UiA

Group number and student names:

gruppe 22

Aleksander Kolsrud

Emmanuel Medard

Erik Aasen Eskedal

Kevinas Maksevicus

Kristian Skibrek

Tobias Vetrhus

Course Code:	***IS-105***
Course Name:	***Datakommunikasjon og operativsystem***
Responsible teacher/professor:	***Janis Gailis***
Deadline / Due date:	
Number for pages included in the deliverable:	*10*
Title:	*modul 4*

We confirm hereby that we are not citing or using in other ways other's work without stating that clearly and that all of the sources that we have used are listed in the references	Yes	No
We allow the use of this work for educational purposes	Yes	No
We hereby confirm that every member of the group named on this page has contributed to this deliverable/product	Yes	No

IS 105 modul 4

innhold:

Introduksjon

Denne teksten tar for seg oppgavene tildelt av faglærer. Oppgavene hører til boka *Computer Networking 6th edition*. Programmene beskrevet i teksten ligger lastet opp på https://github.com/TheSkibb/Gruppe-22-IS-105

Kapittel 1

R1

What is the difference between a host and an end system? List several different types of end systems. Is a Web server an end system?

Et 'end system' er for eksempel laptoper, smarttelefoner, tablets, TVer, spillkonsoller, kameraer, sensorer, og sikkerhetssystemer tilkoblet internett, gjennom kommunikasjonslinjer og pakkesvitsjing som grupperer data (Kurose, James, Ross, Keith 2013, s. 2-3) . End system er enheter koblet til internett som lar oss interagere med brukergrensesnittet på ulike måter, både motta og sende data. Det kan være å surfe på nettet, laste ned eller opp filer, sende bilder eller sjekke e-posten (McMahon, 2021).

Et 'host system' er end-systemer som kjører applikasjonsprogrammer, som for eksempel nettlesere. En Web Server er et end-system, men blir referert som 'host' siden det kjører et nettleserprogram. Hosts blir ofte delt inn i *klienter* og *tjenere*, hvor klientene ofte er datamaskiner og mobiltelefoner, mens tjenerne, eller serverene, samler og distribuerer nettsider, e-poster, videoer, med mer gjennom bruk av store kraftige maskiner (Kurose, et. al., 2013, s. 10).

R2

The word protocol is often used to describe diplomatic relations. How does Wikipedia describe diplomatic protocol?

Wikipedia definerer "Diplomatic protocol" slik:

> ***Protocol*** *is commonly described as a set of international courtesy rules. These well-established and time-honored rules have made it easier for nations and people to live and work together. Part of protocol has always been the acknowledgment of the hierarchical standing of all present. Protocol rules are based on the principles of civility.—Dr. P.M. Forni on behalf of the International Association of Protocol Consultants and Officers* ("Protocol (diplomacy)", 2021).

Generelt forklarer Wikipedia det som diplomati i internasjonal politikk og affære, samt at det dreier seg om hvordan en spesifikk aktivitet skal utføres, særlig når det gjelder diplomati. De kan også være uskrevne regler, normer, særlig på statlige felt. Respekt og passende oppførsel er også nevnt ("Protocol (diplomacy)", 2021).

R3

Why are standards important for protocols?

Standarder er viktig for protokoller slik at det er et system i hvordan ting gjennomføres. For eksempel internett protokoller, som TCP og IP er avgjørende for riktig sending og mottagelse av informasjon gjennom internett (Kurose, et. al., 2013, s. 5). Det er viktig at alle er enige om hva hver protokoll gjør, slik at folk kan skape systemer og produkter hvor alle forstår hva som skjer, og lett kan samarbeide med hverandre. For eksempel ved programmeringsspråk er det viktig at vi har standarder i hvordan koden skal skrives og utformes. UTF-8 er også et eksempel på et standardisert universelt binært språk, som er felles for alle. Hvis vi skulle hatt hundrevis av slike språk, ville ikke internett fungert på en global skala.

Det er utviklet internett standarder av IETF, for å løse protokoll design og problemer (Kurose, et. al., 2013, s. 5). Generelt sett er standardiserte protokoller viktig for at vi skal kunne samhandle og forstå hverandre på mest effektiv måte, om det skulle gjelde internett eller hverdagslige ting som oppførsel, eller mer diplomatisk oppførsel i det statlige.

Tilbake til nettverksprotokoller er det avgjørende at alle komponenter kommuniserer; dette styres av protokoller, for eksempel bit-flow på kablene (bussen) mellom to nettverkskort (Kurose, et. al., 2013, s. 9)

R4

List six access technologies. Classify each one as home access, enterprise access, or wide-area wireless access.

Access network er det som fysisk kobler et end-system til en ruter, altså veien fra et end-system til et annet

Seks access-teknologier:

Home access:

- Digital subscriber line (DSL): Gir internett-tilgang til boligstrøk.
- Satellitt: Kan brukes til å gi boligstrøk internett-tilgang når DSL, kabel eller FTTH ikke er tilgjengelig.

Enterprise/Home access:

- LAN, hovedsakelig Ethernet i bedriftssammenheng.
- Wireless LAN / WiFi

Wide-area wireless access

- Pakkesvitsjet wide-area trådløst nettverk (3G/4G/5G)
- LTE teknologier

(Kurose, et. al., 2013, s. 12 - 18)

R7

What is the transmission rate of Ethernet LANs?

Overføringshastigheten for Ethernet-LAN er for brukere vanligvis mellom 10 Mbps og 100 Mbps, men tjenere kan ha mellom 1 Gbps og 10 Gbps (Kurose, et. al., 2013, s. 17).

R8

What are some of the physical media that Ethernet can run over?

Fysiske medier er gjerne et par av transmitter-recievere, som for eksempel et par kobber-ledninger, en coax-kabel, radio- eller satellitt spektrum, og så videre. Disse blir delt inn i *guided media*, som gjerne er fysiske kabler, og *unguided media*, som er bølger i atmosfæren, som for eksempel trådløst nettverk, og radiobølger (Kurose, et. al., 2013, s. 19). Noen av de fysiske mediene som Ethernet kan 'kjøre over' er de såkalte guided media ledningene, som kobber-ledninger, coax og fiber-optics.

R10

Describe the most popular wireless Internet access technologies today. Compare and contrast them.

De to mest populære wireless Internet access teknologiene I dag er:

1. Wireless LAN
2. Wide-area wireless access network.

Wireless LAN bruker radiooverføring for å sende data. Brukere kan veksle data mellom hverandre med rundt 10-20 meters avstand. LAN connectionen er koblet direkte til nettet, som vil si at brukerne blir direkte koblet til internett.

Det blir brukt hjemme, på jobb, i skoler og universiteter. Et eksempel på Wireless LAN er Wi-Fi

Wide-area wireless access network kan bli forklart som Mobile Broadband. Den bruker mobile telecommunication cellular network teknologi som 2G, 3G, 4G og 5G for å sende data. Den kan sende og motta data innen en radius på flere titalls kilometer fra hovedstasjonen/serveren. Stort sett alle mobiler bruker denne teknologien med 3G og 4G.

R11

Suppose there is exactly one packet switch between a sending host and a receiving host. The transmission rates between the sending host and the switch and between the switch and the receiving host are R1 and R2, respec- tively. Assuming that the switch uses store-and-forward packet switching, what is the total end-to-end delay to send a packet of length L? (Ignore queuing, propagation delay, and processing delay.)

R1 = Overføringshastighet/mengde mellom senderen og switchen.

R2 = Overføringshastighet/mengde mellom switchen og receiveren.

L= Packet of length

End-to-end delay vil se slik ut: L= L/R1+L/R2 fordi du må vurdere hvor lang tid det tar på begge delene. Her må du dele lengden(hvor mange bits) L med R(overførings mengde/hastighet) for å vite hvor lang tid det tar for biten å bli sendt mellom sender og R1, og så R2 til receiver. (geeks for geeks, 2021)

R13(r)

Suppose users share a 2 Mbps link. Also suppose each user transmits continu- ously at 1 Mbps when transmitting, but each user transmits only 20 percent of the time. (See the discussion of statistical multiplexing in Section 1.3.)

a. **When circuit switching isused, how many users can be supported?**
   Når circuit switching blir brukt, så kan det være maks 2 personer fordi hver person bruker 1Mbps, når linken er 2Mbps. Med circuit switching må man reservere en plass for hver bruker på en link, som vil si at etter 2 brukere vil linkens reservasjonspris bli brukt opp.
b. **For the remainder of this problem, suppose packet switching is used.Why will there be essentially no queuing delay before the link if two or fewer users transmit at the same time? Why will there be a queuing delay if three users transmit at the same time?**
   Siden det maksimale er 2Mbps, vil det å ha to eller færre brukere ikke gå over grensen, som vil si at det vil ikke være noe queuing delay. Hvis en tredje person hopper inn vil det oppstå queuing delay fordi det tilsvarer 3 Mbps totalt.
c. **Find the probability that a given user is transmitting.**
   Sjansen for at en bruker bruker linken er 0.2 (20/100).
d. **Suppose now there are three users. Find the probability that at any given time, all three users are transmitting simultaneously. Find the fraction of time during which the queue grows.**
   Hvis det er tre brukere av linken så er sjansen for at alle 3 er på samtidig = 0.2^3 =

0.008. Fraction of time during which queue grows er lik sjansen for at alle tre brukerne er på samtidig, som er 0.008.

R15

Some content providers have created their own networks. Describe Google's network. What motivates content providers to create these networks?

