# Teori ITGK-Eksamen

Innhold

[Teori ITGK-Eksamen 1](#_Toc500537666)

[Hardware – Kapittel 1: Definere Informasjonsteknologi 1](#_Toc500537667)

[Hardware – Kapittel 9: Følge instruksjoner – Prinsipper for Datamaskinens Virkemåte 5](#_Toc500537668)

[Digital Representasjon – Del 1 9](#_Toc500537669)

[Digital Representasjon – Del 2 12](#_Toc500537670)

[Nettverk – 1. Introduksjon og oversikt 15](#_Toc500537671)

[Nettverk – 2. Internett-trender 18](#_Toc500537672)

[Nettverk – 8. Pålitelighet og kanalkoding 18](#_Toc500537673)

[Nettverk – 13. LANs, pakker, rammer og topologier 20](#_Toc500537674)

[Nettverk – 20. Konsepter, arkitektur og protokoller 23](#_Toc500537675)

[Nettverk – 21. Adressering på Internett 25](#_Toc500537676)

[Nettverk – 25. TCP: Reliable Transport Service 27](#_Toc500537677)

[Nettverk – 27. Nettverksytelse (QoS og DiffServ) 29](#_Toc500537678)

[Nettverk – 29. Nettverkssikkerhet 31](#_Toc500537679)

[Nettverk – 32. The internet of Things 37](#_Toc500537680)

## Hardware – Kapittel 1: Definere Informasjonsteknologi

**Historie**

* Startet med hullkort
* Første programmerbare digitale datamaskin i 1941 (Konrad Zuse)
* ENIAC, første elektroniske general-purpose datamaskin, finansiert av militære i USA og programmerbar
* Transistoren (1947)
* Integrerte kretser (60-tallet)
* Mikroprosessor (1971🡪)
* ARPANET, datanettverk som senere utviklet seg til Internett. Etablert i 1968. Basert på pakkesvitsjing. Startet av militæret i USA
* 1993
  + WWW (World Wide Web), globalt informasjonsrom som gjør tekst, bilder, multimedia og mange andre typer informasjon tilgjengelige over internett
  + HTML (Hypertext Markup Language), markeringsspråk for formatering av nettsider med hypertekst og annen informasjon som kan vises i en nettleser

**Terminologi**

* Akronymer – første bokstav i hvert ord
  + WYSIWYG (What You See Is What You Get)
  + SOC (System On a Chip)

**Datamaskinen**

* I/O (mus, tastatur, skjerm, kamera …)
* Hovedenhet (CPU, RAM)
* Eksterne enheter (Harddisker, dvd-disker …)

**Hovedkort (Motherboard)**

* Kretskort i kabinettet
  + Inneholder mestepartene av kretsene til PC-systemet
  + Kan ha mange forskjellige utseender
* Mindre kretskort kalt datterkort plugges inn i hovedkortet for å få ekstra funksjonalitet
  + Med raske eksterne porter (f.eks. Thunderbolt-3 eller USB3.1) kan man koble til eksterne kort
* Hovedkortet inneholder mikroprosessor-chipen, også kalt den sentrale prosessor-enheten (CPU), og minnet

**Mikroprosessoren (CPU)**

* Delen av systemet som foretar de faktiske utregningene
  + Ble kalt «mikro» i 1980 for å skille den fra større sammensatte prosessorenheter
    - Central Processing Unit
* Dagens prosessorer er av flerkjerne-typen (Multi Cores)
  + Det kan være vanskelig å bryte programmene opp i deler som kan kjøre samtidig på de forskjellige, slik at regnekraften utnyttes

**Minne (Primær-/Hoved-minne)**

* Stedet hvor programmer og data lagres mens programmet kjører
* Slettes når datamaskinen slås av
* RAM: Random Access Memory
  + Flyktig (ikke permanent)
* En PC inneholder flere millioner/milliarder bytes med RAM
  + Megabytes (MB)/Gigabytes (GB)
* Hva betyr «Random Access»
  + Alle elementer kan hentes direkte
  + Forskjellig fra sekvensiell aksess (f.eks. LP/kassetter/spolebånd)

**Harddisk (Sekundærminne)**

* Høy-kapasitet, persistent perifer lagringsenhet
  + Lagrer programmer og data som ikke er i øyeblikkelig bruk i datamaskinen
  + Laget av magnetiserte jern-legeringer
    - Informasjonen beholdes enten PC-en er på eller av
    - Kalles permanent eller persistent minne
      * Ikke-volatilt (ikke-flyktig, ikke-temporært)
* Liten stabel med blanke metallskiver med en arm som sveiper over/mellom skivene
* 2014: Solid State (Hard) Drives
  + Laget av «Flash Memory»
  + Ingen bevegelige deler – dvs.: Rask
  + Stor «permanent» «RAM»
  + Slites akkurat som magnetiske disker
* Harddisker er også direkte aksess (ikke sekvensielle)

**Lagring fra RAM til Harddisk**

* Lagring (Saving) flytter informasjon fra RAM til harddisk
  + Forsiktige brukere lagrer ofte
  + Moderne OS gjør vanligvis automatisk mellomlagringer
* RAM-minnet er flyktig (volatilt)
  + Informasjon forsvinner når strømmen skrus av
  + Hvis datamaskinene feiler eller restartes vil bare data på harddisk overleve
  + Moderne systemer tar gjerne backup, slik at data overlever krasj

**Integrerte Kretser («Integrated Circuits») (IC)**

* Miniatyrisering:
  + Klokkehastighetene kan være så høye fordi CPU-ene er så små (elektriske signaler kan forflytte seg omtrent 33cm på et nanosekund)
* Ca. 1 milliard transistorer på en overflate på ca. 150 mm2 – 1\*1.5 cm
* Fotolitografi
  + Trykkeprosess (som trykte bøker, side for side)
  + Flere lag oppå hverandre
  + Istedenfor å koble sammen kretser for hånd
  + «Fotografere» de kretsene man ønsker, og etse bort tomrommene imellom «ledningene»
  + Uansett hvor komplisert kablinger er, så er kostnaden og mengden arbeid alltid den samme

**Halvleder-teknologien**

* Integrering
  + Aktive komponenter, og kontaktene som kobler dem sammen, er alle sammen laget av lignende materialer i en enkelt prosess
  + Dette sparer plass og fører til at hele systemet er bare en monolittisk del, som er mer pålitelig enn flere smådeler
* Silisium er en halv-leder – noen ganger leder den strøm, andre ganger ikke
  + Evnen til å kontrollere når en halv-leder leder eller ikke er hoved-redskapen i all datamaskinkonstruksjon

**Silisiums oppførsel**

* En krets er laget for å beregne «x og y» for alle mulige logisk verdi av x og y («True/False»)
* Hvis x er «sann» skal kretsen lede elektrisitet og signalet passerer til den andre enden av ledningen; hvis x er «usann» skal signalet ikke passere
* Det samme gjøres for y-kretsen
* Hvis begge kretsene leder strøm (x og y er sanne), så har vi beregnet logisk «and»

**Felt-effekten**

* Kontrollerer ledeevnen til halvlederen (Silisium)
* Objekter kan bli positivt eller negativt ladet
  + Like ladninger frastøter hverandre, men ulike ladninger tiltrekkes. Denne effekten kalles felt-effekten
* Spalten mellom to ledninger behandles for å øke spaltens ledende/ikke-ledende egenskaper
* Spalten blir da en kanal (en sti hvor elektrisitet kan bevege seg mellom de to kablene)
* En isolator dekker kanalen
* En ledning kalt porten går over isolatoren
* Porten er adskilt fra kanalen av isolatoren – ikke direkte kontakt med verken ledningene eller kanalen
* Elektrisitet ledes ikke mellom de to kablene, bortsett fra gjennom kanalen som bare leder når porten er «ladet»
* Silisiumet i kanalen leder strøm (bare) når den befinner seg i et ladet felt
  + Elektroner blir tiltrukket eller frastøtt i Silisium-materialet
  + Ved å lade porten positivt lages et felt over kanalen slik at strømmen kan gå mellom de to ledningene

**Transistorer**

* Transistor: En kobling mellom to ledninger som kan styres til å la elektriske ladninger flyte, eller ikke, mellom to kabler
  + En bryter uten bevegelige deler
* Dette er beskrivelsen av en MOS-transistor: Metall-Oksyd Halvleder («Semiconductor»)

## Hardware – Kapittel 9: Følge instruksjoner – Prinsipper for Datamaskinens Virkemåte

**Instruksjons-Utførings-Maskin**

* Hva datamaskiner kan
  + Deterministisk utføre (eksekvere) instruksjoner for å behandle informasjon
  + Datamaskinen må ha instruksjoner å følge

**Oppbygningen av en datamaskin**

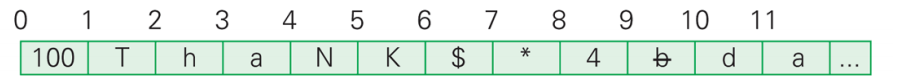
* 5 grunnleggende deler/ subsystemer
  + Minne
  + Kontrollenhet
  + Aritmetisk/logisk enhet (ALU)
  + Input-enheter
  + Output-enheter
* Bundet sammen av
  + Busser

**Minne**

* Minnet (RAM) lagrer programmet som kjører og dataene som programmet behandler
  + Atskilte lagringsplasser/lokasjoner
    - Hver plass består av 1 byte
  + Adresser
    - Hver minnelokasjon (byte) har en adresse (hele tall som starter på null)
  + Verdier
    - Minnelokasjoner tar opp eller lagrer verdier
  + Begrenset kapasitet
    - En gitt størrelse – programmerere må huske at dataene kanskje ikke «får plass» i minnelokasjonen
* 1-bytes minnelokasjoner kan lagre ett ASCII-tegn, eller ett nummer <256 (0-255)
* Bruker ofte en sammenhengende rekke av minnelokasjoner, uten å bry seg om at hver enkeltlokasjon har en egen adresse
  + Word
    - Blokk på fire bytes blir så ofte brukt som en enhet at de kalles for (minne-) ord («Words)
* Forskjellige datatyper tar opp forskjellig plass
  + Char (tar vanligvis opp 1 byte (ASCII) eller 2-4 bytes (UNICODE)
  + Float (8 eller 10 bytes)
  + Etc. …
  + Stort minne fører til mindre I/O

