segmendid kaduma minna või kohale jõuda vales järjekorras. Connectionless – ühendust ei looda. „Best effort" – püüab antud tingimustel anda oma parimat. UDP on lihtsaim ja kiireim. Lühem segmendi päis. (8-baidine) Võrgus ei toimu koormuse reguleerimist! Seega võib võrgu umbe ajada. Kasutatakse DNS-is ja SNMP-s. UDP tegeleb vigade avastamisega (UDP checksum), aga mitte vigade parandusega, seda peaks tegema rakenduskiht. UDP-d kasutatakse lühikeste andmete edastamiseks.

20. Marsruutimine

Optimaalse tee valimine. Peab olema korrektne, õiglane, lihtne, stabiilne (üritab jagada ressursse nii, et ei tekiks ummikuid), veakindel, optimaalne ja efektiivne.
Jõudluse kriteeriumid: lõikude arv, hind, viide, läbilaskevõime.

Marsruutimine koosneb kahest põhilisest komponendist: optimaalse marsruutimistee kindlaksmääramine ja andmepakettide transport ehk kommuteerimine (*switching*). Kui andmepakettide transport on küllaltki triviaalne toiming, siis optimaalse marsruutimistee leidmine võib olla vägagi keerukas. Marsruutimistee kindlaksmääramisel kasutatakse mitmesuguseid erinevaid mõõte (algoritmisliste arvutuste resultaate, näiteks tee pikkust) või mõõtude kombinatsioone. Marsruutimisalgoritmide tarkvara arvutab optimaalse tee leidmiseks marsruutimismõõte.

Tee määramiseks kasutavad marsruutimisalgoritmid marsruutimistabeleid, mis sisaldavad algoritmist sõltuvat marsruutimisinformatsiooni. Marsruutimisalgoritmid täidavad need tabelid mitmesuguse informatsiooniga. Näiteks tabel, kus igale võrgu numbrile on vastavusse seatud marsruuteri port, aitab marsruuterit otsustada, missugusesse porti missugune andmepakett suunata. Marsruutimistabelid võivad sisaldada ka muud informatsiooni, näiteks ühenduste või teede mõõte.

Selleks, et hoida marsruutimistabelites ajakohast informatsiooni, suhtlevad marsruuterid omavahel mitmesuguste sõnumite vahetamise teel. Üheks niisuguseks sõnumiks on marsruutimisvärskendus (*routing update*). Analüüsides kõikidelt marsruuteritelt saabuvaid marsruutimisvärskendusi, saab marsruuter kokku panna pildi võrgu topoloogiast. Teiseks niisuguseks näiteks on lüli oleku kuulutuse (*link state advertisement*) sõnum, mis täpsustab võrgu pilti ühenduste koormatuse ja kvaliteedi osas. Sellist informatsiooni kogudes ja süstematiseerides saab marsruuter leida optimaalseid teid võrgu sihtpunktidesse.

21. IPv4 ja IPv6

IP-l on kaks peamist ülesannet – pakkuda ühendusevaba võimaluste piires parimat datagrammide kohaletoimetamist ning pakkuda (de)fragmenteerimist, et võimaldada andmeedastust erinevate maksimaalse andmeühikuga (MTU) võrkudes.

IPv4 – Igale võrgusõlmele eraldatakse üks 32-bitine unikaalne aadress, mis on jagatud kaheks loogiliseks osaks: võrgu- ja hostiosaks. Võrguosa identifitseerib konkreetse alamvõrgu, hostiosa aga konkreetse masina selles alamvõrgus. IP aadress on jagatud neljaks osaks, mis on üksteisest eraldatud punktiga. Igat konkreetset võrku saab omakorda jagada alamvõrkudeks. Alamvõrgu täpse suuruse määrab kasutatav võrgumask. Võrgumaski kahendväärtuse ja IP aadressi kahendväärtuse loogiline korrutamine annab alamvõrgu esimese aadressi – alamvõrgu aadressi.

IPv6 – 32-bitine aadressiruum ammendub lõplikult 2008. aastaks. IPv6 päise formaat peaks kiirendama pakettide töötlust ja edastamist. Päist on muudetud, et see hõlbustaks QoS kasutamist. Kasutusele on võetud uus „anycast" aadress, mis peaks võimaldama valida optimaalsema tee üheni mitmest võimalikest serveritest. IPv6 puhul ei ole lubatud fragmenteerimine, kasutatakse 40-baidilist päist.

Erinevused
IPv6 on täielikult ära kaotatud kontrollsumma, et vähendada töötluseks kuluvat aega. Kõik lisavalikud on küll lubatud, kuid asuvad väljaspool päist. Neile viidatakse väljaga „Next

Header". Kasutusele võetakse ka ICMPv6, mis sisaldab täiendavaid teateid (nt. „Packed too big"), samuti administreerimist multisaategruppide kaupa.