Google sitt nettverk gir hvem som helst informasjon om nesten hva som helst, bare man skriver inn et spørsmål eller nøkkelord i søkemotoren. De bruker det som kalles Edge Network for å samhandle med ISP-er. Gjennom ISP-en(e) frakter de informasjon mellom serverne sine og brukerne. Google er ikke en bruker av bedrifter som kontrollerer ISP-er, de er på et måte sitt eget nettverk, eller er en del av internett. Så informasjonen blir sendt til googles Edge Network som videresender informasjonen til google sitt datasenter.

Det som gir motivasjon til bedrifter eller «content providers» til å lage egne nettverk er at de selv har kontroll over sine tjenester ved å begrense bruken av bare et par ISP-er, og at de kan spare penger ved å sende mindre trafikk inn i ISP-er. (Google, n.d)

R16

Consider sending a packet from a source host to a destination host over a fixed route. List the delay components in the end-to-end delay. Which of these delays are constant and which are variable?

Når man sender en pakke fra en møter er flere delaykomponenter underveis. Kurose, James, Ross og Keith forklarer dem som *"nodal processing delay, queuing delay and transmission delay"*. Av disse er queuing delay den eneste som er variabel. Dette er fordi det er den eneste som vil være forskjellig for forskjellige pakker som sendes gjennom samme path (Kurose, et al., 2013, s. 36).

R19(r)

Suppose Host A wants to send a large file to Host B. The path from Host A to Host B has three links, of rates R1 = 500 kbps, R2 = 2 Mbps, and R3 = 1 Mbps.

a. **Assuming no other traffic in the network, what is the throughput for the file transfer?**
   Throughput for en path med tre linker R1, R2 og R3, hvor R1 = 500kbps, R2 = 2Mbps og R3 = 1Mbps, vil bli min{ 500kbps, 2Mbps, 1Mbps} = 500kbps.
b. **Suppose the file is 4 million bytes. Dividing the file size by the throughput, roughly how long will it take to transfer the file to Host B?**
   Vi kan finne filstørrelsen i bits ved å gange 4 millioner med 8, da får vi at filen er 32 000 000 bits stor. 500kbps er det samme som 500 000 bps (bits/sekund). Det vil da ta 32 000 000 / 500 000 = 64 sekunder å sende filen til host B
c. Repeat (a) and (b), but now with R2 reduced to 100 kbps.
   Om R2 = 100kbps, så vil throughput bli min{500kbps, 100kbps, 1Mbps} = 100 kbps. Det vil nå ta 32 000 000 / 100 000 = 320 sekunder å sende filen til host B.

R22

List five tasks that a layer can perform. Is it possible that one (or more) of these tasks could be performed by two (or more) layers?

Ved å for eksempel se på OSI modellen (Open System Interconnect), kan vi finne oppgaver layere kan utføre (Tutorialspoint n.d). Modellen har syv layere. Det første Layeret i modellen er det fysiske layeret, en oppgave det utfører er at det multiplexer. Neste layer, data link layer, utfører mange forskjellige oppgaver, blant annet error kontroll, flow kontroll og adressering og framing. Samme oppgave kan bli utført i forskjellige layer. I OSI modellen utføres for eksempel flow kontroll også i transport layer. Error kontroll er også vanlig å utføre i flere enn ett layer.

Kapittel 2

R1

List five nonproprietary Internet applications and the application-layer protocols that they use.

Internett: HTTP; filoverføring: FTP; ekstern pålogging: Telnet; e-post: SMTP; BitTorrent fildeling: BitTorrent-protokoll (Kurose, et al., 2013, s.131)

En applikasjonslagsprotokoll definerer hvordan applikasjonens prosesser, som kjører på forskjellige sluttsystemer, overfører meldinger til hverandre. Det utvikles stadig ny offentlig domene og proprietære Internett-applikasjoner. I denne oppgaven vil vi ta for oss de fem viktigste applikasjonene: the Web, file transfer, electronic mail, directory service, og P2P applikasjoner (Kurose, et al, 2013, s.97). Internett var det første Internett-programmet som fanget opp allmennhetens blikk. Det endret seg dramatisk og fortsetter å endre hvordan mennesker samhandler i og utenfor arbeidsmiljøene sine. Det må også legges til at Internett fungerer etter behov. Brukerne får tilgang til det meste til ønskelig tid. Dette er i motsetning til tradisjonell kringkastingsradio og TV, som kun er tilgjengelige for brukerne når innholdsleverandøren gjør innholdet tilgjengelig alminneligheten (Kurose, et al., 2013, s.98).

HyperText Transfer Protocol (HTTP), nettets applikasjonslags protokoll, og anses som hjertet av nettet. HTTP er implementert i to programmer: et klientprogram og et serverprogram. Klientprogram og serverprogram, som kjøres på forskjellige sluttsystemer, snakker med hverandre ved å utveksle HTTP-meldinger. HTTP definerer strukturen til disse meldingene og hvordan klienten og serveren utveksler meldingene.

I en typisk sett fungerer FTP ved at brukeren overføre filer (fra sin lokalenhet) til eller fra en eksternenhet. For at brukeren skal få tilgang til den eksterne kontoen, må brukeren oppgi en brukeridentifikasjon og et passord. Etter å ha gitt denne autorisasjonsinformasjonen, kan brukeren overføre filer fra det lokale filsystemet til det eksterne filsystemet og omvendt (Kurose, et al, 2013, s.116).

SMTP overfører meldinger fra avsenders e-postservere til mottakerens e-postservere. SMTP er mye eldre enn HTTP. Til forskjell, HTTP overfører filer fra et web server til en webklient, mens SMTP ovrefører filer (e-post meldinger) fra en e-postserver til en annen e-postserver (Kurose, et al., 2013, s.121).

P2P (peer-to-peer) organiseres på annerledes måte enn klient-tjener servere. Arkitekturen bak P2P er at alle datamaskiner samspiller med hverandre på like fot, hvorav en datamaskin utfører tjenester for andre.

P2P støttes som regel av en annen for sentral tjeneste som muliggjør at maskinene kobler seg sammen. Dette kan skje ved bruk av Bluetooth. Eksempelsvis er BitTorrent-protokkollen et eksempel på P2P-arkitektur, denne brukes til fildeling (Ulseth, Dvergsdal, 2021)

R2

What is the difference between network architecture and application architecture?

Nettverksarkitektur referer til organiseringsmønsteret av kommunikasjonsprosessen i lag (f.eks Fem-lags internettarkitektur). Applikasjonsarkitektur er derimot designet av en applikasjonsutvikler og beslutter den brede strukturen i applikasjonen (f.eks Klientserver eller P2P).

> I en klient-server-arkitektur er det en enhet som alltid er på, kjent som serveren, hvilke tjenesteforespørsler fra mange andre enheter, kjent som klienter. Et klassisk eksempel er Webapplikasjon som gjerne fungerer ved at webserver tjeneste ber nettlesere kjøre på klient-enheter.
>
> I en P2P-arkitektur er det minimal (eller ingen) avhengighet av dedikerte servere i datasentre. I stedet utnytter applikasjonen direkte kommunikasjon mellom par av intermitterende tilkoblede enheter, kjent som peers. P2P er designet slik at den vanligvis støttes av en annen sentral tjeneste som fører til at maskinene kan koble seg sammen (Kurose, et al., 2013, s.86).

R3

For a communication session between a pair of processes, which process is the client and which is the server?

Når det gjelder kommunikasjonsøkt mellom et par prosesser, kan vi se på en prosess som klienten og den andre prosessen som serveren. I følge læreboken, defineres klienten og serveren prosesser slik: Prosessen som igangsetter kommunikasjonen er klienten, mens prosessen som venter på å bli kontaktet er serveren. For eksempel i webapplikasjonen en klientleser prosessen utveksler meldinger med en webserverprosess. Men i P2P-fildeling, når Peer A ber Peer B om å sende en bestemt fil, er Peer A klienten

og Peer B er serveren i henhold til denne spesifikke kommunikasjonsøkten (Kurose, et al., 2013, s.89).

R4

For a P2P file-sharing application, do you agree with the statement, "There is no notion of client and server sides of a communication session"? Why or why not?

Nei. Grunnet i et P2P-fildelingsapplikasjon er peer som mottar en fil, vanligvis klienten, og den peer som sender filen er som regel serveren (Kurose, et al., 2013, s.88-89).

R6

Suppose you wanted to do a transaction from a remote client to a server as fast as possible. Would you use UDP or TCP? Why?

Ville velge å bruke UDP, grunnet med UDP kan transaksjonen fullføres på én runde (RTT), klienten sender transaksjonsforespørselen til en UDP-kontakt, og serverne sender svaret tilbake til klientens UDP-kontakt. Med TCP er det minimum to RTT-er som trengs for å fullføre transaksjonen, hvorav en for å sette opp TCP-tilkoblingen, og en for at klienten skal sende forespørselen, og for at serveren skal sende svaret tilbake (Kurose, et al., 2013, s.94-96).