**Byte-store Minnelokasjoner**

* Kan representere de atskilte lokasjonene i minnet ved et boks-diagram (hver boks inneholder en byte)
* Adressen til minnelokasjonen viser over boksen
* Verdien eller innholdet i lokasjonen vises i boksen



**Kontrollenheten (ALU)**

* Maskinvare-implementasjon av **Hente/utføre-kretsen**
  + Henter en instruksjon fra minnet
  + Dekoder instruksjonen
  + Og henter operandene som trengs
* En typisk instruksjon kan være på formen
  + ADD 4000, 2000, 2080 Op Dest, src1, src2
  + Numrene lagret i lokasjon 2000 og 2080 adderes. Resultatet lagres i lokasjon 4000 [4000] = [2000] + [2080]
* «Data/Operand Fetch»-steget må hente disse to verdiene og etter at de er lagt sammen må «Returner Resultat»/lagre-steget sende svaret til minnelokasjon 4000

**Input enhet og Output enhet (I/O)**

* De kablene og kretsene som informasjon flyttes inn og ut av datamaskinen gjennom
* **De ytre enhetene**
  + Kobles til maskinens input/output porter
  + Regnes ikke som en del av maskinen, men som spesialiserte enheter som koder eller dekoder informasjonsflyten mellom datamaskinen og den fysiske verden. (datamus, keyboard, printer, kamera, etc. …)

**De ytre enhetene («Peripherals»)**

* Tastatur
  + Koder tastetrykk til binærform for datamaskinen
* Skjerm
  + Dekoder informasjon fra datamaskinens minne og viser det på en skjerm
* Harddiskene brukes både til input og output
  + Lagringsenheter hvor datamaskinen legger informasjon mens den ikke trengs, og hvor den kan hentes fram igjen når den trengs igjen
* En Driver for hver ytre enhet

**Hente/Utføre – kretsløpet**

* Fem-skritts syklus
  + Instruksjonshenting (IF)
  + Instruksjonsdekoding (ID)
  + Datahenting (DF) / Operandhenting (OF)
  + Instruksjonseksekvering (EX)
  + Resultatretur (RR) / Lagre (ST = Store)

**ALU**

* Utfører regneoperasjonene
* En krets i ALU kan legge sammen to tall
  + Det finnes også kretser for å multiplisere, sammenligne, osv. …
* Gjør vanligvis jobben i instruksjonsutføringssteget (EX) i syklusen, men instruksjoner som bare flytter på data bruker ikke ALU
* Data/Operand Hente-steget (DF) i syklusen henter de verdiene som ALUen trenger å jobbe med (operandene til operatoren)
* Når ALU-en fullfører operasjonen sørger Resultat Returnerings/Lagrings-steget for å flytte svaret fra ALU-n til den minnelokasjonen som er spesifisert i instruksjonen.

**Instruksjonstolkning**

* Programkjøringsprosessen
  + Maskinen tolker våre kommandoer, men i sitt eget språk
* Eksempel fra boka:
  + ADD 800, 428, 884
  + Legg sammen verdiene fra minne 428 og 884, og lagre resultat i minneposisjon 800
  + Før F/E-syklusen begynner, er bare noen minnelokasjoner og PC-en synlige i kontroll-kretsen

**Programtelleren: PC-ens PC (Program Counter)**

* Adressen til den neste instruksjonen lagres i kontroll-delen til maskinen. Den kalles programteller (PC)
* Hvis instruksjoner bruker 4 byte minne, må neste instruksjon være på lokasjon PC + 4, 4 bytes lenger avgårde i sekvensen/programmet (generelt sett)
* Maskinen plusser på fire i PC, så når F/E-syklusen kommer tilbake til Instruksjonshentesteget neste gang, så «peker» PC på den neste instruksjonen

**Forgrening og Hopp-instruksjoner**

* Enkelte instruksjoner kan inkludere en adresse for neste steg. Det vil endre PC-en, slik at istedenfor å gå automatisk til PC+4, så «hopper» den eller «bryter av» («brancher») til den spesifiserte minnelokasjonen for neste instruksjon

**IF -**  Kjøringen starter med å flytte instruksjonen i adressen gitt i PC-en fra minnet til kontrollenheten

**ID -** Dekoder instruksjonen

**DF -** Flytter operandene fra minnet til ALU-en

**EX -** Beregner resultatet i ALU

**RR -**  Returnerer resultatet fra ALU til minnelokasjonen til adressen

**Mange, mange enkle operasjoner**

* Datamaskiner kan bare utføre omkring 100 forskjellige instruksjoner
  + Omtrent 20 forskjellige typer operasjoner (Det trengs forskjellige instruksjoner for å legge sammen enkelt-bytes, «ord», desimaltall, osv. …)
* Alt datamaskinene gjør må reduseres til en kombinasjon av disse primitive, hardkodete instruksjonene

**Eksempler på andre instruksjoner**

* Foruten ADD, MULT og DIV, så finnes også følgende andre instruksjoner
  + Flytt bit-ene i et ord til venstre eller høyre (shift), fyll de tomme bitsene med 0 og kast vekk de som detter utenfor enden
  + Beregn logisk AND (test om par av bit-er begge er sann), og logisk OR (test om minst en bit i paret er sann)
  + Test om en bit er null eller ikke-null, og hopp til et nytt sett instruksjoner avhengig av utfallet
  + Flytt informasjon i minnet
  + Sjekk signaler fra input/output-enhetene

**Drive F/E-syklusen**

* Datamaskiner får sine imponerende evner ved å kjøre mange av disse enkle instruksjonene hvert sekund
* Maskin-klokka: Bestemmer hastigheten på F/E-syklusen
  + Måles i gigahertz (GHz), antall milliarder sykluser per sekund

**Hvordan forbedre klokke-hastigheten?**

* Prøver å starte en instruksjon på hvert klokke-tikk
* Overlater avslutning av instruksjonen til andre kretser (dette kalles «pipelining», som er en slags parallell-utføring. Forskjellig fra flerkjerne parallell-prosessering)
  + Fem instruksjoner kan behandles samtidig
* Kan en datamaskin med 1 GHz klokke virkelig utføre en milliard instruksjoner per sekund?
  + Ikke helt presist mål. Datamaskinen klarer ikke alltid å starte en ny instruksjon på hvert tikk, men klarer noen ganger å starte mer enn en instruksjon samtidig

**Programvare (Software)**

* En maskins syn på programvare
  + Maskinen ser binære objekt-filer, som en lang sekvens av 4-bytes «ord» (0’ere og 1’ere)
* Assemblerspråk
  + En alternativ form for maskinspråk som bruker bokstaver og normale tall, slik at mennesker kan forstå det
  + Maskinen leser assemblerkode, og etter hvert som den finner ord slår den opp i en tabell for å konvertere ordene til binærkoder, tall konverteres til binær form, og så settes de binære kodene sammen (assembles) til en instruksjon
  + De forskjellige prosessorene har forskjellig assemblerspråk
* Høynivå programmeringsspråk
  + De fleste moderne program er skrevet med høynivå notasjon, som deretter kompileres (oversettes) til assemblerspråk, som igjen assembles til binærkode
  + Har spesielle uttrykksformer som hjelper programmerere å gi kompliserte instruksjoner
    - Eksempel: Tre-delt «if-statement» (hvis)
      * Ja/Nei spørsmål som skal testes
      * Instruksjon skal utføres hvis testen er true
      * Instruksjon skal utføres hvis testen er false

**Operativsystem (OS)**

* Grunnleggende operasjoner som er nødvendig for å kunne bruke maskinen effektiv, men som ikke er bygd inn i maskinvaren
* De tre mest brukte operativsystemene:
  + Microsoft Windows
  + Apple’s Mac OS X
  + Unix/Linux
  + Mobiltelefonoperativsystemer øker (Android, iOS …)
* OS-et tar seg av «booting» (oppstart), minnehåndtering, enhetshåndtering, internettforbindelse, filhåndtering, etc. …

**Programmering**

* Programmerere bygger på tidligere utviklet programvare for å gjøre jobbene sine lettere
* Eksempel: GUI Programvare
  + Ramme rundt vindu, skyveknapper, knapper, pekere, osv. er ferdig pakket inn i OS-et og tilgjengelig for programmerere

## Digital Representasjon – Del 1

**Representasjon av informasjon**

* Bits: 0/1
* Bytes: 8 bits, 28 = 256 symboler (mønster)
  + 0000 0000, 0000 0001, 0000 0010, …
* Digitalisering: Representere informasjon ved symboler
  + Diskret versus analog (kontinuerlig), distinkt, skillbart
* P and A-koding (Present and Absent)
  + Presence/absence av fysisk fenomen
    - Strøm, spenning, magnetisme, lys/ikke-lys
* Datamaskinens minne
  + Sekvens av bits gruppert i bytes
  + Hver byte, unik adresse

**Data + tolkning = informasjon**

* Åtte byte (64 bits), for eksempel:
  + 1011 1101 0011 1001 1111 1011 1000 1001
* Hva kan dette representere?
  + 8 tegn (bokstaver, tall, etc. …)
  + Et flyttall
  + 4 (16 bits) heltall (med størrelse fra 0 til 65536)
  + 64 logiske verdier (0/1, usant/sant)
  + Et binært tall med 64 binære siffer
* Metadata er data om data, det vi trenger for å tolke riktig

