Kommunikasjonsteknologi, Tjenester og Nett

Kompendie 2016

Av Mats Jørgen Skaslien

Basert på Kurose & Ross: Computer Networking – A top down approach (6th edition)

Innholdsfortegnelse

Forfatterens kommentarer:

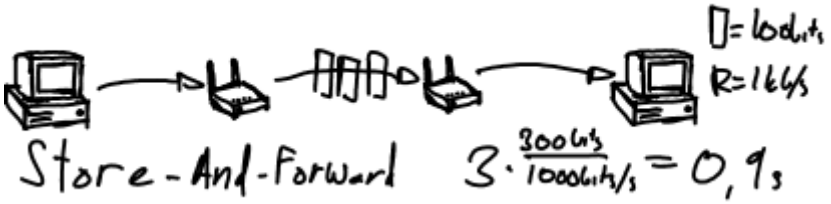
Dette kompendiet er først og fremst skrevet for egen læring. Det har veldig mye til felles med den fantastiske Wikipendium siden til KTN, fordi den simpelthen oppsummerer faget bra. Når det er sagt er det mye informasjon her fra boka og wikipedia samt andre sider som Wikipendiumsiden *ikke dekker*. Oppsummeringen er skrevet på engelsk da jeg egentlig skrev den for å legge ut på nettet. Det er mulig det er noen faktafeil, skrivefeil og ukjente typer feil i dette kompendiet – ikke ta alt for god fisk.

Håper du får nytte av dette kompendiet.

Kapittel 1

Overføringshastighet

For å sende **L** bits over en link med hastighet **R** tar det $\frac{L}{R}$ sekunder. Om ***Store-and-forward*** benyttes må hele pakken mottas før den sendes videre. Vi må da legge på en multippel ***N*** som er antall links (koblinger mellom rutere/datamaskiner), og får formelen $\frac{NL}{R}$



Figur 1 Store-And-Forward illustrert med tre links. L=300, R=1kb/s, N=3

Delay

Delay kan deles opp i fire hovedtyper:

- d_{proc} – Processing delay:
  Tiden det tar for en ruter å sjekke pakkeheaderen, om det er feil på pakken og hvor den skal sendes videre.

- d_{queue} – Queuing delay:
  Hvis pakker ankommer en ruter/switch raskere enn den kan videresende danner det seg en kø. Queuing delay er tiden det tar fra en pakke ankommer køen til den er klar til å videresendes.

- d_{prop} – Propagation delay:
  Dette er tiden en pakke bruker i linken. Dette begrenses av type link og lysets hastighet.

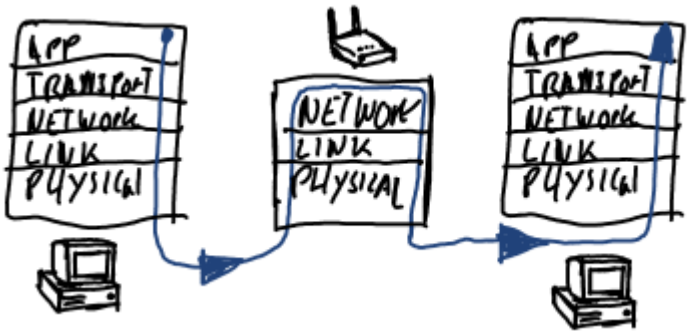
- d_{trans}– Transmission delay:
  Tiden det tar for en ruter å dytte en pakke ut på linken.