R9

Recall that TCP can be enhanced with SSL to provide process-to-process security services, including encryption. Does SSL operate at the transport layer or the application layer? If the application developer wants TCP to be enhanced with SSL, what does the developer have to do?

SSL fungerer ved applikasjonslaget. SSL-kontakten tar ukryptert data fra applikasjonslaget, krypterer den og overfører den til TCP-kontakten. Dersom programutvikleren ønsker at TCP skal forbedres med SSL, er det nødvendig at SSL-koden inkluderes i applikasjonen (Kurose, et al., 2013, s.96-97).

R10

What is meant by a handshaking protocol?

En protokoll bruker håndtrykk (handshaking) dersom de to gjaldte enhetene først bytter kontrollpakker før de sender data til hverandre. SMTP bruker håndtrykk på applikasjonslaget, mens HTTP ikke gjør det. SMTP oppnår dette ved å sende en melding fra en sendende e-postserver til en mottakende e-postserver. Når denne forbindelsen blir opprettet/vellykket, utfører serveren og klienten noe applikasjonslag håndtrykk (handshaking), SMTP-klienter og servere introduserer seg selv før informasjonen overføres (Kurose, et al., 2013, s.122).

R11

Why do HTTP, FTP, SMTP, and POP3 run on top of TCP rather than on UDP?

Først skal vi først se nærmere på hva TCP og UDP er. TCP er en tilkoblingsorientert protokoll som datamaskiner til å kommunisere over internett. TCP tilbyr feilkontroll og garanterer levering av data, og at data blir levert i den rekkefølgen de ble sendt. UDP er en tilkoblingsfri protokoll som fungerer akkurat som TCP, men forutsetter at det ikke er behov for feilkontroll og gjenoppretting tjenester. I stedet sender UDP kontinuerlig dataprogrammer til mottakeren om de mottar dem eller ikke (Geeksforgeeks, 2021).
Forespørslene som er knyttet til protokollene; HTTP, FTP, SMTP, og POP3 krever at alle applikasjonsdata mottas i riktig rekkefølge og uten feil, TCP passer dermed bedre til disse protokollene da TCP tilbyr denne tjenesten mens UDP ikke opererer slik (Kurose, et al., 2013, s.117).

R13

Describe how Web caching can reduce the delay in receiving a requested object. Will Web caching reduce the delay for all objects requested by a user or for only some of the objects? Why?

Web cache, eller proxy server er en nettverks entitet som tar imot HTTP requests på vegne av origin Web serveren (Kurose, et al., 2013, s. 110). Ved å lagre objekter "nærmere" klienten får man raskere tilgang til objektene. Det er riktignok ikke alltid det er mindre delay med en web cache, ettersom at det er ikke gitt at det etterspurte objektet ligger lagret i cachen.

R19

Is it possible for an organization's Web server and mail server to have exactly the same alias for a hostname (for example, foo.com)? What would be the type for the RR that contains the hostname of the mail server?

Det er mulig at en organisasjons web server og mail server har nøyaktig samme alias. For eksempel Universitetet i Agder har både alias nettsiden uia.no, og mailadresser knyttet til UiAs mail server ender med @uia.no også. RR recorden vil da være MX ettersom den tillater en institusjon å ha samme aliaserte navn for mail server og en av dens andre servere (Kurose, et al., 2013, s. 140).

R26

InSection2.7, the UDP server described needed only one socket, where as the TCP server needed two sockets. Why? If the TCP server were to support n simultaneous connections, each from a different client host, how many sockets would the TCP server need?

TCP serveren krever opprettelse av to sockets fordi det er en connection-oriented protokoll. Som betyr at før klienten og serveren kan begynne å sende data til hverandre må de "handshake" og etablere en TCP connection. Den ene socketen er for å utføre denne "handshaken", den andre er for å sende data (Kurose, et al., 2013, s. 163).

R27

For the client-server application over TCP described in Section 2.7, why must the server program be executed before the client program? For the client- server application over UDP, why may the client program be executed before the server program?

Grunnen til at server programmet må utføres før klient programmet for TCP serveren er på grunn av handshaken forklart i forrige oppgave. Om serverprogrammet ikke er utført før klientprogrammet utføres, vil ikke klienten klare å etablere en TCP kobling med den. UDP

kalles ofte for en "fire and forget"- protokoll og bryr seg ikke om dataen som sendt blir mottatt på den andre siden (Powercert 2016)

P4

Consider the following string of ASCII characters that were captured by Wireshark when the browser sent an HTTP GET message (i.e., this is the actual content of an HTTP GET message). The characters <cr><lf> are carriage return and line-feed characters (that is, the italized character string <cr> in the text below represents the single carriage-return character that was con- tained at that point in the HTTP header). Answer the following questions, indicating where in the HTTP GET message below you find the answer.

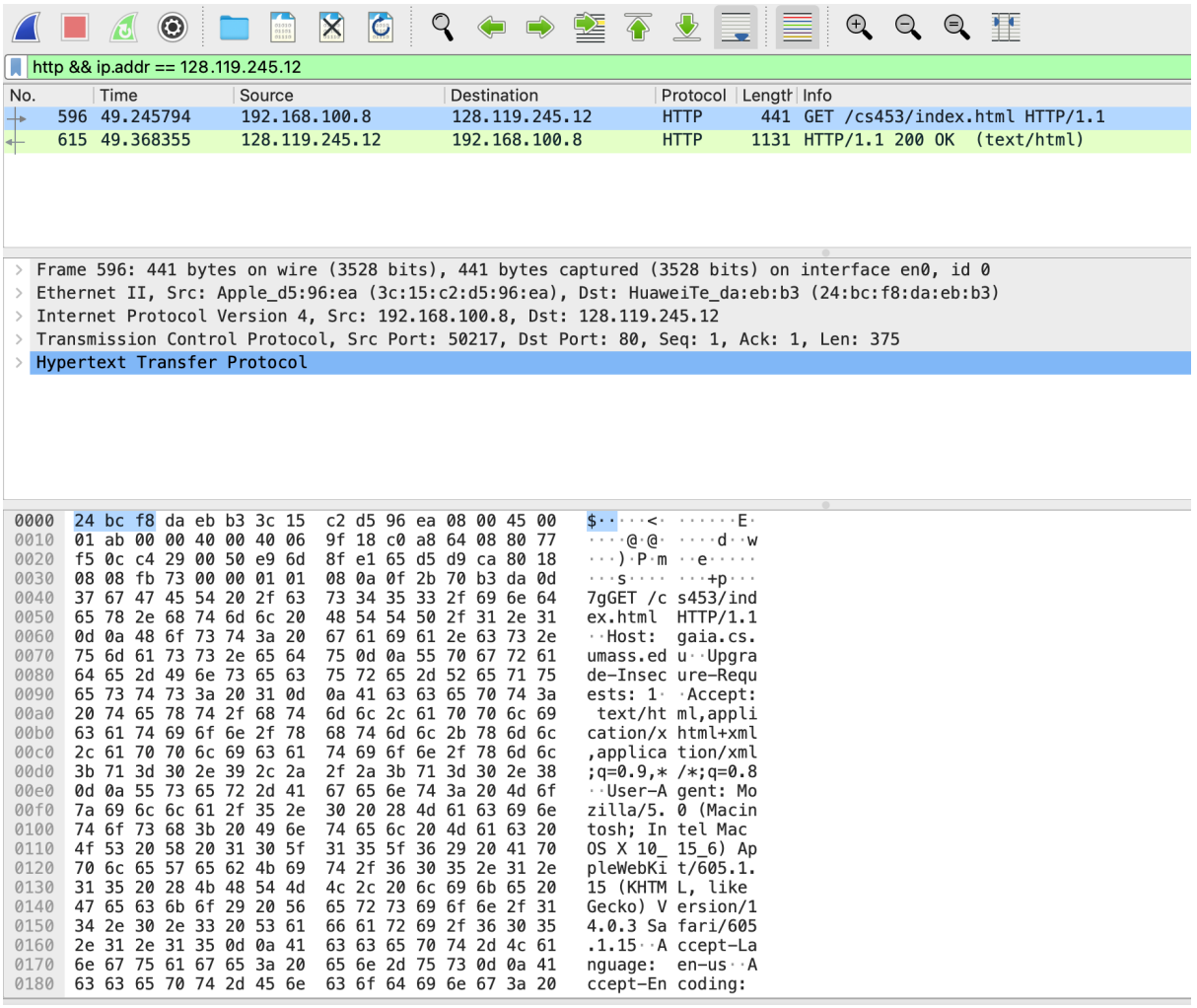
GET /cs453/index.html HTTP/1.1<cr><lf>Host: gai a.cs.umass.edu<cr><lf>User-Agent: Mozilla/5.0 (Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gec ko/20040804 Netscape/7.2 (ax) <cr><lf>Accept:ex

t/xml, application/xml, application/xhtml+xml, text /html;q=0.9, text/plain;q=0.8,image/png,*/*;q=0.5 <cr><lf>Accept-Language: en-us,en;q=0.5<cr><lf>Accept- Encoding: zip,deflate<cr><lf>Accept-Charset: ISO -8859-1,utf-8;q=0.7,*;q=0.7<cr><lf>Keep-Alive: 300<cr> <lf>Connection:keep-alive<cr><lf><cr><lf>

a. **What is the URL of the document requested by the browser?**
   Den første linjen, request linjen, forteller oss at objektet vi er ute etter er /cs453/index.html. Linjen "Host:" forteller oss oss navnet på hosten, eller serveren, hvor objektet ligger, som er gaia.cs.umass.edu. URLen blir da gaia.cs.umass.edu/cs453/index.html
b. **What version of HTTP is the browser running?**
   Den første linjen viser at det er HTTP versjon 1.1.
c. **Does the browser request a non-persistent or a persistent connection?**
   Linjen "connection: keep alive" forteller oss at browseren spør om en persistent tilkobling. Om tilkoblingen hadde vært non persistent, ville linjen sagt "connection: close".