**Representasjon av tegn**

* ASCII / ISO-8859-1 / Latin-1
  + En byte for hvert tegn
  + 256 forskjellige tegn
* Håndtere flere tegn?
  + Bruke flere byte per tegn
  + Unicode-standarden
    - Standard tegnsett som støtter alle språk som er i praktisk bruk
* Tegn kan brukes til å representere tall (som tekst)

**Representasjon av heltall**

* Fast størrelse i antall byte
* Int8 (1 byte)
  + 28 = 256 symboler
  + Bare positive tall (unsigned): 0 – 255
  + Positive og negative (signed): -128 – 127
* Int32 (4 byte)
  + 232 = 4 294 967 296 symboler
  + Bare positive tall: 0 – (232-1)
  + Positive og negative tall: -2 147 483 648 – 2 147 483 647
* Kan få overflyt
  + Int8: 255 + 1 kan ikke representeres

**Positive og negative tall: Fortegnsbit**

* Fortegnsbit: Første bit er fortegn:
  + 0 = positivt
  + 1 = negativt
* Case: 3 bits heltall (8 symboler)
* Enkleste representasjon vist til høyre
* Dårlig løsning:
  + To nuller
  + Tungvint for ALU i prosessoren

**Positive og negative tall: 2-komplement**

* Fra + til -, og – til +, med samme oppskrift
  + Ta positiv koding
  + Bytt alle 0 og 1
  + Legg til 1
* Bonus: Subtraksjon som addisjon
* Forenkler datamaskinens ALU

**Representasjon av reelle tall**

* Tallene representeres i scientific notation
  + 1234,56 = 1,23456 \* 103
  + 0,0011 = 1,1 \* 10-3
  + Normalisert med ett siffer foran komma
* Kan representere flyttall med:
  + Mantisse (frictional part) inkludert fortegnsbit (som heltall)
  + Eksponent
* Komma alltid etter første siffer i mantissen (floating point)
* Eksempel:
  + 1234,56 = 1,23456 \* 103 som representeres som:
    - Fortegn: 0, Mantisse: 123456, Eksponent: 3

**IEEE standard for floating-point numbers**

* Single precision
  + 32 bits (4 byte)
  + 24 bits til fortegn + mantisse
  + 8 bits til eksponenten
  + Absoluttverdier omtrent 1,2 \* 10-38 – 3,4 \* 1038
  + Omtrent 7 signifikante desimale siffer
* Double precision
  + 64 bits (8 byte)
  + 53 bits til fortegn + mantissen
  + 11 bits til eksponenten
  + Absoluttverdier: 2,2 \* 10-308 – 1,8 \* 10308
  + Nøyaktighet: Omtrent 16 desimale siffer

## Digital Representasjon – Del 2

**Digitalisering av lyd**

* Et objekt lager lyd ved å vibrere et medium
* Kraften eller intensiteten av bølgene avgjør volumet
* Frekvensene (antall bølger per sek) avgjør tonehøyden (pitch)

**Analog til digital**

* «Sample» eller ta målinger på bestemte intervall
* Antall samples per sekund kalles samplingsfrekvens (rate)
* Jo raskere frekvens, desto mer nøyaktig opptak av lyden

**Nyquist-regelen** **for sampling**

* Hvis samplinga er for treg kan lydbølger plasseres mellom samplene og vi vil miste viktige segmenter av lyden.
* Nyquist-regelen sier at samplingsfrekvensen må være minst dobbelt så rask som den raskeste frekvensen
* Ettersom menneskelige ører kan høre lyder opp til ca. 20 000 Hz, vil samplingsfrekvens på 40 000 Hz oppfylle Nyquists regel for digitalt lydopptak
* Av tekniske grunner ble samplingsfrekvensen for CD valgt til å være 44,100 Hz (digital lyd)

**Hvor mange bits per sample?**

* Vi kan kun få omtrentlige målinger av lyd
* Hvis en ekstra bit vil bli brukt, vil lydsamplet bli dobbelt så nøyaktig
* Flere bits oppløsning gir mer nøyaktig digitalisering
* Digital CD lyd representeres ved 16 bit, dvs. 65536 forskjellige lydnivåer

**Fordeler med digital lyd**

* En viktig fordel med digital informasjon er muligheten til å kjøre databehandling på representasjonen
* En viktig databehandling er å komprimere digital lyd, eller redusere antall bit som trengs
* Hva med lyder som det menneskelige øre ikke kan høre fordi de er for høye eller lave?
* MP3 er representasjon av digital lyd som er databehandlet for å ta mindre plass (fjerner unødvendig informasjon)
  + Gir komprimering på mer enn 10:1 (mindre filer)
* Digital data kan reproduseres nøyaktig
  + Kan dette være et problem?

**Digitale bilder og video**

* Et bilde er en lang sekvens av RGB piksler
* Bildet er todimensjonalt, men tenk pikslene strekt ut rad etter rad i minne
* Eksempel:
  + 8 x 10 bilde skannet til 300 piksler per tomme (dpi)
  + Det er 80 tommer2, hver trenger 300 x 300 = 90 000 piksler (7,2 megapiksler)
  + Representerer farger med 3 bytes per piksel, tar 21.6 MB (3\*7.2) i minnet til lagring av 8 x 10 fargebilde på 300 dpi (Dots per pixels)
  + Sende dette bildet over en 56 Kb/s linje vil ta minst 21 600 000 x 8/56 000 = 3 085 sekunder (mer enn 51 minutter)

**RGB – Rød, grønn, blå**

* 0-255 enheter av hver farge
* 256\*256\*256 = 16.777.216 farger
* FFFF00 = Gul, 00FFFF = Turkis, FF00FF = Lilla, 000000 = Svart og FFFFFF = Hvit

**Bildestørrelse**

* Størrelse (dimention):
  + Bredde (cm/tomme) x høyde (cm/tomme)
* Oppløsning (resolution):
  + Antall piksler per cm/tomme (dpi = dot pr inch)
* Fargedybde
  + Antall fargealternativer for hver piksel
    - 1 bit: 2 alternativ
    - 8 bit: 28 = 256
    - 24 bit: 256\*256\*256 = 224 = 16,7 millioner
* På skjerm er oppløsning irrelevant
  + Bredde (antall piksler)
  + Høyde (antall piksler)
  + Fargedybde (antall byte per piksel)
* Filstørrelse = Bredde\*Høyde\*Fargedybde
  + 600\*400\*1 byte = 240.000 bytes

**Filformater**

* Regler for koding av bilder
* Fargedybde
  + 24 bit/indekserte farger
* Komprimering
  + Dataene kodes slik at de tar mindre plass
    - Komprimering: Bilde -> Kodet fil
    - Dekomprimering: Kodet fil -> Bilde
  + Tapsløs / tapskoding
    - Tapsløs: Halvering – tredeling
    - Tapskoding: Større effekt, avhengig av tap
  + TIFF, PSD, PNG, GIF, JPEG/JPG

**GIF**

* 8 bit fargetabell (=256 farger)
* Alfa-kanal for transparens
* Tapsløs komprimering
* Bra for «data grafikk»
* Mindre bra for fotografier

**JPEG**

* 24 bits
* Komprimering med tap, styrbar
* Bra for fotografier (glatter skarpe kanter + artifakter)
* Ikke bra for «datagrafikk»
* Org 560 KB
  + 14 Kb og 162 KB

**Bildekomprimering**

* Komprimering betyr å endre representasjon slik at færre bit trengs til å lagre eller overføre informasjon
  + Eks: Fakser en sekvens av 0’ere og 1’ere som koder en side av hvitt (0) og sort (1)
  + Bruker run-length-koding for å spesifisere hvor lang første sekvens av 0’ere er, så hvor lang neste sekvens av 1’ere er, osv. …
* Run-length-koding er tapsløs (lossless) komprimering
  + Den originale representasjon av 0’ere og 1’ere kan bli rekonstruert perfekt fra den komprimerte versjonen
* Motsatte er lossy-(taps)-komprimering

**Komprimering**

* MP3 er sannsynligvis den mest kjente typen komprimering
  + MP3 er lossy ettersom de høye notene ikke kan gjenskapes
* JPG (eller JPEG) er lossy komprimering av bilder
  + Utnytter karakteristikker av menneskelige oppfatningsevne for å lage forenklinger iht. lyst og farge som ikke er så synlig
* Mennesker er temmelig sensitive til små endringer i lysstyrke (luminance)
* Lysstyrkenivåer i et bilde må bevares mellom ukomprimert og komprimerte versjoner
* Mennesker er ikke sensitive til små forskjeller i farger (chrominance)

**JPEG – komprimering**

* JPEG er i stand til 10:1 komprimering uten synlig tap av klarhet i bilde ved å gjøre områdene der man komprimerer små

**Bits: Universalmedium**

* 4 bytes kan representere ulike typer informasjon
* Fundamental egenskap av informasjon:
  + Bias-Free Universal Medium Principle:
  + Bits kan representere all diskre informasjon
* Bits har i utgangspunktet ingen iboende mening
* All diskre informasjon kan representeres ved bits
* Diskrete ting – ting som kan separeres fra hverandre
  + Kan representeres ved hjelp av bits

## Nettverk – 1. Introduksjon og oversikt

**Fem sentrale aspekter**

* Nettverksapplikasjoner og programmering
* Datakommunikasjon
* Pakkesvitsjing og nettverksteknologier
* Internett og TCP/IP
* Andre nettverkskonsepter og teknologier

**Nettverksapplikasjoner og programmering**

* Et underliggende nettverk som støtter alle applikasjoner gjør det enklere for programmereren
  + Ett grensesnitt
  + Ett sett med funksjoner

**Datakommunikasjon**

* Refererer til lavnivås mekanismer og teknologier som brukes for å sende informasjon over et kommunikasjonsmedium (Kabel, radiobølger, lys)
* Verden er i utgangspunktet «analog», i den forstand at de menneskelige sanser (f.eks. syn, hørsel) fungerer på denne måten
  + Så hvorfor gå over til digitale nettverk?