Total delay kalles $d_{end-to-end}$ og er summen av delayene ganget med antall links (gitt at det er samme delay på alle links). Ved flere rutere bruker man d_{trans} til den tregeste ruteren, da dette blir en flaskehals.
$d_{end-to-end} = N(d_{proc} + d_{trans} + d_{prop} + d_{queue})$

Nettverkslag

Nettverket er delt opp i fem lag med forskjellige protokoller:

Navn	Forklaring	Protokoller
Application Layer	Toppnivå-protokoller som direkte samhandler med programvare	HTTP, SMTP, FTP, DNS
Transport Layer	Sørger for end-to-end kommunikasjon	TCP, UDP
Network Layer	Fikser alt som ligger mellom hosts, eks. hvor en pakke skal sendes videre for å nå sin destinasjon. Øverste lag i en ruter	IP
Link Layer	Protokoller for hvordan data skal sendes i det fysiske laget	Ethernet, 802.11/b/g/n
Physical Layer	Det fysiske laget, altså hva slags type kabel eller antenne som brukes	Twisted pair, coax, fiber, wi-fi..



Figur 2 Informasjonsgangen i et nettverk

Kapittel 2 - Application layer

Dette laget er som navnet tilsier, det laget som kommuniserer direkte med programmene på datamaskinen din. For å sende og motta informasjon må et program ta i bruk en eller flere **sockets**. En socket et endepunkt mellom to kommuniserende prosesser, man kan se på det som en ende av en slange som det sendes strømmer av informasjon gjennom. **En socket benytter seg av portnummer og IP-adresse** for å bestemme hvor den skal koble seg til.

Protokoller

HTTP

HyperText Transfer Protocol bygger opp nettsider. Når man kobler seg til en nettside får man først tilsendt en **HTML** fil. Denne er skjelettet til nettsiden, og sier hvordan den er strukturert.
Etter denne kommer objektene på siden i tur og orden.

HTTP støtter **både persistent og non-persistent tilkoblinger**. Ved persistent holdes en kanal åpen mellom server og klient, og alle forespørsler og svar blir sendt gjennom denne. Ved non-persistent åpnes en ny tilkobling for hver forespørsel og svar.

HTTP støtter **cookies** og **web caching**, og er **stateless**.

FTP

File transfer protocol er protokollen som brukes for filoverføring over internett. Den bruker to TCP-forbindelser samtidig, der en står for kommunikasjon mellom klient og tjener, mens den andre sender filene som blir forespurt. FTP er **ikke stateless**, altså den må vite om klientens tilstand. Dette betyr hvorvidt klienten fortsatt er koblet til FTP serveren, og hvilke kommandoer klienten har sendt.

STMP

Simple mail transfer protocol er protokollen som brukes for e-post. Denne krever at all informasjon er i 7-bits ASCII, noe som betyr at bokstaver som «æøå» og vedlegg må dekodes til ASCII. For dette benytter vi **MIME** – multipurpose internet mail extensions.

DNS

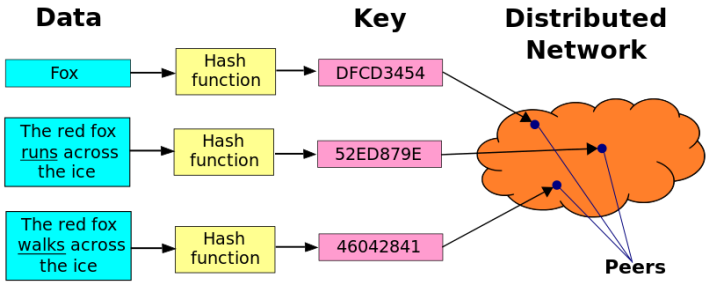
Domain name system er internettets gule sider- den knytter domenenavn til IP-adresser. DNS benytter seg kun av UDP tilkoblinger. Når en klient vil koble seg til «eksempel.com» sender den først denne adressen til en DNS server som oversetter adressen til en IP-adresse klienten kan koble seg til.

DNS tar seg også av **aliasing**. Dette er om du har kjøpt adressen «eksempel.net», og vil at denne skal sende brukeren til «eksempel.com». Net adressen er dermed aliaset til com adressen.

Det finnes et stort hierarki av DNS servere, med de 13 rot DNS-serverne på toppen. Under dette faller servere for alt for land til byer og institusjoner. Dette er for å begrense trafikk og ventetid på oppslag, og å ikke ha et enkelt point of failure.