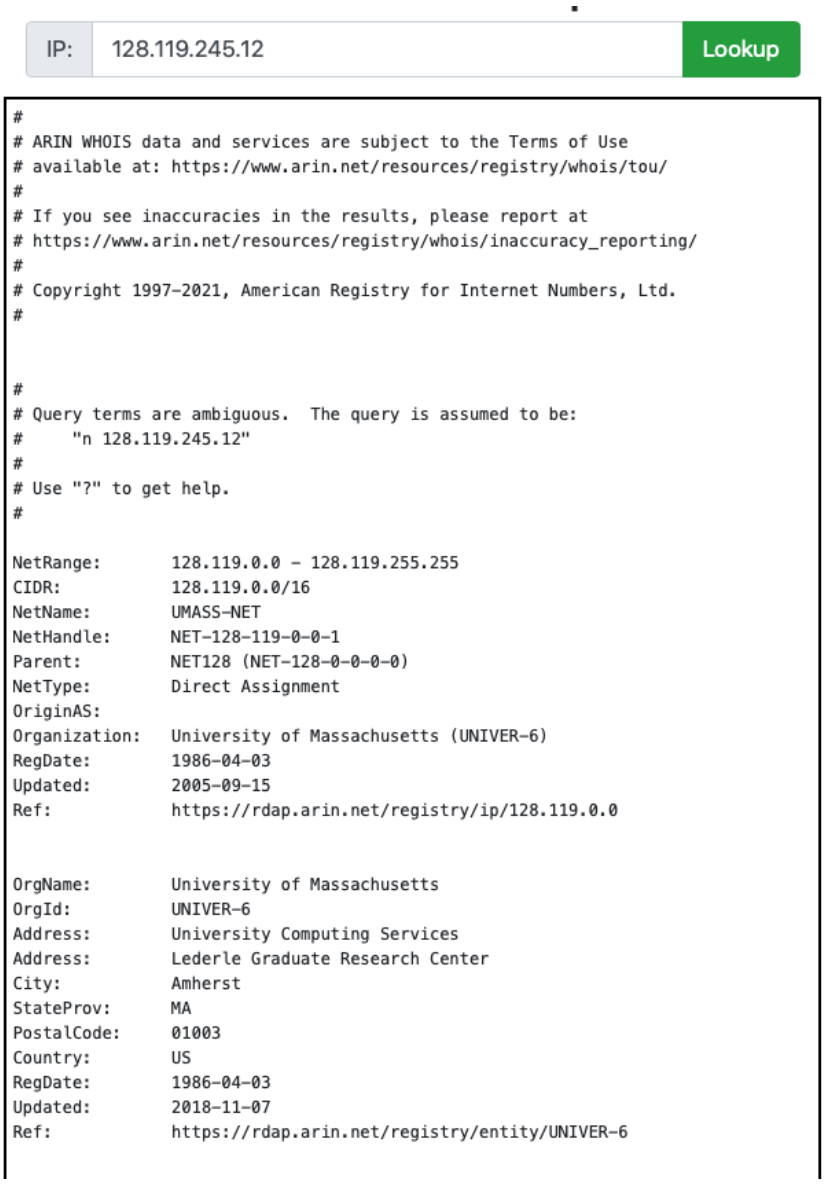
d. What is the IP address of the host on which the browser is running?

IP adressen til host er ikke en del av en HTTP request message, derfor kan men ikke se hva IP adressen er ut fra informasjonen gitt. Men om vi vil finne den ut likevel, så kan vi bruke verktøyet wireshark.



screenshot fra wireshark

Først startet vi capture på wireshark, så brukte vi URLen som vi fant i deloppgave a) for å gå inn på siden i nettleseren safari. Vi har lagt på et filter som gjør at man kun kan se pakkene som er relevante for oppgaven. I første pakke kan vi se at IP adressen til siden er 128.119.245.12, fordi dette er destinasjonen til GET meldingen. Vi kan så sjekke denne IP addressen på en whois database for å se om dette faktisk er IPen til nettsiden.



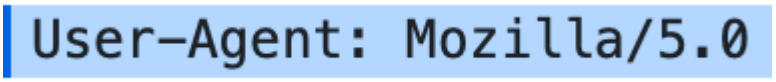
screenshot fra https://www.whatismyip.com/ip-whois-lookup/

Her har vi brukt whois databasen https://www.whatismyip.com/ip-whois-lookup/ for å se hvem som eier IP adressen vi fant med wireshark. Vi kan se at den tilhører University of massachusetts.

e. What type of browser initiates this message? Why is the browser type needed in an HTTP request message?

Grunnen til at nettleseren må spesifiseres er fordi det kan hende at objektet har flere versjoner skreddersydd til forskjellige nettlesere. Det er for eksempel noen funksjoner i html som ikke fungerer i alle nettlesere. Headeren "User agent:" gir oss informasjonen vi trenger for å finne ut hva slags nettleser GET meldingen er sendt fra.

Det er lett å tenke at siden det første som står er mozilla, så er dette nettleseren det er blitt sendt fra. Det var nettopp dette vi trodde etter å ha lest Kurose, et al., 2013, s. 105.
Når vi brukte wireshark for å undersøke GET meldingen på vår egen maskin ble vi mistenksomme til denne informasjonen, for det sto Mozilla der også, enda det var blitt sendt fra en safari nettleser.



screenshot fra wireshark

Ved å undersøke litt nærmere på nettet fant vi ut at User-Agent består av flere komponenter, og at for å se hvilken nettleser det er sendt fra, så må man se litt nærmere enn vi gjorde først. Den første av dem er en generell token som forteller om nettleseren er kompatibel med mozilla. Av historiske grunner sender nesten alle nettlesere denne. Etter dette kommer hvilken plattform som har sendt GET meldingen. Videre er det ganske forskjellig fra nettleser til nettleser hva som kommer, men hvilken nettleser som har sendt meldingen står i alle tilfeller vi fant helt på slutten (MDN, User-Agent 2021).

screenshot fra wireshark

På slutten av User-Agent headeren (markert i blått), kan vi se at det ble sendt fra en Safari nettleser. Med denne informasjonen kunne vi nå se at GET meldingen i oppgaven var sendt fra en **Netscape** nettleser, ikke mozilla.

P5

The text below shows the reply sent from the server in response to the HTTP GET message in the question above. Answer the following questions, indicat- ing where in the message below you find the answer.

HTTP/1.1 200 OK<*cr*><*lf*>Date: Tue, 07 Mar 2008 12:39:45GMT<*cr*><*lf*>Server: Apache/2.0.52 (Fedora) <cr><lf>Last-Modified: Sat, 10 Dec2005 18:27:46 GMT<*cr*><*lf*>ETag: "526c3-f22-a88a4c80"<*cr*><*lf*>Accept- Ranges: bytes<*cr*><*lf*>Content-Length: 3874<*cr*><*lf*> Keep-Alive: timeout=max=100<*cr*><*lf*>Connection: Keep-Alive<*cr*><*lf*>Content-Type: text/html; charset= ISO-8859-1<*cr*><*lf*><*cr*><*lf*><!doctype html public "- //w3c//dtd html 4.0 transitional//en"><*lf*><html><*lf*> <head><*lf*> <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"><*lf*> <meta name="GENERATOR" content="Mozilla/4.79 [en] (Windows NT 5.0; U) Netscape]"><*lf*> <title>CMPSCI 453 / 591 / NTU-ST550A Spring 2005 homepage</title><*lf*></head><*lf*> <*much more document text following here (not shown)*>

a. **Was the server able to successfully find the document or not? What time was the document reply provided?**
   I første linje forteller statuskoden "200 OK" at serveren fant dokumentet. Linjen "Date:" forteller oss at det ble funnet 7. mars 2008 12:39 GMT (Greenwich Mean Time, som ligger to timer foran norsk tid).
b. **When was the document last modified?**
   Linjen "Last modified:" forteller oss at dokumentet ble sist modifisert 10. desember 2005 18:27 GMT.
c. **How many bytes are there in the document being returned?**
   Linjen "content-length:" forteller oss at dokumentet består av 3874 bytes.
d. **What are the first 5 bytes of the document being returned? Did the server agree to a persistent connection?**
   Under responsen får vi se en smakebit av dokumentet, første fem bytes av dokumentet er "<!doc". Som vi kan se i linjen "connection: keep-alive", gikk serveren med på å ha en persistent connection.

P6

Obtain the HTTP/1.1 specification (RFC 2616). Answer the following questions:

a. **Explain the mechanism used for signaling between the client and server to indicate that a persistent connection is being closed. Can the client, the server, or both signal the close of a connection?**

"An HTTP/1.1 server MAY assume that a HTTP/1.1 client intends to maintain a persistent connection unless a Connection header including the connection-token "close" was sent in the request. If the server chooses to close the connection immediately after sending the response, it SHOULD send a Connection header including the connection-token close."