**Analog/Digital – Hvorfor digitalisere?**

* Integrasjon – Ønske om å kombinere tjenester
* Kapasitet – Enklere å utnytte kapasitet bedre med felles nett
* Kvalitet – Digital koding kan gi bedre feilsjekk og korrigering
* Terminaler (f.eks. PCer) er digitale

**Pakkesvitsjing og nettverksteknologier**

* Telekommunikasjon i gamle dager:
  + Fysiske kabler måtte sammenkobles for at to parter skulle kunne kommunisere
  + Dedikerte linjer (linjesvitsjing)

**Offentlige- og private nettverk**

* Industrien skiller gjerne mellom offentlige og private nettverk
* **Offentlige nettverk**: Nettjeneste som er **tilgjengelige for abonnenter**
* Tilbys av en leverandør (service provider), f.eks. Telenor, mot betaling
* **Private nettverk**: Kontrolleres av en **bestemt gruppe**. Bruk av nettverket er begrenset til denne
* Kontroll betyr ikke nødvendigvis eierskap

**Private nettverk: Fire kategorier**

* Consumer
* Small Office/Home Office (SOHO)
* Small-to-Medium Business (SMB)
* Large Enterprise

**Nettverk, interoperabilitet og standarder**

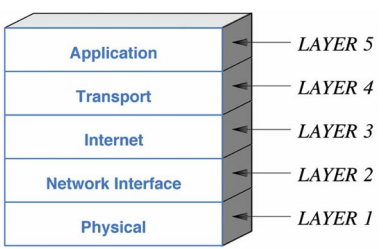
* Alle entiteter i et nettverk må være enige om hvordan informasjon skal representeres og kommuniseres
* Hvordan kommunikasjonen skal foregå er spesifisert i en protokoll
* En protokoll beskriver prosedyrer for
  + Feilhåndtering
  + Uventede hendelser (f.eks. ingen respons fra mottaker)

**Protokoller: Familier og (lag)modeller**

* En protokoll bør kun håndtere deler av kommunikasjonen som ikke håndteres av andre protokoller
* Dette oppnås gjennom å:
  + Lage protokoller som komplette samarbeidende sett (engelsk: suites)
  + La hver protokoll i en suite (familie) håndtere ett aspekt av kommunikasjonen
  + Designe en familie med protokoller slik at de kan arbeide sammen på en effektiv måte

**Nettverk, interoperabilitet og standarder**

* Protokoller i en familie er organisert i lag
* Den lagbaserte modellen (layering model) gjør det lettere å håndtere kompleksiteten i nettverkskommunikasjon



**TCP/IP stabelen (stack)**

* Applikasjon
  + WWW, Snapchat, YouTube
* Transport
  + TCP: ntnu.no, google.no
* Internett
  + IP: 129.241.103.4
* Nettverksgrensesnitt
  + MAC AB:CD:EF:12:34:56
* Fysisk lag
  + Spenning i kobber, radiofrekvens i luft, lys i rør, …

**Hvordan data går gjennom lagene**

* Output fra en protokoll i et lag er input til en protokoll i neste lag
* Tilgrensende protokoller sender pekere til pakker seg i mellom -> økt effektivitet

**Hoder (headers) og lag**

* Hvert lag i protokoll-stacken bidrar til at en melding kommer frem som forventet
* Ettersom datapakker blir sendt gjennom lagene i protokoll-stacken legges det til hoder (ekstra informasjon)
* Hodene leses og fjernes av tilsvarende lag på mottakersiden (og underveis!)

**OSI-Referansemodellen**

* OSI (Open Systems Interconnection) er en ISO-standard for lagdeling av kommunikasjonsprotokoller
* Har to lag i tillegg til de vi finner i TCP/IP-stacken
* Presentasjon (LAYER 7)
* Sesjon (LAYER 6)

**Oppsummering kapittel 1**

* Internett består av mange teknologier og produkter
* For at ulike nettverksnoder skal kunne snakke sammen trengs nettverksprotokoller som spesifiserer detaljer for hvordan kommunikasjonen skal foregå
* For at protokoller skal kunne virke godt sammen lages de gjerne i sett (suites) hvor hvert lag har ansvar for et kommunikasjonsaspekt
* Lagmodellen bidrar til å redusere kompleksiteten i nettverkskommunikasjon
* TCP/IP protokollene som brukes i internett har fem lag

## Nettverk – 2. Internett-trender

**Internett-trender**

|  |  |
| --- | --- |
| Før | Nå |
| Ressursdeling | Kommunikasjon |
| Tekst | Multimedia |
| Analog telefon | Voice over IP (VoIP), 4G |
| Analog TV | IP-basert TV |
| Kablet aksess | Trådløs aksess (Wi-Fi) |
| Sentraliserte tjenester | Distribuerte tjenester |
| Personlig datamaskin | Skybaserte tjenester |

**Nye applikasjoner**

* Sosiale medier
* Sensornettverk
* Høykvalitets telekonferansesystemer
* Bank og betalingsapplikasjoner

**Fra personlig datamaskin til Cloud Computing**

* En tjenestetilbyder (Cloud provider) tilbyr nettbasert tjenester som f.eks. lagring i et Cloud data center
* En bruker eller en bedrift kan abonnere på tjenesten og dermed få tilgang til f.eks. datafiler uavhengig av aksessmedium (f.eks. PC, tablet, smarttelefon)
* Forenkler fildeling mellom ulike brukere

## Nettverk – 8. Pålitelighet og kanalkoding

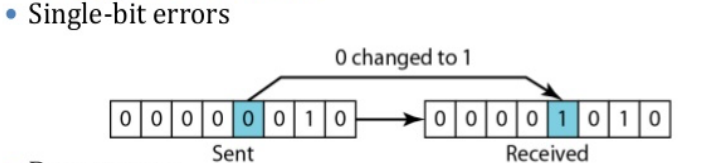
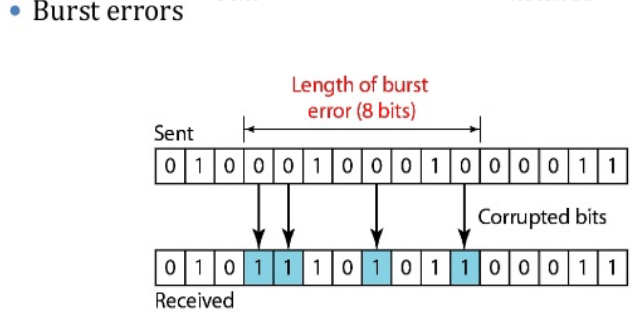
**Feilkilder i overføring av data**

* Feil i dataoverføringer er uunngåelig
* Feildetekterings-mekanismer er ressurskrevende
* Kostnad kontra konsekvens av feil
  + Mht. f.eks. pengetransaksjoner vil feildetektering gis prioritet
  + Mht. f.eks. bildeoverføring vil feildetektering gis mindre prioritet
* *Interference* (forstyrrelse):
  + Elektromagnetisk stråling kan forstyrre radiosignaler
* *Distortion* (forvrenging):
  + Signaler kan forvrenges f.eks. av metallobjekter
* *Attenuation* (Svekking):
  + Signaler kan svekkes når de sendes over et medium

**Effekt av feil i overføring av data**

* Single Bit Error
  + En enkelt bit i en blokk med bits endres, men øvrig bits forblir uendret
* Burst Error
  + Flere bits i en blokk endres
* Erasure (ambiguity)
  + Signalet som kommer frem hos mottaker er tvetydig (verken logisk 1 eller logisk 0) pga. f.eks. forvrenging

**Burst Error vs. Single Bit Error**

**Strategier for håndtering av feil: Kanalkoding**

* Forward Error Correction (FEC)
  + Legger til tilleggsinformasjon som lar mottaker verifisere at korrekt data har blitt mottatt og rette feil hvis mulig
* Automatic Repeat reQuest (ARQ)
  + Forutsetter samarbeid med avsender. Avsender og mottaker utveksler informasjon for å sikre at data blir overført korrekt

**FEC-korrigeringsteknikker**

* Block Error Codes:
  + Dataene som sendes deles inn i blokker, med ekstra informasjon (redundancy) tilknyttet til hver blokk
  + Convolutional Error Codes

**Eksempel på Block Error Code: Single Parity Checking (SPC)**

* Sender legger til en ekstra (paritets-) bit (0 eller 1) til hver byte
* Mottaker fjerner paritetsbit og sjekker om bits i mottatt byte er korrekt
* Avsender og mottaker må enes om partalls-paritet eller oddetalls-paritet
* **Partalls-paritet:** Paritetsbit settes til 0 dersom blokken har et likt antall 1-bits og 1 dersom blokken har oddetall
* For **Oddetalls-paritet** er det motsatt
* SPC sjekker om en av de ni overførte bits (inkludert paritetsbits) har endret seg for å avgjøre om mottatt byte er korrekt
* **SPC kan kun avdekke feil i tilfeller hvor en odd antall bits har endret seg**
* SPC avdekker ikke hvilken bit som er feil (**kan ikke korrigere**)

**Row and Column (RAC) parity**

* **RAC kan avdekke og korrigere (single-bit) feil**
* Mottaker organiserer mottatt data som en tabell med paritetsbits for hver rad og kolonne
* Bitfeil forårsaker feil i to paritetssjekker, som igjen identifiserer hvor i tabellen feilen ligger
* RAC kan korrigere single-bit feil
* **RAC kan avdekke multi-bit feil i tilfeller hvor en odd antall bits er endret**

**Internett bruker 16-bit sjekksum**

* Deler data inn i 16-bits enheter. Legger til 0’er til slutt for å fylle opp siste enhet ved behov. RAC kan korrigere sin