Peer-to-peer

I peer-to-peer nettverk er det ingen sentral server, men et nettverk der alle deltagere er både tjenere og klienter (så sant de seeder …). Brukerne av netteverket ser hverandre ved hjelp av en tracker. Filene som deles blir delt opp i biter (**chunks**), og hver bit får sitt kjennetegn (**hashkey**). Hver bruker får en oversikt over hvem som har hvilke haskeys, og dermed bitene assosiert med de. Dette kalles et **Distributed Hash Table**.



Figur 3 Et Distributed Hash Table (DHT)

Distribution time

Dette er tiden det tar å sende alle klienter i et nettverk en fil. Her er to formler, en for et klient-tjener nettverk, og et peer-to-peer nettverk.

$$T_{CS} = Max(\frac{N * F}{U_S}, \frac{F}{d_{min}})$$

$$T_{P2P} = Max(\frac{N * F}{U_s + N * U_{total}}, \frac{F}{U_s}, \frac{F}{d_{min}})$$

F = filstørrelse, N= antall klienter, U_s= Opplastingshastighet server, U_{total}= Total opplastingshastighet for peers, d_{min}=Laveste nedlastningshastighet for klient i nettverket

Tiden det tar å spre en fil i et nettverk er altså gitt av den brøken som er størst av de som er listet opp for et gitt P2P/CS nettverk. Disse formlene, om du bytter ut likhetstegnet med et ≥ vil angi minste mulige distribution time.

Kapittel 3 - Transport Layer

Multiplexing and Demultiplexing

Når transportlaget mottar data fra applikasjonslaget skjer dette via en eller flere sockets. Deretter pakker transportlaget den inn med en transportlagheader. Dette kalles multiplexing.

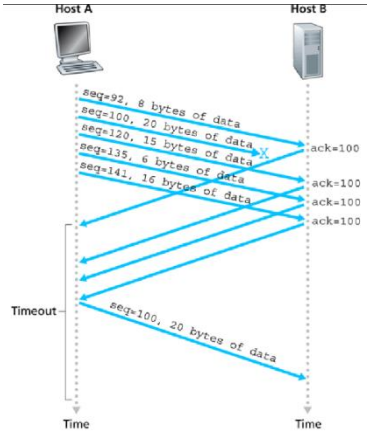
Dataen med den nye headeren sendes så videre ned til nettverkslaget, og til destinasjonen sin.

Ved destinasjonen blir dataen ført opp igjen fra nettverkslaget til transportlaget. Her leses transportlagheaderen fra senderen, og mottaker bruker denne informasjonen til å føre dataen videre opp til applikasjonslaget via riktig socket(s).

Principles of Reliable Data Transfer

ARQ (Automatic Repeat reQuest) protokollen brukes til å sjekke om mottatt informasjon inneholder feil. For å gjøre dette trenger vi tre ting:

- Error detection
- Reciever feedback – muligheten til å sende en tilbakemelding til avsender. En **ACK** sendes om pakken ble mottatt og alt er bra, en **NAK** sendes om det blir oppdaget feil.
- Retransmission – muligheten til å sende tapt/ korrumpert informasjon på nytt.
- Timeouts - forsikrer oss om at data blir sendt på nytt hvis ikke en ACK har blitt mottatt innenfor en gitt tidsramme.



Figur 4 Et eksempel på en forbindelse med pålitelig dataoverføring

Det settes sekvensnummer (SEQ) på hver enkelt pakke. Ved hjelp av de tre overnevnte metodene kan vi oppdage når og hvilken pakke som har gått tapt, og sende den på nytt.

ARQ protokoller

- **Stop-and-wait:** Én pakke sendes om gangen. Når ACK mottas sendes neste. Om en pakke tapes sørger timeout for at den sendes på nytt.
- **Go-back-N:** Her sendes et «vindu av pakker» der de første i vinduet må ACK'es før vinduet flyttes frem. Med vindusstørrelse på 8 vil ikke pakke 4 bli sendt før pakke 0-3 er ACK'et.
- **Selective Repeat:** Mottaker sender ACK på alle pakker, og kun de som er tapt sendes på nytt. Benytter seg som Go-back-N av pakkevinduer.