Üleminek IPv4 IPv6-le
Mitte kõiki ruutereid ei ole võimalik korraga uuendada, s.t. tekib segatud võrk (IPv4+IPv6). Kasutatakse kahestackilisi ruutereid, mis võimaldavad pakette teisendada ühest aadressiruumist teise. Teine võimalus on kasutada tunneleid, kus IPv6 paketid liiguvad kapseldatuna IPv4 sees.

/---/

26. Datagrammide edastus läbi võrkude

Igas IP datagrammi päises on kirjas saatja ja saaja aadressid. Selle järgi toimetatakse pakett konkreetse masinani.

Igas seadmes on olemas oma ruutimise tabel, mille alusel otsustatakse, kuhu pakett vaja toimetada on.
Kui saadetakse välja pakett, mis on mõeldud mõnele samas võrgus asuvale terminaalile, siis toimetatakse see vahetult kohale. Kui sihtarvuti ei asu samas võrgus, saadetakse see edasi võrguväravasse (gateway), mis uurib, kas paketti on võimalik vahetult edasi toimetada (s.t. kas sihtarvuti asub samas alamvõrgus, mis gatewaygi). Kui see pole võimalik, saadetakse pakett edasi järgmisesse ruuterisse (see tehakse kindlaks samamoodi gateways asuva ruuditabeli põhjal). Nii toimitakse senikaua, kui pakett on jõudnud sellesse alamvõrku, kus asub sihtarvuti ja see on võimalik vahetult kohale toimetada.

27. Vigade avastamine ja parandamine

EDC (error detection and correction bits) – liiasus, mida on vaja selleks, et vigu parandada.

Paarsuse kontroll
Ühedimensioonilise paarsuse kontrolli korral on võimalik avastada paarituarvu bittide moondumist. Samas ei ole võimalik kindlaks teha, milline bittidest täpselt moondus. Kahedimensioonilise paarsuse kontrolli korral on võimalik vigu parandada, kui moondunud on üks bitt.

Interneti kontrollsumma
Eesmärk on avastada vigu (nt. moondunud bitte) saadetud segmendis. Saatja implementeerib segmendi sisu kui 16-bitist täisarvu. Kontrollsumma arvutamiseks teostatakse komplementaarne ühtede liitmine, tulemus paigutatakse UDP kontrollsumma väljale. Vastuvõtja arvutab analoogiliselt andmete kontrollsumma ja võrdleb seda paketi päises olevaga. Kui need on võrdsed, siis viga ei ole.

Tsükliline liiasuse kontroll
Arvutatakse CRC kontrollsumma. Peaaegu võimatu on juhuslike bitimuudatuste tulemusena saada sama kontrollsummat. Andmeid käsitletakse bitijadana. Esimesed 8 bitti laaditakse arvuti registrisse ja teostatakse XOR-tehe. Esimeseks operandiks on registris olevad 8 bitti, teine on vabalt valitud polünoom, mis peab olema teada ka andmete saajale (et oleks võimalik sama arvutus paketi saamisel ka läbi viia). Tehte tulemus salvestatakse uuesti registrisse, selle järel nihutatakse registri sisu vasakule ja madalamale järgule salvestatakse uus andmebitt. Tekkinud arvuga tehakse uuesti XOR-tehe (kasutades sama polünoomi) ja tulemus salvestatakse uuesti registrisse. Tsüklit korratakse senikaua, kuni andmeid jätkub. Antud tsükli lõppedes on registris kontrollsumma, mis salvestatakse paketi päisesse. Vastuvõtmisel teostatakse sama operatsioon. Kui saadakse päises identne tulemus päises olevaga, ei ole andmete sisu moondunud.

28. Lokaalvõrgud. Topoloogiad

Topoloogia – kuidas on võrgud füüsiliselt üles ehitatud

Siinivõrk (bus) – kõik arvutid asuvad ühe liini peal. Siinivõrgu otstes asuvad terminaatorid, mis tagavad signaali leviku magistraalkaablis.
Ringvõrk (ring) – peamagistraal, kuhu kõik arvutid on ühendatud, moodustab ringi. Iga ühenduse juures, mis viib arvutini, asub repiiter e. võimendi.
Puu (tree) – peamagistraali küljest hargnevad harud, mille külge on ühendatud arvutid. Puu harud moodustavad omaette siinivõrgud.
Täht (star) – Kõik arvutid on ühendatud ühe kesksеadme külge, moodustades tähekujulise struktuuri.