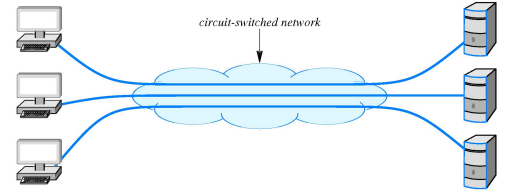
**Oppsummering**

* Forstyrrelser, forvrenging og svekkelse av signal kan forårsake feil i dataoverføring
* Feil kan medføre single-bit errors, burst errors, eller erasures
* Forward error correction og Automatic repeat request er teknikker som har hensikt å avdekke og rette feil

## Nettverk – 13. LANs, pakker, rammer og topologier

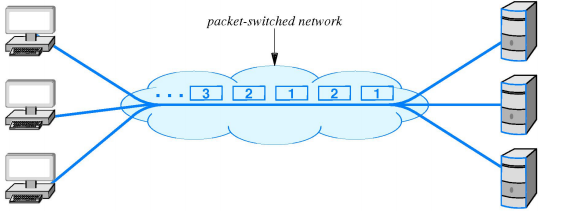
**Linjesvitsjing (circuit switching)**

* Mekanismer som oppretter en dedikert kommunikasjonskanal mellom sender og mottaker
* Brukt i f.eks. analoge telefonsystemer
* Ikke betinget av fysisk forbindelse
* Egenskaper som betegner linjesvitsjing:
  + Kommunikasjon mellom to endepunkter
  + Permanente kanaler kontra midlertidige (svitsjede)
  + Kommunikasjon mellom to parter i linjesvitsjede nettverk påvirkes ikke av kommunikasjon mellom andre
* Linjesvitsjing gir en illusjon av en isolert fysisk kobling mellom to kommunikasjonsparter. Koblingen opprettes ved behov og avsluttes etter endt bruk



**Pakkesvitsjing**

* Kommunikasjonskanalene (mediene) er delt mellom ulike parter
* Hver melding deles inn i mindre blokker med data. Disse kalles pakker eller datagram
* Betegnelsen ramme (frame) brukes for å spesifisere pakkeformatet i et gitt nettverk
  + Hode (metainformasjon) + dataene i meldingen
* Størrelsen på pakkene avhenger av svitsje-teknologi (typisk 1500 bytes)

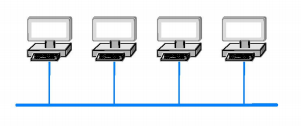
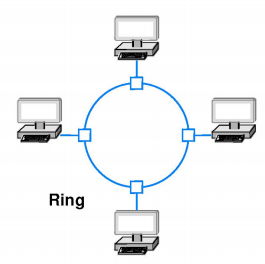
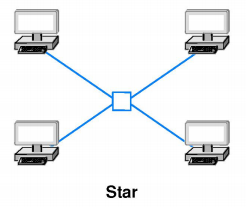
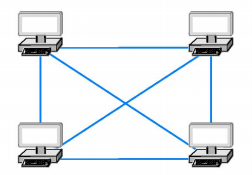


* Egenskaper som betegner pakkesvitsjing:
  + Sender kan kommunisere med 1 … n mottakere samtidig. Mottaker kan motta meldinger fra 1 … n sendere samtidig
  + Asynkron kommunikasjon. Kommunikasjon mellom sender og mottaker kan skje når som helst. Forsinkelser håndteres.
  + Ingen krav til å sette opp kommunikasjonslinjene: En pakke kan sendes når som helst til hvilken som helst destinasjon
  + Nettverksytelse avhenger av hvor mange sendere som prøver å sende pakker samtidig. Kapasiteten fordeles ikke etter et fast mønster, men etter behov (statistisk multipleksing)
* **Pakkesvitsjing gir mer effektiv utnyttelse av nettverksressurser**

**LAN, MAN og WAN**

* Pakkesvitsjede nettverk karakteriseres gjerne etter områdene de dekker:
  + LAN (Local Area Network) dekker typisk et rom eller bygning
  + MAN (Metropolitian Area Network) dekker typisk en stor by
  + WAN (Wide Area Network) dekker typisk flere byer
* Jo større dekningsområde, desto høyere kostnad

**Lan topologier**

* **Buss-topologi**
* Alle datamaskiner i nettverket er koblet til et felles medium (kabel)
* Hver datamaskin kan sende data til hvilken som helst annen i samme LAN
* Kun en datamaskin kan sende signaler i gangen (krever koordinering)
* Krever mindre kabling, men ett felles medium gjør nettverket sårbart
* **Ring-topologi**
* Lukket sløyfe. Pakken sendes gjennom ringen til den når riktig mottaker
* Enklere å koordinere aksess og detektere nettverksproblemer
* Sårbart dersom man mister en forbindelse
* **Stjerne-topologi**
* Datamaskinene er koblet sammen i sentralt punkt (hub eller switch)
* Hub’en tar imot data fra sendere og ruter den videre til riktig mottaker
* Pålitelig. Om en node slutter å virke påvirker ikke dette resten av nettverket
* **Mesh-topologi**
* Direkte kobling mellom hver datamaskin i nettverket
* Høy kostnad. Antall forbindelser i et nettverk med n datamaskiner: (n2-n)/2
* Pålitelig. Om en node slutter å virke påvirker ikke dette resten av nettverket

**Adressering i LANs**

* Hver datamaskin i et pakkesvitsjet LAN har en unik adresse: Media Access Control (MAC) adresse.
  + Kalles også fysisk adresse, hardware adresse
* Hver pakke inneholder MAC-adressen til avsender-node og mottaker-node
* Lag 2 i TCP/IP-stacken (Network Interface Layer) legger til MAC-adresseinformasjonen på pakken

**Hva MAC-adressen inneholder**

* En MAC-adresse består av en 48 bits binær verdi delt inn i to 3 bytes blokker
  + Den første blokken (OUI) spesifiserer utstyrsleverandøren, som får disse tildelt
  + Den andre blokken (NIC) spesifiserer en unik verdi utstyrsleverandøren gir den spesifikke maskinvaren
* Bit nr. 7 i den første blokken spesifiserer om OUI er globalt unik eller lokalt tilordnet (for eksperiment arbeid)
* Bit nr. 8 i den første blokken spesifiserer henholdsvis om adressen er unicast (0) eller multicast (1)

**Unicast, Broadcast, Multicast**

* Tre typer adressering som korresponderer til tre typer pakkelevering:
  + Unicast: Identifiserer en enkelt datamaskin i nettverket. Bare denne skal motta pakken
  + Broadcast: Adresserer alle datamaskiner i nettverket. Alle mottar kopi av pakken
  + Multicast: Identifiserer et subsett av datamaskiner i nettverket. Alle i subsettet mottar kopi av pakken
* Broadcast kan betraktes som en form for multicast, dvs. et sett som inneholder alle datamaskiner i nettverket

**Oppsummering kapittel 13**

* Pakkesvitsjing deler meldinger som skal sendes over et nettverk inn i mindre pakker. Dette bidrar til bedre utnyttelse av nettverksressurser
* Nettverk som buker pakkesvitsjing klassifiseres gjerne etter dekningsområde: LAN, WAN, MAN
* Fire typiske LAN-topologier: Buss, stjerne, ring og mesh
* Hver datamaskin i et pakkesvitsjet LAN har en unik 48-bits adresse: MAC adresse. Denne består av to deler: OUI og NIC
* Unicast, broadcast og multicast er tre typer adressering som brukes i forbindelse med pakkelevering

## Nettverk – 20. Konsepter, arkitektur og protokoller

**Universelle tjenester**

* Ulike nettverksteknologier dekker ulike behov
  + F.eks. LANs kontra WANs
* Universal service: Tillate kommunikasjon mellom datamaskiner uavhengig av type nettverk de sitter på
* Internett er en sammenkobling av ulike fysiske nettverk. Universal service er et grunnleggende premiss for at internett fungerer

**Rutere**

* Maskinvare med prosessor, minne og I/O-grensesnitt som brukes for å koble sammen heterogene nettverk
* Kan koble sammen ulike typer nettverk (f.eks. to LANs, et LAN og et WAN, eller to WANs) og ulike nettverksteknologier (medium, adresseringssystem, rammeformat)
* Bruker protokoller for å støtte kommunikasjon mellom ulike typer nettverk

**Internett-arkitektur**

* En ruter kan brukes til å koble sammen flere nettverk
* Det er knyttet ulemper til å bruke bare en ruter for å koble sammen mange nettverk:
  + Forutsetter en kostbar ruter. Ofte mer gunstig med flere små som oppgraderes uavhengig og etter behov
  + Mindre pålitelighet. Kommunikasjon mellom nettverk blir umulig dersom ruteren krasjer
* Anskaffelse av rutere i organisasjoner styres gjerne av krav til pålitelighet, kapasitet og kostnad

**Virtuelle nettverk**

* Internett-programvare skaper en illusjon av et enkeltstående, sømløst kommunikasjonssystem bestående av mange datamaskiner
* Internett er et eksempel på et (verdensomspennende) virtuelt nettverk
  + De underliggende kommunikasjonsmekanismene er abstrahert bort
  + Brukere av internett trenger i mindre grad å forholde seg til dem
* Illusjonen muliggjøres av TCP/IP

**TCP/IP**

* TCP/IP brukes av både verter (hosts) og rutere
* Lag 3 (IP) spesifiserer:
  + Formatet på pakkene
  + Mekanismer for å sende pakker fra en datamaskin via rutere, til en annen
* Lag 4 (TCP) sørger for pålitelig overføring av data

**Oppsummering kapittel 20**

* Universal service: Tillate kommunikasjon mellom datamaskiner uavhengig av type nettverk de sitter på
* En ruter er maskinvare som brukes for å koble sammen heterogene kommunikasjonsnettverk
* TCP/IP har en sentral rolle mht. universal service.
  + Er med på å skape en illusjon av et enkeltstående, sømløst kommunikasjonssystem bestående av mange datamaskiner
  + Benyttes av både verter (hosts) og rutere