Transportlagprotokoller

UDP

User Datagram Protocol er en forbindelsesløs overføringsprotokoll. Den er omtrent så minimal som den kan bli: Det er ingen handshake, det er ingen garanti for rekkefølge, levering eller duplikatpakker. Protokollen legger på en header med kilde- og destinasjons- IP på segmentet.

UDP fokuserer på rask levering og høyt throughput over pålitelighet.

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															

Figur 5 UDP header. Felt i rødt er valgfrie i IPV4. Kun Source Port er valgfritt i IPV6

UDP benytter seg av cheksummer for å avgjøre om dataen har blitt korrumpert på veien, og portnummer for å benytte seg av ulike tjenester hos tjener/klient.

UDP cheksum fungerer på følgende måte:

1. Summer Src port, Dst port og length
2. Adder eventuell overflyt fra 1. med summen av 1.
3. Inverter alle bits

For å avgjøre om feil har skjedd:

1. Adder Src port, Dst port, length og checksum
2. Hvis resultatet er 16 enere er pakken feilfri.

TCP

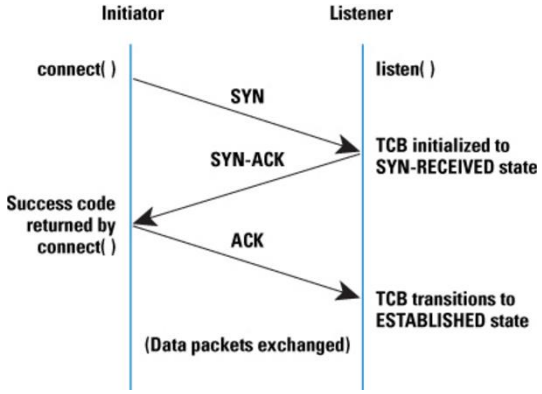
Transmission Control Protocol er en forbindelsesorientert overføringsprotokoll. Den tilbyr pålitlig, i-rekkefølge og feilkontrollert dataoverføring. Til gjengjeld er TCP tregere enn UDP.

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																	Urgent pointer (if URG set)														
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

Figur 6 TCP header

Tcp har en rekke egenskaper:

- Full-duplex service: Kommunikasjon kan gå fra klient til tjener og tjener til klient samtidig.
- Point-to-point: Det finnes bare én sender og én mottaker i hver forbindelse
- Three way handshake: A sender et segment til B, B svarer, A svarer svaret og tilkoblingen er skapt.
- Filer større enn *maximum segment size* blir delt opp i flere segmenter ved sending, og satt sammen igjen på mottakersiden.



Figur 7 SYN-ACK three way handshake

Sekvensnummeret på en pakke vil være det forrige sekvensnummeret, pluss størrelsen (length) av forrige mottatte pakke. Den første pakken vil ha et vilkårlig sekvensnummer. ACK nummeret som blir sendt tilbake er sekvensnummeret til neste pakke mottakeren forventer å få.

Three-way-handshake fungerer ved at synkroniseringspakken (SYN) blir sendt fra klient til server, her er SYN biten satt til 1. I tillegg sender klienten med et tilfelig generert segmentnummer av sikkerhetsgrunner. Serveren åpner et buffer for klienten, og svarer med en ACK der SYN biten er satt til 1, kalt en SYNACK pakke. Klienten åpner så et buffer for serveren og sender en ACK med SYN biten lik 0, og handshaket er gjennomført.

Når TCP-koblingen skal avsluttes sender klienten en pakke med FIN-biten satt til 1. Deretter venter klienten på en ACK fra serveren. Når denne mottas venter klienten på en pakke til, nå med FIN-biten satt til en. Klienten sender så en ACK, om denne timer ut er forbindelsen avsluttet.