(The Internet Society, 1999, seksjon 8.1.2.1).

Dette er den grunnleggende mekanismen for lukkingen av en persistent connection mellom server og klient. De vil anta at en persistent connection skal opprettholdes inntil de mottar en connection header med connection token close. Både klient og server kan sende en slik token og dermed signalisere at connectionen skal lukkes.

b. **What encryption services are provided by HTTP?**

HTTP tilbyr ikke kryptering. HTTPS er en kryptert versjon av HTTP som sender data sikkert over en kryptert tilkobling (Jackson B 2016). Man kan se hvilken av protokollene en nettside ved å se på URLen, som enten starter med http:// eller https://. Mange moderne nettlesere vil også vise en liten lås ved siden av URLen om det er en HTTPS side, om det er en HTTP side vil det typisk stå “ikke sikker” eller lignende. Dette kommer av at HTTPS er inkludert i SSL sertifikatet, som betyr at det er godkjent av nettleseren. Siden HTTP ikke krypterer data bør man være forsiktig hva man gjør på den. Om man for eksempel skriver inn et passord så vil det være veldig lett for noen med et “sniffer” verktøy (for eksempel wireshark som ble brukt i en tidligere oppgave).

c. **Can a client open three or more simultaneous connections with a given server?**

"Clients that use persistent connections SHOULD limit the number of simultaneous connections that they maintain to a given server. A single-user client SHOULD NOT maintain more than 2 connections with any server or proxy.

...*

These guidelines are intended to improve HTTP response times and avoid congestion", (The Internet Society, 1999, , seksjon 8.1.4).

Ordlyden til dette sitatet fra RFC 2616 forteller oss at det er mulig å ha tre, men at det sterkt frarådes fordi det påvirker negativt på responstiden og fører til opphopning.

*guidelines som ikke har med oppgaven å gjøre er utelatt.

d. Either a server or a client may close a transport connection between them if either one detects the connection has been idle for some time. Is it pos- sible that one side starts closing a connection while the other side is trans- mitting data via this connection? Explain.

"A client, server, or proxy MAY close the transport connection at any time. For example, a client might have started to send a new request at the same time that the server has decided to close the "idle" connection. From the server's point of view, the connection is being closed while it was idle, but from the client's point of view, a request is in progress."

(The Internet Society, 1999, seksjon 8.1.4)

Sitatet fra RFC 2616 forklarer at et scenario hvor en server lukker en connection fordi den bedømmer den som idle, samtidig som en klient begynner å sende en ny request. Løsningen på dette problemet er at klienter, servere og proxier må ha muligheten til å recovere fra asynkrone close-events (The Internet Society, 1999, seksjon 8.1.4).

P12

Write a simple TCP program for a server that accepts lines of input from a client and prints the lines onto the server's standard output. (You can do this by modifying the TCPServer.py program in the text.) Compile and execute your program. On any other

machine that contains a Web browser, set the proxy server in the browser to the host that is running your server program; also con- figure the port number appropriately. Your browser should now send its GET request messages to your server, and your server should display the messages on its standard output. Use this platform to determine whether your browser generates conditional GET messages for objects that are locally cached.

```
import java.io.*;

import java.net.*;


class TCPServer{

public static void main(String argv[]) throws Exception{

String clientMessage;

//setter socket til 6789

ServerSocket messageSocket = new ServerSocket(6789);

//evig loop som tar imot messages fra socket og printer til
konsollen (obs ingen løsning i programmet for å avslutte det,
må derfor manuelt termineres)

while(true) {

   Socket connectionSocket = messageSocket.accept();

   BufferedReader bd = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));

   clientMessage = bd.readLine();

   System.out.println("Client message recieved: " +
clientMessage + "\n");

}

}

}
```

Output i terminalen:

```
Kristians-MacBook-Pro:oppgaveP12_kap2 kristianskibrek$ java TCPServerDemo
Client message recieved: CONNECT clients2.google.com:443 HTTP/1.1

Client message recieved: GET http://google.com/ HTTP/1.1
```

Kommentarer til programmet:

Programmet er hentet fra http://www.sr2jr.com/textbook-solutions/computer-science/10202012/computer-networking-a-top-down-approach-application-layer det er ukjent hva slags operativsystem det er skrevet på, men det ble i eksempelet utført på macOS big sur. For å utføre oppgaven startet vi først programmet på en datamaskin (mac). Så satte vi opp en annen pc (windows 10). Vi satte proxy server på windows pcen til IP adressen til macen og brukte så google chrome for å gå inn på google.com. Dette resulterte i at to http meldinger ble sendt. Først CONNECT hvor windows pcen ber macen (som nå er proxy), om å koble seg til en kilde (DNS, CONNECT, 2021). Det neste er GET, vi kan se at vi vil ha nettsiden google.com og at den bruker http 1.1. Her kan vi også se at denne nettleseren ikke bruker kondisjonelle GET meldinger, ettersom den da ville inneholdt en kondisjon som for eksempel "if-unmodified-since" eller "if-match" (DNS, HTTP conditional headers, 2021).

P18

a. **What is a whois database?**
   Navnet WHOIS forteller oss allerede mye om hva en WHOIS database er. Det er ikke et forkortelse, det er rett og slett det engelske uttrykket "who is" (på norsk "hvem er"). WHOIS databaser forteller oss hvem som er ansvarlige for domener og IP adresser. Gjennom WHOIS databaser kan man finne info som for eksempel identifikasjon og kontaktinformasjon som navn, telefonnummer, administrative og tekniske kontakter. Dataen blir administrert av såkalte registrars (WHOIS, 2017).

b. **Use various who is databases on the Internet to obtain the names of two DNS servers. Indicate which whois databases you used.**
   Vi valgte å bruke nettsiden w3schools.com som vårt eksempel. Først nettsiden https://lookup.icann.org og fant de to navneserverne NS1.MAXIMUMASP.COM og NS2.MAXIMUMASP.COM. Vi brukte også http://ipinfo.info/html/ip_checker.php og fant de to samme serverne.

Nameservers:
NS1.MAXIMUMASP.COM
NS2.MAXIMUMASP.COM

screenshot fra https://lookup.icann.org

Name Server: NS1.MAXIMUMASP.COM
Name Server: NS2.MAXIMUMASP.COM

screenshot fra http://ipinfo.info/html/ip_checker.php

c. **Use nslookup on your local host to send DNS queries to three DNS servers: your local DNS server and the two DNS servers you found in part (b). Try querying for Type A, NS, and MX reports. Summarize your findin**

```
Kristians-MacBook-Pro:~ kristianskibrek$ nslookup
> set type=A
> ns1.MAXIMUMASP.COM
Server:         192.168.100.1
Address:        192.168.100.1#53

Non-authoritative answer:
Name:   ns1.MAXIMUMASP.COM
Address: 216.128.31.5
> ns2.MAXIMUMASP.COM
Server:         192.168.100.1
Address:        192.168.100.1#53

Non-authoritative answer:
Name:   ns2.MAXIMUMASP.COM
Address: 216.128.31.6
```