## Nettverk – 21. Adressering på Internett

**IP-adressering**

* Internett er en abstraksjon; en illusjon av ett enkelt, stort kommunikasjonsnettverk
  + Protokoll-programvare har en sentral rolle i å skape denne illusjonen
  + Designere kan velge adresser, pakkeformater, og bestemme hvordan data skal sendes uavhengig av den underliggende maskinvaren
* For at abstraksjonen skal fungere må alle datamaskiner på internett ha: Felles adresseringsformat
* Unik adresse (MAC-adressen er generelt kun tilgjengelige innenfor samme lokalnett)
* IP-adressering sørger for at en kan sende data til alle maskiner på internett
  + Sender merker pakken med IP-adressen til mottaker (i tillegg til egen IP-adresse)
  + IP-protokoll-programvare sørger for at pakken når frem
* En IPv4-adresse består av 32 bits (25) og har et *prefiks* og et *suffiks* som til sammen gir en unik adresse
* *IP-prefikset* identifiserer det unike fysiske nettverket (nettverksnummeret) en datamaskin er tilknyttet
* *IP-suffikset* identifiserer en spesifikk datamaskin på nettverket

**Den originale IP-adresseringsmåten (Classful IP addressing)**

* Dilemma: Mange nettverk med begrenset størrelse (flere prefiks-bits) eller færre nettverk med stor størrelse (færre prefiks-bits)
* For å løse dilemmaet ble IPv4-adresseområdet delt inn i tre primærklasser (A, B og C). Med ulike størrelser på prefiks og suffiks.
* De første fire bits i en adresse spesifiserer klassen og prefiks/suffiks-inndelingen

**IPv4 Dotted Decimal Notation**

* Dotted Decimal Notation er en uttrykksform som gjør det lettere å lese, skrive og snakke om IP-adresser
* Hver 8-bits seksjon av adressen uttrykkes som et desimaltall og separeres med punktum
* Minste verdi i hver 8-bits seksjon er 0 (åtte 0’er); høyeste verdi er 255 (åtte 1’ere)

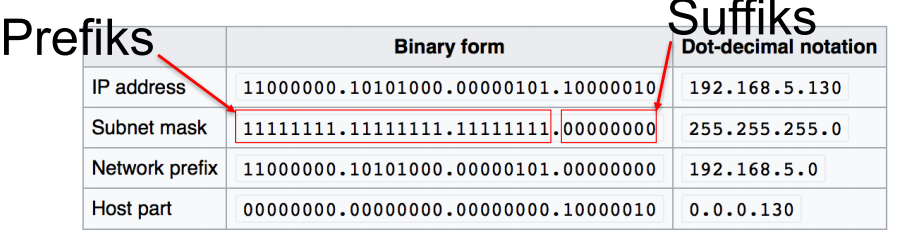


**IPv4 Subnetting og klasseløs adressering**

* Classful IP addressing utnyttet ikke adresseplassen godt nok. Behov for finere prefiks-/suffiksinndeling
* Subnetting og klasseløs inndeling ble lagd for å overkomme begrensinger mht. IPv4s klasseinndeling.
  + Tillater vilkårlig deling mellom prefiks og suffiks (unngår å kaste bort adresseplass)
  + En tjenestetilbyder på Internett kan dele ut prefiks med passelig størrelse

**Subnett-masker (adresse-masker)**

* En subnett-maske deler en IP-adresse inn i nettverks- (prefiks) og host-adressen (suffiks). Verter og rutere må lagre subnett-masken
* En subnett-maske er en 32-bits verdi hvor
  + 1 spesifiserer hvilke bits av IP-adressen som utgjør prefiks
  + 0 spesifiserer hvilke bits av IP-adressen som utgjør suffiks
* Ved å bruke en bit-for-bit logisk AND-operasjon på IP-adressen og subnett-masken finner protokollen ut hva som er nettverks- og host-adressen



**Classless Inter-Domain Routing (CIDR)-notasjon og IPv4**

* Utvidelse av dotted decimal notation
* En adresse og en maske kan spesifiseres ved hjelp av en adresse og en / fulgt av et nummer
* Nummeret (m) spesifiserer antall venstrejusterte 1-bits i masken
  + ddd.ddd.ddd.ddd/m
  + Eksmpel: 192.5.48.69/26 spesifiserer en maske på 26 bits

**Fra IPv4 til IPv6**

* Internett er i en transisjon fra IP versjon 4 til IP versjon 6
* Størrelsen på Internett mer en fordobles hvert år
* Behovet for IPv6 er knyttet til at veksten av internett har skapt mangel på mulige IP-adresser
  + Fra 32 bits (IPv4) til 128 bits (IPv6) (27)
* Høy kostnad
* Endring av IP medfører en endring av hele internett

**IPv6-adressering**

* IPv6 benytter et tre-nivås hierarki med mht. til adresser:
  + Globalt (unikt) prefiks (tilordnet en organisasjon). Størrelse på IPv6 prefiks kan variere (tilsvarende som IPv4)
  + Subnett (nettverk). Størrelse på subnett og prefiks er 64 bits
  + Vert (en bestemt datamaskin på nettverket). Fast størrelse på 64 bits
* Gir mulighet for adresser som bare kan brukes innenfor et bestemt nettverk og som er begrenset til en organisasjon
* Har ikke en spesiell adresse for broadcast – benytter multicast
* Hver IPv6 adresse har en av tre basistyper
  + Unicast: Korresponderer til en bestemt datamaskin. Pakker som sendes til adressen rutes langs den korteste veien
  + Multicast: Korresponderer til et sett med datamaskiner. En kopi av hver pakke sendes til alle i settet. Settet kan endres
  + Anycast: Korresponderer til et sett med datamaskiner med felles prefiks. Pakker sendes til en av disse datamaskinene

**IPv6 Colon Hexadecimal Notation**

* Forenkler måten vi skriver IPv6-adresser på
* Følgende 128-bits adresse skrevet med dotted decimal notation:
  + 105.220.136.100.255.255.255.255.0.0.18.128.140.10.255.255
* Forenkles ved å gjøre om hver gruppe av 16 bits til heksadesimalverdier (med : som skilletegn)
  + 69DC:8864:FFFF:FFFF:0:1280:8C0A:FFFF
* Zero compression reduserer kompleksiteten ytterligere, ved å erstatte serier med 0 med to :’er. For eksempel kan adressen:
  + FF0C:0:0:0:0:0:0:B1 forenkles til FF0C::B1

**Oppsummering**

* For å skape en illusjon av et sammenhengende, sømløst nettverk bruker internett en uniform adressering: IP-adresser
* IPv4 bruker 32-bits adresser som deles inn i prefiks (nettverksnummer) og suffiks (host)
  + Classful IP addressing: Antall bits som definerer prefikset er forhåndssatt (avhengig av klassen)
  + Ved bruk av klasseløsinndeling og subnett-masker kan antall bits som definerer prefiks være vilkårlig. Gir bedre utnyttelse av adresserommet.
* Transisjon fra IPv4 til IPv6
* IPv6 bruker 128-bits adresser som deler IP-adressen inn i prefiks, subnett og host

## Nettverk – 25. TCP: Reliable Transport Service

**Tjenester TCP tilbyr**

* Connection orientation
  + Først etablere forbindelsen, deretter bruke den til å sende data
* Punkt-til-punkt-kommunikasjon
  + Hver forbindelse har nøyaktig to endepunkter
* Pålitelighet
  + TCP sørger for at all data blir levert slik den ble sendt; komplett og i riktig rekkefølge
* Full dupleks kommunikasjon
  + Tillater at data kan sendes begge veier, når som helst
* Stream interface
  + Muligheten for å sende en kontinuerlig strøm med enkelt-tegn (tilfeldig pakking)
* Pålitelig opprettelse av forbindelse
  + Sørger for at to applikasjoner kan begynne å kommunisere på en pålitelig måte
* Avslutning av forbindelse
  + Sørger for at all data har blitt levert og at begge sider er enig i at forbindelsen kan avsluttes

**«Ende-til-ende»-tjenester og virtuelle forbindelser**

* TCP tilbyr virtuelle ende-til-ende forbindelser (opprettes av programvare)
* TCP bruker IP for å sende meldinger
* Hvert endepunkt trenger TCP-programvare; ikke rutere

**Håndtering av duplikater og pakker i feil rekkefølge (Sequencing)**

* Brukes for å håndtere duplikater og pakker som kommer i feil rekkefølge
* Avsender merker hver pakke med et sekvensnummer
* Mottaker lagrer sekvensnummer på siste mottatt pakke i tillegg til en liste med andre pakker som kom i feil rekkefølge
  + Hvis forventet pakke -> lever til neste lag
  + Hvis uforventet pakke -> legg pakken i ventelisten
  + Mottaker forkaster nye pakker som er mottatt fra før, eller som ligger i ventelisten

**Håndtering av pakketap (retransmisjon)**

* Når en pakke ankommer intakt sender mottaker en bekreftelse (ACK)
* Hver gang avsender sender en pakke starter den en timer
* Dersom bekreftelse fra mottaker ankommer før timeren utløper kanselleres timeren
* Dersom timeren utløper før bekreftelse blir mottatt sendes en kopi av den (tapte?) pakken (retransmisjon) og en ny timer startes
* Retransmisjoner kan føre til duplikater
* Begrensinger på antall retransmisjoner

**Håndtering av forsinkelser (replay errors)**

* Replay error: Forsinket pakke fra tidligere sesjon blir akseptert i senere sesjon. Riktig pakke blir avvist som duplikat
* Kan oppstå i forbindelse med svært lange forbindelser
* Kan også oppstå i forbindelse med kontrollpakker (etablering/avslutning av kommunikasjon)
* Håndteres gjennom å merke pakker i en sesjon med en unik ID (f.eks. sesjonenes starttidspunkt)
  + Protokoll-software avviser pakker med feil ID

**Flytkontroll**

* Teknikker som forhindrer at en rask avsender overkjører en sen mottaker
  + Stop-and-og:
    - Sender venter på bekreftelse fra mottaker om mottatt pakke før ny pakke sendes
    - Lite effektivt
  + Sliding window:
    - Variasjon av stop-and-og med en bestemt buffer
    - Neste slide