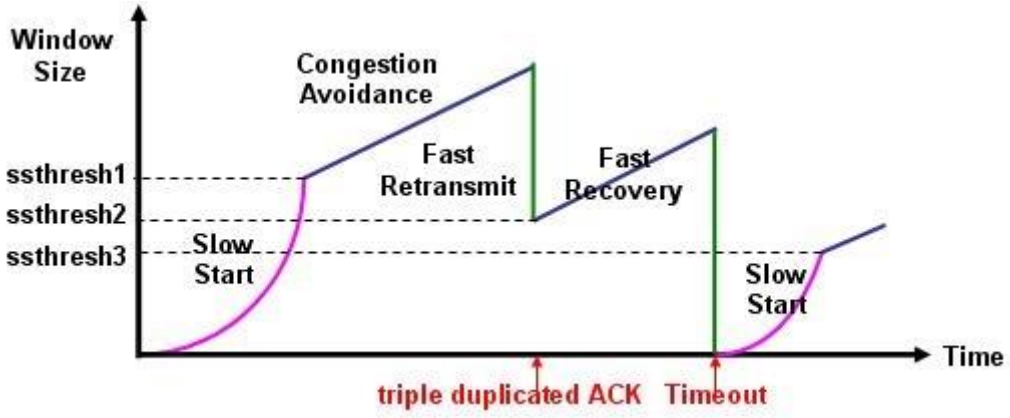
Congestion Control

Congestion – også kalt kø eller forstoppelse på godt norsk – skjer når ruteren ikke klarer å sende unna pakker like fort som den mottar dem. Dette fører til at ruterens buffer blir fullt, og ikke kan ta imot flere pakker. Disse blir da tapt.

Det er to typer congestion control:

- End-to-end congestion control: Transportlagene i endesystemene må finne ut om det er stor pågang i nettverket, da nettverkslaget ikke selv har noen oversikt over dette. Transportlagene må.
- Network-assisted congestion control: Rutere kan sende tilbakemelding om hvor stor pågang det er, for eksempel ved å sende en choke packet.

TCP congestion control skjer via end-to-end metoden. Her benyttes tre metoder; **slow start, congestion avoidance** og **fast recovery**. Kort sammenfattet: TCP starter med å sende et relativt lite antall pakker. Så lenge alt går greit øker throughputen eksponensielt til den når en satt grenseverdi, deretter øker den linjært frem til feil oppdages. Ved feil faller den tilbake til en annen satt grenseverdi, før den øker igjen. Se figur.



Figur 8 TCP congestion control

Kapittel 4 - Network layer

Virtual Circuit and Datagram Networks

En **virtual circuit** gjør det mulig å sende data over et nettverk på en slik måte at det virker som om det er en direkte kobling mellom kilde og destinasjon. En VC består av:

1. En vei fra kilde til destinasjon mellom en serie av linker og rutere
2. VC nummere, ett for hver link på veien
3. En Forwarding tabell på hver ruter på veien, som sier hvor neste «stopp» på stien er.

Pakker sendt over en VC får et VC-nummer på headeren. Dette kalles linknummeret, og endres for hver ruter for å få rett nummer for neste link.

Et **datagram network** tilbyr tilkoblingsløse tjenester på nettverkslaget. Pakkene merkes med destinasjonen og sendes ut i nettverket. Det er etter det opp til ruterene å bestemme hvilken vei pakken skal ta for å komme frem.

What's inside a router?

Moderne rutere består av fire hoveddeler:

1. Input porter: Her kommer pakker inn og blir plassert i bufferet. Her skjer også oppslag i forwardingtabellen og linklayerfunksjoner.
2. Switching fabric: Denne delen kobler input- og outputporter sammen og bestemmer dermed hvor data skal sendes videre.
3. Output porter: Fysisk utgang, også her er det en buffer
4. Routing prosessor: Oppdaterer routingtabellen

The Internet Protocol

Når en pakke ankommer en router blir den splittet i ene enden, og satt sammen igjen i andre.