screenshot terminal nslookup type A

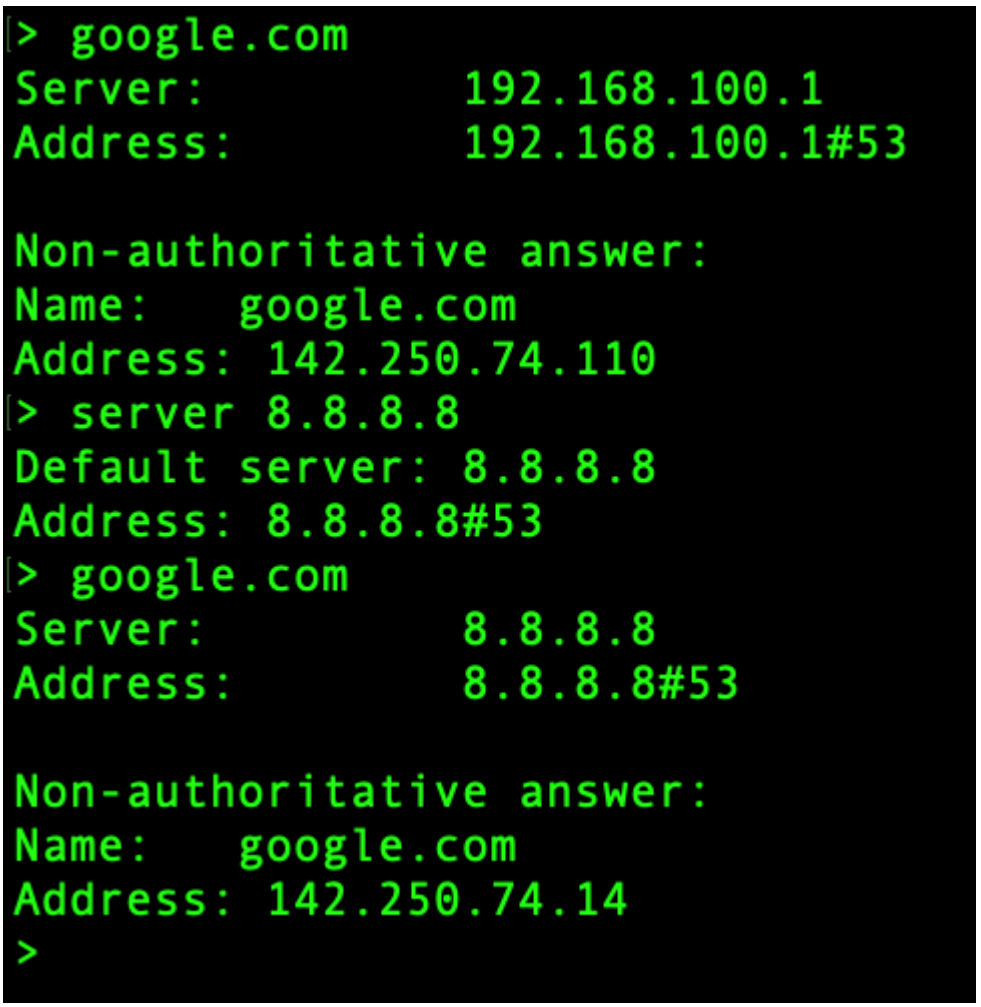
```
> set type=NS
> ns1.MAXIMUMASP.COM
Server:         192.168.100.1
Address:        192.168.100.1#53

Non-authoritative answer:
*** Can't find ns1.MAXIMUMASP.COM: No answer

Authoritative answers can be found from:
MAXIMUMASP.COM
        origin = ns1.MAXIMUMASP.COM
        mail addr = admin.MAXIMUMASP.COM
        serial = 2008048957
        refresh = 7200
        retry = 600
        expire = 1209600
        minimum = 60
> ns2.MAXIMUMASP.COM
Server:         192.168.100.1
Address:        192.168.100.1#53

Non-authoritative answer:
*** Can't find ns2.MAXIMUMASP.COM: No answer

Authoritative answers can be found from:
MAXIMUMASP.COM
        origin = ns1.MAXIMUMASP.COM
        mail addr = admin.MAXIMUMASP.COM
        serial = 2008048957
        refresh = 7200
        retry = 600
        expire = 1209600
        minimum = 60
```

screenshot terminal nslookup type NS

```
> set type=MX
> ns1.MAXIMUMASP.COM
Server:         192.168.100.1
Address:        192.168.100.1#53

Non-authoritative answer:
*** Can't find ns1.MAXIMUMASP.COM: No answer

Authoritative answers can be found from:
MAXIMUMASP.COM
        origin = ns1.MAXIMUMASP.COM
        mail addr = admin.MAXIMUMASP.COM
        serial = 2008048957
        refresh = 7200
        retry = 600
        expire = 1209600
        minimum = 60
> ns2.MAXIMUMASP.COM
Server:         192.168.100.1
Address:        192.168.100.1#53

Non-authoritative answer:
*** Can't find ns2.MAXIMUMASP.COM: No answer

Authoritative answers can be found from:
MAXIMUMASP.COM
        origin = ns1.MAXIMUMASP.COM
        mail addr = admin.MAXIMUMASP.COM
        serial = 2008048957
        refresh = 7200
        retry = 600
        expire = 1209600
        minimum = 60
>
```

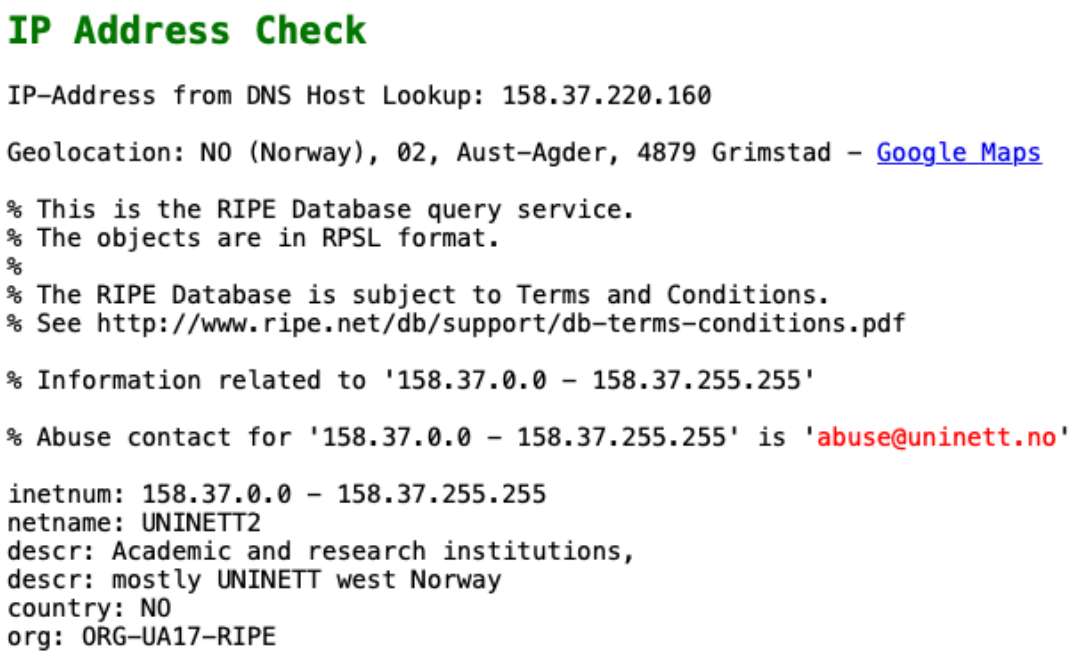
screenshot terminal nslookup type MX

d. **Use nslook up to find a Web server that has multiple IP addresses. Does the Web server of your institution (school or company) have multiple IP addresses?**
Et eksempel på er server med flere IP adresser er google.com. På bildet under er det demonstrert dette. Ved å bytte DNS server kan man se at nslookup kommer fram til to forskjellige IP adresser for google 142.250.74.110 og 142.250.74.14.



screenshot terminal nslookup med via to forskjellige DNS servere

e. **Use the ARIN whois database to determine the IP address range used by your university.**
Fordi uia har en .no side er det ikke mulig å bruke ARINs whois database. Vi brukte derfor heller bruke en annen whois database. Vi brukte http://ipinfo.info/html/ip_checker.php. Vi brukte denne fordi den var anbefalt av foreleser. Den bruker NORID databasen, som har informasjon om .no sider Ved å søke på uia.no fant vi ut at den har en IP range mellom 158.37.0.0 - 158.37.255.255



IP Address Check

IP-Address from DNS Host Lookup: 158.37.220.160

Geolocation: NO (Norway), 02, Aust-Agder, 4879 Grimstad - Google Maps

% This is the RIPE Database query service.
% The objects are in RPSL format.
%
% The RIPE Database is subject to Terms and Conditions.
% See http://www.ripe.net/db/support/db-terms-conditions.pdf

% Information related to '158.37.0.0 - 158.37.255.255'

% Abuse contact for '158.37.0.0 - 158.37.255.255' is 'abuse@uninett.no'

inetnum: 158.37.0.0 - 158.37.255.255
netname: UNINETT2
descr: Academic and research institutions,
descr: mostly UNINETT west Norway
country: NO
org: ORG-UA17-RIPE

f. **Describe how an attacker can use whois databases and the nslookup tool to perform reconnaissance on an institution before launching an attack.**
En angriper kunne brukt nslookup til å vite hvilke IP adresser som tilhører institusjonen, og dermed hvilke IP adresser den skal angripe.

g. **Discuss why whois databases should be publicly available.**
Brownell forklarer at WHOIS databaser demokratiserer internettet, fordi hvem som helst kan få tilgang til en WHOIS database og finne ut hvem som er bak et domenenavn og assosiert nettside. WHOIS databaser kan også brukes hvem som står bak domener og IPer som driver med uetiske og ulovlige ting (Chen T 2014).

Assignment 2

In this programming assignment, you will write a client ping program in Python. Your client will send a simple ping message to a server, receive a corresponding pong message back from the server, and determine the delay between when the client sent the ping message and received the pong message. This delay is called the Round Trip Time (RTT). The functionality provided by the client and server is similar to the functionality provided by standard ping program available in modern operating systems. However,

standard ping programs use the Internet Control Mes- sage Protocol (ICMP) (which we will study in Chapter 4). Here we will create a nonstandard (but simple!) UDP-based ping program.

Your ping program is to send 10 ping messages to the target server over UDP. For each message, your client is to determine and print the RTT when the correspon- ding pong message is returned. Because UDP is an unreliable protocol, a packet sent by the client or server may be lost. For this reason, the client cannot wait indefinitely for a reply to a ping message. You should have the client wait up to one second for a reply from the server; if no reply is received, the client should assume that the packet was lost and print a message accordingly.

In this assignment, you will be given the complete code for the server (avail- able in the companion Web site). Your job is to write the client code, which will be very similar to the server code. It is recommended that you first study carefully the server code. You can then write your client code, liberally cutting and pasting lines from the server code.