**Håndtering av pakkeopphopning (congestion)**

* På 1GHz nett kan man sende 1GHz fra en maskin på Switch1 til en på Switch2
  + Men ikke fra to maskiner på 1 til to på 2 samtidig … Da dropper mer enn 50% av pakker
* Løsning: Reduser senderaten/vinduet (raskt)

**Oppsummering kapittel 25**

* TCP
  + TCP er den mest brukte transport-protokollen i TCP/IP-stacken
    - Pålitelig
    - Flyt-kontroll
    - Full-Duplex
    - Stream
  + Pålitelig dataoverføring ved å kompensere for
    - Duplikater og pakker som kommer i feil rekkefølge
    - Pakketap
    - Forsinkelser
    - Overkjøring («Data overrun»)
    - Pakkeopphopning

## Nettverk – 27. Nettverksytelse (QoS og DiffServ)

**Kvantitative mål på nettverksytelse**

* Latency (delay)
  + Tiden det tar å overføre data over et nettverk fra et endepunkt til et annet
* Throughput (capacity, aka bandwidth)
  + Mengden data som kan overføres innenfor en gitt tidsenhet
* Jitter (variability)
  + Endringer i forsinkelser og lengden på forsinkelsene

**Ulike typer latens (forsinkelser)**

* Propagation delay:
  + Tiden det tar for et signal å sendes over et medium (f.eks. kabel)
* Access delay:
  + Tiden det tar å få aksess til et delt medium (f.eks. en kabel i et nettverk)
* Switching delay:
  + Tiden det tar å rute en pakke videre (som følge av f.eks. tabelloppslag)
* Queuing delay:
  + Tiden en pakke er i minnet til en switch/ruter mens den venter på å bli videresendt
* Server delay:
  + Tiden det tar for en server å respondere på en forespørsel og sende respons

**Capacity, Throughput og Goodput**

* Kapasiteten til et nettverk uttrykkes gjerne som maks Throughput
* Throughput betegner hvor mye data som kan sendes gjennom et nettverk målt i bits per sekund (bps)
* Goodput er mengden data selve applikasjonen kan sende (uten overhead)

**Throughput og forsinkelser**

* Buss vs. Racerbil
* Hjelper det å lage flere kjørebaner på motorveien?
  + Mer trafikk?
  + Raskere?

**Jitter**

* Jitter beskriver variansen i forsinkelser
* Oppstår dersom pakker bruker ulik tid på å bli overført fra sender til mottaker

**Quality of Service (QoS)**

* Noen vil betale sin ISP mer for å få garantier om hastighet, båndbredde, etc. …
  + Service Level Agreement (SLA) sier noe om vanlig og garantert overføringshastighet (Committed Information Rate)
  + Committed Information Rate (CIR) kan ofte være 0 selv om vanlig hastighet er 155 Mbps
  + Brukeren som aksepterer CIR = 0 må være forberedt på ubestemt nede-tid ved hendelser som at kabelen ryker, grave-uhell, strømbrudd, etc. …

**Oppsummering**

* Forsinkelse og Throughput (Hvordan måle fart/hastighet)
* Forsinkelse øker drastisk når throughput nærmer seg 100% av kapasiteten til nettverket
* Jitter
* Balanse mellom rettferdighet og beregnings-kostnader

## Nettverk – 29. Nettverkssikkerhet

**Sikkerhetstrusler**

* Med økt digitalisering og tilgjengelighet av informasjon og tjenester på nett endres også trusselbildet
* For mange organisasjoner og bedrifter er nettverkssikkerhet viktig for å unngå bl.a.:
  + Tap av omdømme
  + Tap av tillit blant f.eks. brukere eller kunder
  + Tap av åndsverk (intellectual property)

**Sikkerhetsutfordringer**

* **Phising**: Opptre som en kjent nettside (f.eks. nettbank) for å få tak i personlig informasjon som f.eks. aksesskoder, kontonummer, etc.
* **Misrepresentation**: Oppgi feilaktige opplysninger om et produkt eller tjeneste eller levere produkter tjenester som er falsk eller av dårlig kvalitet
* **Scams**: Lure brukere av f.eks. et nettsted eller tjeneste til å investerer penger eller gjøre noe ulovlig
* **Denial of service**:Bevist blokkering av tilgang til et nettsted eller tjeneste
* **Loss of control:** Uvedkommende tar kontroll over en brukers datamaskin
* **Loss of data:** Tap av åndsverk eller annen verdifull informasjon

**Sikkerhetsangrep**

* **Wiretapping** (avlytting): Kopiere datapakker som traverserer nettet for å få tak i informasjon
* **Replay:** Sende pakker fanget opp fra tidligere sesjoner (f.eks. passord-pakker fra tidligere pålogginger)
* **Buffer-overflow:** Oversende mer data enn hva mottaker forventer. Overskrider en databuffers grenser og skriver til nabolokasjoner i minnet. Kan føre til minneaksessproblemer, feil resultater og krasj.
* **Spoofing:** Bruke falsk IP-kildeadresse for å lure mottaker til å prosessere pakken. Bruke falske e-postavsendere, misbruke domenenavn. Avlede datatrafikk rettet mot en server
* **Denial of Service (DoS/DDoS):** Overøse en vert med datapakker (fra en eller flere kilder) for å bruke opp alle ressurser. Skaper lange forsinkelser
* **SYN flood**: Form for DoS-angrep. Angriperen sender en serie med SYN-forespørsler (synkroniseringsforespørsel om å opprette forbindelse) mot et datasystem for å forhindre annen trafikk)
* **Password breaking**: Automatiserte systemer laget for å «knekke» passord og dekrypteringsnøkler for å uautorisert aksess til en nettressurs
* **Port Scanning:** Forsøk på å koble til åpne protokollporter på en vert for å finne en svakhet (porter brukes for å identifisere prosesser, eller type nettverkstjeneste)
* **Packet Interception:** Modifisering av pakker på vei fra avsender til mottaker

**Sikkerhetsretningslinjer (policy)**

* Det finnes ingen absolutt definisjon på et sikkert nettverk
* I praksis må organisasjoner balansere sikkerhet og enkelhet i bruk
* En organisasjons nettverkssikkerhets-policy beskriver *hva* som bør beskyttes (ikke hvordan)
  + Komplekst. Involverer en kombinasjon av mennesker, datamaskiner og nettverk)
* Typiske sikkerhetsaspekter en organisasjon må vurdere?
  + **Dataintegritet:** Er dataene som sendes identiske med de som mottas?
  + **Datatilgjengelighet:** Er dataene tilgjengelige for personer som skal ha tilgang?
  + **Datakonfidensialitet:** Er dataene beskyttet mot uautorisert tilgang?
  + **Personvern:** I hvilken grad blir anonymitet ivaretatt? Avsløres senderens identitet?

**Ansvar og kontroll**

* En organisasjon må også spesifisere hvordan ansvar for informasjon tildeles og kontrolleres:
  + Ansvar: Hvilken gruppe er ansvarlig for hvilke data? Hvordan loggføre av gruppen aksess og endringer?
  + Autorisasjon: Hvem har ansvar for hvor informasjon ligger? Hvordan tillater ansvarlig person aksess og endringer
* Kontroll er sentralt mht. til både ansvar og autorisasjon
  + En organisasjon må kunne kontrollere tilgang til informasjon
  + Autentisering (validering av identitet) er en viktig del av kontroll
  + En autentiserings-policy forutsetter autentiserings-mekanismer
  + Eksempel: Rollebasert tilgang til informasjon forutsetter mekanismer som muliggjør å skille mellom brukerroller som f.eks. student, studentassistent og faglærer

**Sikkerhetsteknologier**

Ulike sikkerhetsteknologier tjener ulike formål:

* **Hasing:** Dataintegritet
* **Aksesskontroll og passord**: Datakonfidensialitet
* **Kryptering:** Personvern
* **Digitale signaturer:** Meldingsautentisering
* **Digitale sertifikater:** Avsenderautentisering
* **Brannmurer (firewalls):** Nettsteds-integritet
* **Inntrengingsdeteksjonssystem:** Nettsteds-integritet
* **Innholdsskanning og dyp pakkeinspeksjon:** Nettsteds-integritet
* **Virtuelle private nettverk (VPNs)**: Datakonfidensialitet

**Hashing**

* Paritetssjekker sørger ikke for dataintegritet
  + Feil kan resultere i endret sjekksum og endret dataverdi
  + Kan få gyldig sjekksum for data som har blitt endret
  + En angriper kan endre data i pakker og likevel sørge for at sjekksum er gyldig
* Hashing-mekanismer kan lage meldings-autentiseringskoder som ikke kan knekkes eller forfalskes
  + I noen hashing-mekanismer benyttes en hemmelig nøkkel som bare avsender og mottaker har
  + Nøkkelen brukes til å generere en hash-kode (kort bitsekvens) som avsender merker meldingen med
  + Mottaker bruker nøkkel for å undersøke om han/hun får generert samme hash-kode
  + En angriper kan ikke modifisere pakken uten å introdusere feil. En gyldig hash-kode betyr mao, at pakken er autentisk

**Aksesskontroll og passord**

* Aksesskontroll-mekanismer kontrollerer hvilke brukere eller programmer som kan aksessere hvilke data
  + Passord som styrer hvilke data en bruker får tilgang til
  + Aksesskontroll-lister (ACLs) per objekt som spesifiserer hvem som skal ha tilgang
* Tiltak må gjøres for å sikre at passord ikke er for enkle å gjette (krav til lengde, tegninnhold, etc.)
* Ukrypterte passord og ACLs som sendes over nettverk kan potensielt bli plukket opp av noen som «avlytter» nettverket