Datamaskiner koblet til en ruter kalles et subnet. Disse deler en del av en IP adresse, og har sin unike bit i det som kalles **subnet masken.** I adressen 223.1.1.0/24 er de første 24 (Av 32) bitene satt, så datamaskinene i subnettet vil ha sin unike adresse i de siste 8 bitene.

En datamaskin med **dynamisk IP adresse** vil, når den blir koblet til et nett, broadcaste til hele nettet at den ønsker en adresse. En DHCP (Dynamic Host Configuration Protocol) server vil så svare, og tilby en adresse. Klienten godtar så denne adressen og broadcaster ut på nettet at den nå kan nås på den adressen.

Routing algorithms

Routingalgoritmer prøver å finne raskeste vei mellom endepunktene. Vi har

- Global routing algorithms; Disse har oversikt over hele nettverket, og foretar en beregning for å finne korteste vei fra A til B
- Decentralized routing algorithms har kun oversikt over sine nabonoder, og finner korteste vei ut i fra dette.

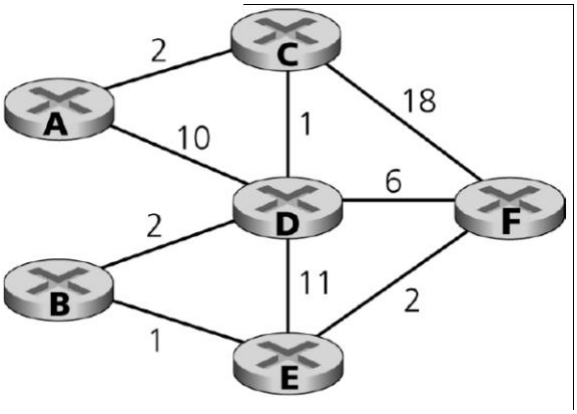
De er også delt inn i:

- Static routing algorithms: Satte ruter informasjon skal flyte
- Dynamic routing algoriths: Endrer seg ut ifra trafikken og routere som blir tilkoblet/ frakoblet. Responsive til forandring, men kan skape problemer ved å sende data i løkker, og få oscillasjoner i congestion.

Eksempel fra pensum – Dijkstras' Routing Algorithm:

Dijkstras algoritme forteller deg at du skal alltid gå minste kostnadsvei. Du fortsetter med dette til du har besøkt alle nodene i nettverket.

Til høyre ser du en relevant eksamensoppgave. Her skal vi bruke algoritmen til først å finne korteste vei fra D til alle nodene, og deretter gjøre det samme for A. Resultatet er tabellen nedenfor.



Figur 9 Et nettverk av noder. "Reisekostnad" for hver link er angitt

Korteste vei fra D til nettverket			**Korteste vei fra A til nettverket**		
Steg	**Sti**	**Korteste vei**	**Steg**	**Sti**	**Korteste vei**
0	**D**		**0**	**A**	
1	**DC**	**C=1**	**1**	**AC**	**C=2**
2	**DB**	**B=2**	**2**	**ACD**	**D=3**
3	**DBE**	**E=3**	**3**	**ACDB**	**B=5**
4	**DCA**	**A=3**	**4**	**ACDBE**	**E=6**
5	**DBEF**	**F=5**	**5**	**ACBDEF**	**F=8**

Kapittel 5 - Link layer

Link laget – kalt «Data Link layer» i OSI-modellen – står for transporten fra node til node. En god analogi finner man i ferieplanlegging:

Du bor i Oslo og skal på ferie til Madrid og får et reisebyrå til å organisere turen din. Reisebyrået kjøper en togbillett fra Oslo til Gardermoen, flybilletten fra Oslo til Madrid og taxien fra flyplassen til hotellet ditt.