Server:

```
#kopiert med noen små endringer fra:
#https://github.com/karanheart96/UDP_Pinger

import random
from socket import *

# Create a UDP socket
# We should use SOCK_DGRAM for UDP packets
serverSocket = socket(AF_INET, SOCK_DGRAM)

# Assign IP address and port number to socket
serverSocket.bind(('127.0.0.1', 6789))

while True:
```

```
    # Generate random number in the range of 0 to 10
    rand = random.randint(0, 10)


    # Receive the client packet along with the address it is
coming from
    message, address = serverSocket.recvfrom(6789)


    # Capitalize the message from the client
    message = message.upper()


    # If rand is less is than 4, we consider the packet lost
and do not respond
    if rand < 4:
        continue


    # Otherwise, the server responds
    serverSocket.sendto(message, address)
```

klient

```
from socket import *
from time import time, ctime
import sys


# Inputs three arguments.


if (len(sys.argv) != 3):
    print(len(sys.argv))
    print("Wrong number of arguments.")
    print("Use: UDPPingClient.py <server_host> <server_port>")
    sys.exit()
```

```
# Preparing the socket
serverHost, serverPort = sys.argv[1:]
clientSocket = socket(AF_INET, SOCK_DGRAM)
clientSocket.settimeout(1)


# Send and receive 10 requests.
for i in range(10):
   startTime = time() # Retrieve the current time
   message = "Ping " + str(i+1) + " " +
ctime(startTime)[11:19]


   try:


       # Sending the message and waiting for the answer
       clientSocket.sendto(message.encode(),(serverHost,
int(serverPort)))
       encodedModified, serverAddress =
clientSocket.recvfrom(1024)


       # Checking the current time and if the server answered
       endTime = time()


       # Modified message is  decoded.
       modifiedMessage = encodedModified.decode()
       print(modifiedMessage)


       # Prints the RTT
       print("Round Trip Time: %.3f ms\n" % ((endTime -
startTime)*1000))
   except:
       print("PING %i Request timed out\n" % (i+1))
```

```
clientSocket.close()
```

Output i terminalen

```
Kristians-MacBook-Pro:Assignment2_kap2 kristianskibrek$ python UDPpingerClient.py 127.0.0.1 6789
PING 1 16:57:11
Round Trip Time: 1.176 ms

PING 2 16:57:11
Round Trip Time: 0.140 ms

PING 3 16:57:11
Round Trip Time: 0.095 ms

PING 4 16:57:11
Round Trip Time: 0.107 ms

PING 5 Request timed out

PING 6 16:57:12
Round Trip Time: 0.345 ms

PING 7 16:57:12
Round Trip Time: 0.196 ms

PING 8 Request timed out

PING 9 Request timed out

PING 10 16:57:14
Round Trip Time: 0.411 ms
```

Kommentar til programmet

Koden er hentet fra https://github.com/karanheart96/UDP_Pinger, ukjent hvilket operativsystem det ble skrevet i, men det ble kjørt på en macOS big sur. Klienten og serveren må kjøres i hver sin terminal. Klienten ble kjørt i en bash terminal, og Serveren ble kjørt fra terminalen i VScode.

Poenget med denne oppgaven er å på en enkel måte simulere pinging. Pinging er å sende en liten datapakke og så måle hvor lang tid det tar før den returneres. Det simuleres her ved at klienten sender pakkene til serveren. Vi satte opp programmet slik at det kan utføres på en maskin, ved å bruke loopback IPen 127.0.0.1. Siden klienten og serveren kobles sammen med en UDP connection, så er det ikke sikkert at alle pakkene kommer seg fram og tilbake, dette simulerer den ved at serveren i linje 26 sier at om et tilfeldig tall mellom 0 og 10 er mindre enn 4 så sendes ikke pakken tilbake. Dette vil da resultere i at klienten på sin side dømmer pakken som "timed out", og gir opp å vente på den. Selv om

En annen måte man kan utforske pinging på er ved å bruke terminalen. På unix og linux kan man bruke kommandoen ping etterfulgt av en IP eller nettside (Ornbo, 2019). Vi testet dette

med forskjellige nettsider som google.com og uia.no. I tillegg testet vi med 127.0.0.1, slik som vi brukte i programmet.

```
Kristians-MacBook-Pro:Assignment2_kap2 kristianskibrek$ ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.081 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.160 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.145 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.132 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.186 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.139 ms
64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.077 ms
64 bytes from 127.0.0.1: icmp_seq=7 ttl=64 time=0.116 ms
64 bytes from 127.0.0.1: icmp_seq=8 ttl=64 time=0.155 ms
64 bytes from 127.0.0.1: icmp_seq=9 ttl=64 time=0.134 ms
^C
--- 127.0.0.1 ping statistics ---
10 packets transmitted, 10 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.077/0.133/0.186/0.032 ms
```

screenshot terminal ping loopback IP

```
Kristians-MacBook-Pro:Assignment2_kap2 kristianskibrek$ ping uia.no
PING uia.no (158.37.220.160): 56 data bytes
64 bytes from 158.37.220.160: icmp_seq=0 ttl=55 time=14.785 ms
64 bytes from 158.37.220.160: icmp_seq=1 ttl=55 time=14.068 ms
64 bytes from 158.37.220.160: icmp_seq=2 ttl=55 time=16.260 ms
64 bytes from 158.37.220.160: icmp_seq=3 ttl=55 time=15.241 ms
64 bytes from 158.37.220.160: icmp_seq=4 ttl=55 time=16.483 ms
64 bytes from 158.37.220.160: icmp_seq=5 ttl=55 time=14.484 ms
64 bytes from 158.37.220.160: icmp_seq=6 ttl=55 time=14.301 ms
64 bytes from 158.37.220.160: icmp_seq=7 ttl=55 time=17.195 ms
64 bytes from 158.37.220.160: icmp_seq=8 ttl=55 time=15.519 ms
64 bytes from 158.37.220.160: icmp_seq=9 ttl=55 time=21.122 ms
^C
--- uia.no ping statistics ---
10 packets transmitted, 10 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 14.068/15.946/21.122/1.978 ms
```

screenshot terminal ping uia.no

Når vi sammenligner resultatene fra resultatene med programmet vårt i forhold til med ping kommandoen, så kan vi se at programmet gir veldig like resultater med 127.0.0.1, med reultater som ligger gjennomsnittlig rundt 0.13 ms. Når vi sammenligner det med pinging mot uia.no, så ser vi at det er ganske mye raskere å pinge loopback IP enn en nettside.

Kapittel 3:

R1

Suppose the network layer provides the following service. The network layer in the source host accepts a segment of maximum size 1,200 bytes and a destination host address from the transport layer. The network layer then guarantees to deliver the segment to the transport layer at the destination host. Suppose many network application processes can be running at the destination host.

a. Design the simplest possible transport-layer protocol that will get application data to the desired process at the destination host. Assume the operating system in the destination host has assigned a 4-byte port number to each running application process.

Vi kaller protokollen for Simple Transport protokollen. På avsender-siden skal dataen ikke overstige 1196 bytes som taes fra Simple Transport Protokollen. Avsender-siden sender også en adresse for destinasjons verten, og et destinasjon portnummer. Simple Transport Protokollen godtar 4 bytes med destinasjons portnummer og vertsadresse. Destinasjons vertsadressen og det resulterende segmentet blir gitt fra protokollen til nettverkslaget, og nettverkslaget gir segmentet til destinasjons verten og protokollen. Deretter vil protokollen undersøke portnummeret i segmentet og trekke ut dataene som portnummeret har identifisert.

b. Modify this protocol so that it provides a "return address" to the destination process.

I segmentet er det to overskriftsfelter:

1. Destinasjons-portfelt
2. Kilde-portfelt

I segmentet blir det laget destinasjons-portnummer, kilde og applikasjonsdata. Og nettverkslaget får destinasjons verten tilsendt av protokollen. Protokollen deretter gir prosessen applikasjonsdata og portnummer etter å ha fått vertsadressen.

c. In your protocols, does the transport layer "have to do anything" in the core of the computer network?

Transportlaget er i "endesystemet", så svaret er nei ettersom den ikke har noe i kjernen.

(Gryczka, 2014)

R2

Consider a planet where everyone belongs to a family of six, every family lives in its own house, each house has a unique address, and each person in a given house has a unique name. Suppose this planet has a mail service that delivers letters from source house to destination house. The mail service requires that (1) the letter be in an envelope, and that (2) the address of the destination house (and nothing more) be clearly written on the envelope. Suppose each family has a delegate family member who collects and distributes letters for the other family members. The letters do not necessarily provide any indication of the recipients of the letters.

a. Using the solution to Problem R1 above as inspiration, describe a protocol that the delegates can use to deliver letters from a sending family member to a receiving family member.

Destinasjonsnavn og adressen blir skrevet på et brev av planetens mail-tjeneste representant, som da må bli gitt ut. Mottakerdelegaten vil ta brevet og se på mottakerens navn på toppen og så gi det til familiemedlemmer som er relatert til mottakerens navn.

b. In your protocol, does the mail service ever have to open the envelope and examine the letter in order to provide its service?

Mail-tjenesten trenger ikke se på brevet, kun se adressen og sende det videre - så svaret er nei (Gryczka, 2014).

R3

Consider a TCP connection between Host A and Host B. Suppose that the TCP segments traveling from Host A to Host B have source port number x and destination port number y. What are the source and destination port numbers for the segments traveling from Host B to Host A?

Svaret er kilde destinasjons portnummer x og kilde portnummer y (Gryczka, 2014).

R4

Describe why an application developer might choose to run an application over UDP rather than TCP.