**Kryptering**

* Å kryptere en melding vil si å endre på dataene i en melding, slik at kun riktig mottaker kan rekonstruere den opprinnelige meldingen.
  + Plaintext: Originalmeldingen før kryptering
  + Cyphertext: Den krypterte meldingen
  + Encryption key: Bitsekvens som brukes til kryptering
  + Decryption key: Bitsekvens som brukes til dekryptering
* Kan garantere konfidensialitet, meldingsautentisitet og dataintegritet

**Kryptering (som funksjon)**

* Vi kan tenke på kryptering som en return-funksjon som tar i mot to argumenter (nøkkelen K1 og meldingen M), og som returnerer en kryptert variant av meldingen C:
  + C = encrypt(K1, M)
* En dekrypteringsfunksjon reproduserer originalmeldingen:
  + M = decrypt(K2, C)

**Kryptering med privat nøkkel**

* Krypteringsteknikker kategoriseres etter hvordan de bruker krypteringsnøkler:
  + Private key systems (systemer med privat nøkkel)
  + Public key systems (systemer med offentlig nøkkel)
* Kryptering med privat nøkkel
  + Partene deler en hemmelig nøkkel som brukes både for kryptering og dekryptering
  + Partene kan både sende og motta krypterte meldinger

**Kryptering med offentlig nøkkel**

* Hver part får en (hemmelig) privat og offentlig (distribuerbar) nøkkel
* En melding kryptert med en offentlig nøkkel kan kun dekrypteres med den korresponderende private nøkkelen
* Sikrer konfidensialitet, dvs., at kun tiltenkt mottaker får tilgang til originalmeldingen (man trenger den private nøkkelen for å dekryptere meldingen)

**Digitale signaturer (autentisering)**

* En kan også bruke en privat nøkkel for å kryptere en melding. Dette kalles å gi en melding en digital signatur
* En mottaker kan autentisere avsender (den digitale signaturen) ved å bruke avsenders offentlige nøkkel for å dekryptere meldingen
* Mottaker vet med sikkerhet hvem som sendte meldingen fordi avsenders offentlige nøkkel kan kun dekryptere en melding kryptert av den korresponderende private nøkkelen
* En melding kan krypteres to ganger for å garantere både autentisitet (riktig avsender) og konfidensialitet (riktig mottaker)
* Først dekrypteres (signeres) meldingen ved at avsender bruker sin private nøkkel
* Deretter krypterer avsender meldingen en gang til ved å bruke mottakers offentlige nøkkel
* For å dekryptere mottatt melding, må mottaker første bruke sin private nøkkel (riktig mottaker), deretter avsenders offentlige nøkkel (riktig avsender)

**Digitale sertifikater og utstedere**

* For å forhindre at hvem som helst kan lage private og offentlige nøkler, og dermed opptre som andre, er det behov for en pålitelig tredjepart – en nøkkel/sertifikatutsteder
* Et digitalt sertifikat garanterer at identiteten er knyttet til en offentlig nøkkel
* Ved å ha tilgang utsteders offentlige nøkkel, kan man få tilgang til andre offentlige nøkler

**Brannmurer**

* Overvåker og kontrollerer trafikk inn og ut av et internt nettverk for å beskytte mot problemer som kan komme utenfra
* Typisk innebygd i switcher og rutere
* Sentraliserer kontroll over nettverkstrafikken som går inn og ut
* Implementerer sikkerhetsretningslinjene til en organisasjon og stopper trafikk (pakker) som ikke er i tråd med disse

**Brannmurer og pakkefiltrering**

* En brannmur har et pakkefilter som undersøker informasjonen i pakkehodet og avgjør om pakken skal få passere gjennom ruteren eller ikke
* Kan brukes for å kontrollere aksess til spesifikke tjenester på spesifikke datamaskiner

**Inntrengings-deteksjonssystem (IDS)**

* Mekanisme som monitorer pakker og gir administrator beskjed når sikkerhetsbrudd avdekkes (en brannmur kan kun stoppe enkeltpakker som utgjør en potensiell trussel)
* Konfigureres typisk til å se etter spesielle typer angrep
* Kan kobles sammen med brannmur slik at nye konfigurasjoner/regler kan opprettes på bakgrunn av hva som detekteres
* I motsetning til en brannmur kan et IDS loggføre hvor pakker kommer fra, f.eks. om flere SYN-pakker kommer fra samme kilde

**Innholdsskanning og dyp pakkeinspeksjon**

* En brannmur sjekker kun pakkehodet – ikke pakkeinnholdet
* Innholdsanalyse kan sjekke pakkeinnhold for skadelig programvare, som f.eks. virus
* Fil skanning: Skanner hele filer på jakt etter «fingeravtrykk» (kjente byte-mønstre), som kan indikere virus
* Dyp pakkeinspeksjon (DPI): Analyserer innhold i pakker (i tillegg til hodet)
* DPI er ressurskrevende

**VPN**

* Bruker kryptering for å tilby sikker aksess til et lokalt nettverk for klienter som befinner seg utenfor (f.eks. NTNUs nettverk hjemmefra)
* Opprinnelig laget for å koble sammen lokale nettverk i en organisasjon
* Sikker løsning for lav kostnad
* VPN-funksjonen kan knyttes til dedikerte rutere med brannmur for økt sikkerhet

**Pakkekryptering kontra tunnelering**

* Hovedsakelig tre valgmuligheter mht. hvordan pakker som skal sendes over internett kan krypteres: Payload encryption, IP-in-IP tunneling og IP-in-TCP tunneling
* Payload encryption:
  + Krypterer kun meldingsinnholdet i pakken (ikke pakkehodet)
  + Kilde- og destinasjonsadresse, pakkestørrelse, osv. kan fanges av uvedkommende
* IP-in-IP tunnelling:
  + Krypterer både innhold og hodet, og plasserer resultatet i en ny pakke, som så kan sendes over internett
* IP-in-TCP tunnelling:
  + To parter etablerer TCP forbindelse. Forbindelsen brukes til å sende en strøm av krypterte datapakker
  + Små hoder legges til for å markere skillet mellom datapakkene i strømmen
  + VPN programvare på mottakersiden leser hodet og antall spesifiserte bytes for å finne datapakken
  + Når en komplett kryptert melding er mottatt dekrypteres denne
  + Pakkene må leveres i riktig rekkefølge (kan skape forsinkelser)

**Ytelsesutfordringer tilknyttet VPN**

* Latens/forsinkelser
  + Bruk av VPS resulterer gjerne i at datapakker må traversere internett mange ganger mellom en bruker og det lokale nettverket brukeren er koblet opp til
  + Medfører gjerne økt tidsforbruk for å gjennomføre en transaksjon
* Throughput:
  + Raten data kan sendes på gjennom internett kan være en begrensing
  + Kan medføre at en får gjort færre transaksjoner per tidsenhet
* Overhead/fragmentering
  + Tunnelering gjør at det blir mer data å sende og motta (f.eks. krypterte pakker som puttes i nye pakker)
  + Dersom ny pakke overstiger grensen for hvor mye data som kan kommuniseres per transaksjon (MTU) må pakken deles opp videre
  + Øker sannsynligheten for forsinkelser og tap av pakker

**Oppsummering kapittel 29**

* Kommunikasjonsnettverk er utsatt for flere typer trusler
  + Phising, misrepresentation, scams, DoS, etc.
* Det finnes mange forskjellige typer sikkerhetsangrep
  + Avlytting, spoofing, buffer overflow, denial of service, knekking av passord og nøkler, pakkemanipulering, etc.
* Sikkerhet er en relativ term
* En sikkerhetspolicy spesifiserer en organisasjons dataintegritet, -tilgjengelighet, -konfidensialitet og personvern
* Ulike teknologier håndterer ulike sikkerhetsaspekter
  + Kryptering, hashing, digitale signaturer og sertifikater, brannmurer, IDS, pakkeinspeksjon, VPN, etc.

## Nettverk – 32. The internet of Things

**Integrerte systemer (embedded systems)**

* Kommunikasjon mellom «ting» vi omgir oss med i hverdagen, som f.eks. lysbrytere, alarmsystemer, termostater, etc.
  + Automatisering
  + Maskin-til-maskin-kommunikasjon (M2M)
* Noen sentrale områder for M2M
  + Smart grid (strømstyring)
    - Eksempel: Lading av el-utstyr når strømmen er billigst
  + Online sikkerhetssystemer
    - Automatiske varsler med ulike hendelser (dør, vindu, bevegelsessensorer)
  + Handel
    - Skreddersydde reklamer og tilbud
    - Fange opp kundes oppførsel

**IOT og valg av nettverksteknologi**

* Ulike nettverksteknologier gir ulike fordeler og ulemper mht. til IOT. Typiske hensyn:
  + Trafikkvolum: Dataorientere- kontra kontrollorientere applikasjoner
  + Mobilitet: Kabla kontra trådløst
  + Energiforbruk: Batteritid kontra strømkostnad
* Energy Harvesting: Nyttiggjøre energi fra omgivelsene for å lade batterier
  + F.eks. bruke bevegelsesenergi for å lade et armbåndsur
* Trådløs teknologi med lavt energiforbruk er viktig for mange IOT-applikasjoner
  + Lavt energiforbruk, begrenser hvor langt radiosignalene rekker, gjør kommunikasjonen mer sensitiv mht. forstyrrelser og øker sannsynligheten for pakketap
* Mesh-nettverk muliggjør trådløse kommunikasjonsnettverk bestående av noder med lavt energiforbruk
* Hver node opprettholder en liste over hvilke nabo-noder som er innen radiorekkevidde
* Nodene samarbeider om å få frem en melding sendt fra en node (A) til en node som ligger utenfor radiorekkevidde (C)

**Oppsummering**

* IOT brukes for å beskrive sammenkoblete integrerte systemer hvor datamaskiner kommuniserer med andre datamaskiner
* Valg av nettverksteknologi i M2M styres gjerne av krav til trafikkvolum, mobilitet, og energiforbruk
* Lavt energiforbruk er en viktig faktor i mange IOT-applikasjoner