Det er toget sitt ansvar å få deg til Gardermoen, flyselskapet sitt ansvar til Madrid, og taxien sitt ansvar å få deg til hotellet. Hver av disse tre turene er «direkte» mellom to punkter som er ved siden av hverandre. Merk også at alle forekomstmidlene er forskjellige, men de er like i det at de frakter personer fra punkt A til B. I denne analogien er du datagrammet og hvert ledd i reisen er en link. Fremkomstmiddelet er en link-lagsprotokoll og reisebyrået er routing protokollen.

Services provided by the Link Layer

- **Framing**. Dette er innpakkingen av datagramet fra nettvekslaget slik at det blir em frame. Med innpakking menes det at datagramet får både en header, og en «tail» tilknyttet seg. Selve strukturen på framen varierer fra protokoll til protokoll.

- **Link access.** MAC-protokollen (medium access control) spesifiserer når man kan sende en frame over linken. Dette er ikke et problem før flere noder deler samme link.

- **Reliable delivery.** Ved bruk av linker med høy feilrate som f.eks. trådløse linker, tas reliable delivery tjenester i bruk. Dette funker ganske likt som TCP leveringen, med ACK'er og retransmissions

- **Error detecion and correction.** Såkalte «error detection bits» blir plassert i framen av avsender som mottaker kan sjekke. Dette skjer på hardwarenivå. Videre, om det skal forteas **error correction** må mottaker avgjøre hvor i framen feilen har skjedd, og så fikse den.

Error-detection and –correction techniques

- Parity checks: I tillegg til pakken blir det sendt med en ekstra bit, slik at det finnes et odde (odd parity) eller partall (even parity) antall enere. Denne moteden oppdager kun feil om det har skjedd i et odde antall bits.
- Two-dimensional parity: Dataen som blir sendt deles opp i rader og kolonner. Hver rad og kolonne har sin egen parity bit som funker på samme måte som ovenfor. Dette tillater også å rette opp enkeltfeil.
- Cyclic Redundancy check. Sender og mottaker avtaler først et tall G. For hver D antall bits legges det til r bits, R, slik at $\frac{D*2^r}{G} = R$

Multiple Access Links and Protocols

Det finnes to typer nettverkslinker: Point-to-point, og broadcast. Førstnevnte består av kun én sender og én mottaker, mens sistnevnte har opptil flere sendere og mottakere. Det blir da fare for at nodene «snakker i munnen på hverandre». For å løse dette finnes det en rekke protokoller som gjør det lettere å dele nettverket, disse kan deles inn i tre hovedkategorier:

1. **Channel partitioning protocols:**
   1.1. **Time-division multiplexing (TDM)**. Si at en kanal støtter ***N*** noder, og at den har en båndbredde på ***R*** bps. TDM deler tiden inn i **time frames** og deler videre hver frame inn i ***N*** **time slots**. Hver time slot blir så fordelt mellom de N nodene, og hver node «snakker» kun i sin tilegnede time slot. Maks gjennomsnittlig overføringshastighet blir dermed $\frac{R}{N} = \frac{maks\ overføringshastighet}{antall\ noder}$

   1.2. **Frequency-division multiplexing (FDM)**.Her deles en kanal opp i frekvenser, hver med en kapasitet på $\frac{R}{N}$.

   1.3. **Code-division multiple access (CDMA).** Her fordeles en unik kode til hver node, som i tur enkoder dataen sin med denne koden. Dette gjør at hver node kan sende med nettets fulle hastighet samtidig.

2. **Random accsss protocols:**
   Her forsøker hver node å overføre med nettverkets fulle hastighet samtidig. Når det skjer en kollisjon venter hver av nodene involvert i kollisjonen en tilfeldig mengde tid før den prøver å sende på nytt.

   2.1. **Slotted ALOHA:** Deler tiden opp i time slots store nok til å sende akkurat én frame av gitt størrelse. Alle nodene synkroniseres til denne tiden, og kan kun begynne å sende i starten av en slot. Hvis det skjer en kollisjon, blir alle noder informert før framen er over. For hver frame etter en kollisjon, har nodene involvert i kollisjonen en sannsynlighet på p for å prøve å sende pakken på nytt.

   2.2. **ALOHA (Pure ALOHA)**: som slotted, bare uten tidsoppdelingen. Den er dermed helt desentralistert. Denne kom først, og er på generell basis dårligere enn slotted.

   2.3. **Carrier Sense Multiple Access (CSMA)**: her merker nodene seg om noen «snakker», og venter høflig til de er ferdig før en node prøver å sende sin informasjon. Kollisjoner kan på tross av dette oppstå; Hvis A og B begynner å sende data innenfor et tidsrom som er mindre enn $d_{prop}(A \rightarrow B)$ vil signalene kollidere før en av nodene vet at den andre sender data.

   2.4. **CSMA/ Collision Detection (CSMA/CD):** Som ren CSMA, men her vil en node slutte å transmittere så snart den oppdager at noen andre sender samtidig. Bruker så **exponential backoff** algoritmen for å finne ut når den skal prøve å sende igjen – for hver kollisjon vil den vente lenger og lenger med å sende.

3. **Taking-turns Protocols:**
   Ingen sender med mindre noden får beskjed at det er dens tur.

   3.1. **Polling protocol:** En master-node sier når en gitt node kan sende et bestemt antall frames.
   3.2. **Token-passing protocol:** Ingen master-node. Fungerer som en stafett der hver node sender et gitt maks antall frames, for så å si til neste node i rekken at det er dens tur.

Switched Local Area Networks

Switched Local Area Networks er det vi ser på som subnet til en ruter. Innad i dette subnettet adresseres pakker med **MAC-adressen**. Dette er en unik, permanent adresse alle enheter med et nettverksadapter har.

Address Resolution Protocol (**ARP**) er protokollen som knytter MAC-adresser til IP-adresser. Når en datamaskin vil vite MAC-adressen til en annen i nettet skjer følgende utveksling. Datamaskin A spør alle enheter på subnettet «Hva er MAC-adressen til enheten med IP-adresse X?». De det ikke gjelder ignorerer meldingen, og den det gjelder svarer med MAC-adressen sin. Denne MAC adressen blir så knyttet til IP-adressen og lagret i A sin ARP-tabell. Denne ARP-tabellen oppdateres kontinuerlig.

SUMMARY

Internet protocol suite:

1. Application layer:
   Commonly used protocols include ***HTTP, FTP and SMTP***. To send information the data from this layer is passed into a socket, which are created in the:

2. Transport layer:
   The two main protocols here are ***UDP and TCP***. The data from the application layer is passed down here and are multiplexed; a header containing a port number and a source- and destinationIP is added, with the application layer data as payload. When data arrives here from the network layer, it is demulitplexed and passed up to the application layer.

3. Network layer:
   The main protocol here is the ***Internet Protocol*** (IP), of which there are two versions in use today - IPv4 and IPv6. The job of the network layer is for *outgoing packet*s to: determine the next node to pass the data to and for *incoming packets*: to pass the packets up to the transport layer.

4. Link layer:
   The link layer provides ***framing*** which in short adds a header and a tail to the payload passed from the network layer, and «***de-frames***» incoming packets. The link layer then passes the frame onto the next node specified by the network layer.

   It also provides ***error detection and –correction, flow control*** and ***multiple access control***.

5. Physical layer:
   The physical medium through which information passes. This includes amongst others: **fiberoptic cables, twisted pair copper cables, wireless** signals and so on.

KILDER:

Kurose, James F. & Ross, Keith W. (2012) Computer Networking: A Top-Down Approach (6th Edition). Pearson.

Wikipendiumsiden til TTM4100, hentet vår 2016 - https://www.wikipendium.no/TTM4100_Kommunikasjon__Tjenester_og_nett/nb/#kapittel-1-computer-networks-and-the-internet

For mye annet, denne wikipediasiden og dens undersider: https://en.wikipedia.org/wiki/Internet_protocol_suite