Det kan skje at når TCP'en blir overbelastet, kan programmets sendefrekvens bli redusert. Denne overbelastning kontrollen til TCP er en grunn til at utviklere velger UDP fremfor TCP, nettopp fordi UDP ikke har noe grense (Gryczka, 2014).

R5

Why is it that voice and video traffic is often sent over TCP rather than UDP in today's Internet? (Hint: The answer we are looking for has nothing to do with TCP's congestion-control mechanism.)

Grunnen til at stemme og video trafikk blir sendt over TCP enn UDP er fordi de fleste brannmurene har en konfigurasjon som blokker UDP trafikk. Det å bruke TCP for video og stemme trafikk gjør at man kan sende det gjennom brannmurer.

En annen fordel med å sende voice and video traffic gjennom TCP er fordi voice packets vil ikke fungere hvis du mister en packet her og der under transporteringen, hele streamen må komme som en hel stream for at det skal fungere. TCP har mindre packet loss rate enn UDP, som gjør den mer egnet for å sende voice. Det samme gjelder for video ved at hvis alt for mange packets mistes, vil ikke videoen gå helt gjennom. (*Understanding Voice Clients and Voice Traffic*, 2020)

R6

Is it possible for an application to enjoy reliable data transfer even when the application runs over UDP? If so, how?

Ja. Utvikleren av applikasjonen kan putte reliable data transfer inn i application layer protocol. Men ved å gjøre det må man gå gjennom enda mer arbeid og debugging.

UDP er ikke designet for å være rask eller reliable. TCP er bygd for å sende packets og meldinger raskt, noe UDP mangler. Men du kan legge til reliability i UDP. Eneste problemet med dette er at packet loss fremdeles er en faktor med UDP, slik at applikasjonen må kunne sende en call til UDP hvis en packet ble mistet og må sendes på nytt. (Jatti B. 2020)

R7

Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host A and Host B each send a UDP segment to Host C with destination port number 6789. Will both of these segments be directed to the same socket at Host C? If so, how will the process at Host C know that these two segments originated from two different hosts?

Begge segmentene vil bli rettet mot samme socket. For hvert segment som blir mottatt ved socket interfacen, så vil OS gi en prosess med IP adressen for å finne ut av hvor de individuelle socketene kom fra. (*cpentalk*, 2020)

R8

Suppose that a Web server runs in Host C on port 80. Suppose this Web server uses persistent connections, and is currently receiving requests from two different Hosts, A and B. Are all of the requests being sent through the same socket at Host C? If they are being passed through different sockets, do both of the sockets have port 80? Discuss and explain.

Forespørslene A og B går gjennom forskjellige sockets. Grunnen til dette er at for hver connection, så vil Web serveren lage en separat "connection socket". Hver av disse er identifisert med 4 felt: IP adresse, portnummer, destinasjon IP adresse og destinasjon portnummer. Når Host C mottar et IP datagram, så vil den gå gjennom de 4 feltene i datagrammet for å sjekke hvilken socket den burde gå gjennom i TCP. Siden A og B har forskjellige IP adresser så vil de ikke gå gjennom samme socket. (sr2jr, u.å.)

P1

Suppose Client A initiates a Telnet session with Server S. At about the same time, Client B also initiates a Telnet session with Server S. Provide possible source and destination port numbers for

a) **The segments sent from A to S.**
b) **The segments sent from B to S.**
c) **The segments sent from S to A.**
d) **The segments sent from S to B.**
e) **If A and B are different hosts ,is it possible that the source port number in the segments from A to S is the same as that from B to S?**
f) **How about if they are the same host?**

Her finner jeg ut at en telnet session portnummer er 23, eller for å være mer presis at en Telnet protocol har portnummer 23 så S blir 23 ("List of TCP and UDP port numbers". 2021). For A blir det 546 og for B blir det 547. Disse port nummerene er "Dynamic Host Configuration Protocol 6 (DHCPv6)". DHCPv6 er en nettverk protokoll for å konfigurere internet protocol versjon 6(IPv6). ("DHCPv6". 2021)

a) Et forslag på dette segmentet er hvis A har kilde-portnummer 546 og destinasjons-nummer til S er 23.
b) I dette segmentet har med allerede sluttet 23 til S som destinasjons numrene så trenger vi bare kilde portnummer, som kan være for eksempel 547.
c) S til A blir akkurat som oppgave A bare at nå blir kilde portnummer 23 og destinasjons nummeret 546.
d) Her så tar vi S som er 23 som kilde portnummer og B 547 som destinasjonsnummer

Source port number		Destination port number
A 546	----->	S 23
B 547	----->	S 23
S 23	----->	A 546

S 23	----->	B 547

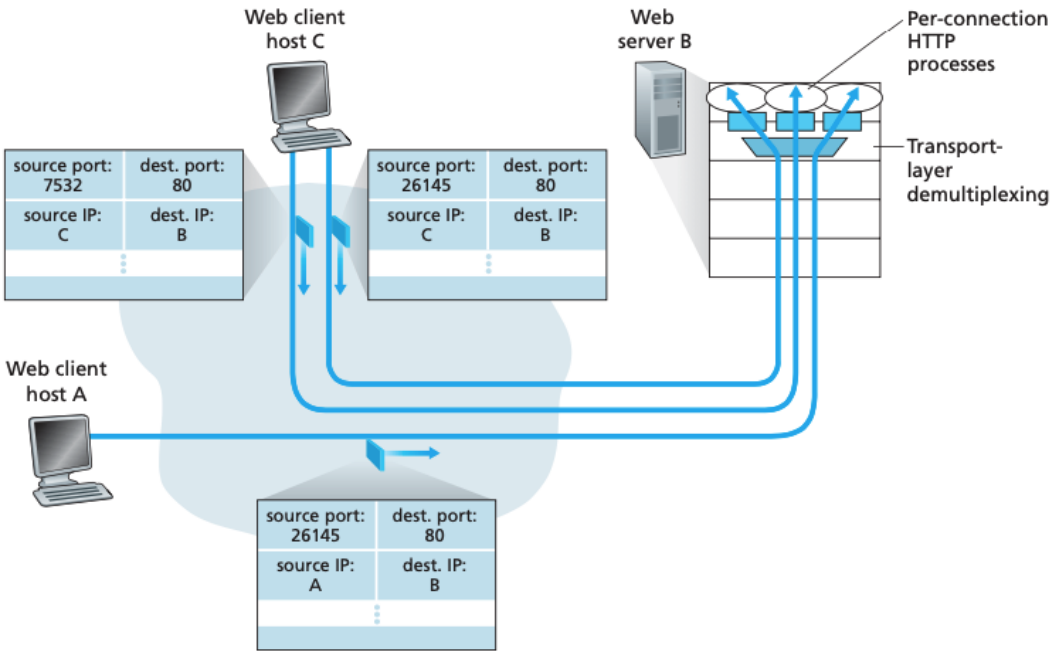
e) Det er mulig ja siden hvis kilde portnummerene i segmenter fra A til S er det samme som det fra B til S kan A og B være forskjellige hosts. Dette funker på grunn det finnes og IP adresser som er med i segmentene og det er sånn de kjenner igjen hostene.

f) Dette ville ikke ha gått siden kilde portene må være forskjellige fra hverandre. Dette er siden det er ikke mulig for kilde portnummerene i segmentene fra A til S er det samme som fra B til S når de er samme host.

P2

Consider Figure 3.5. What are the source and destination port values in the seg- ments flowing from the server back to the clients' processes? What are the IP addresses in the network-layer datagrams carrying the transport-layer segments?

Her har vi figuren 3.5 under som viser kilde portnummer 26145 og destinasjon port nummer 80 til host A. Som betyr at server B kommer til å sende segmentet tilbake til host A med kilde port nummer 80 og destinasjons portnummer 26145. Etter dette vil server B sende tilbake segmentene med kilde port numrene 80 og destinasjons port numrene 26146 og 7532. IP adressene i nettverk-oppsettet datagrammene bærer de transport oppsettet segmentene som er på vei tilbake til enten Host A eller Host C. Kilde IP adressen som er satt til web serverens IP adresse og destinasjons adresse, IP adressen som er satt til klientens IP adresse som er spesifisert som sin egen krav som var originalt tilsendt til serveren.



Figur "3.5" fra boken "*Computer Network: A Top-Down Approach, 6th Edition*. Pearson"(Kurose. et al., 2013, s. 197)

Her ser vi figuren 3.5 fra boken.

P3

UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit bytes: 01010011, 01100110, 01110100. What is the 1s complement of the sum of these 8-bit bytes? (Note that although UDP and TCP use 16-bit words in computing the checksum, for this problem you are being asked to consider 8-bit sums.) Show all work. Why is it that UDP takes the 1s complement of the sum; that is, why not just use the sum? With the 1s complement scheme, how does the receiver detect errors?

01010011 + 01100110 = 10111001

10111001 + 01110100 = 00101110

1s complement = 11010001

For at den skal finne feil så vil mottakeren legge til de 4 ordene(de tre originale og summen). Hvis summen har en null, så vil mottakeren vite at det har oppstått en feil. Alle 1-bit error blir funnet men ikke 2-bit error, de vil forbli usett, som f.eks. hvis det siste tallet på det første

ordet blir gjort om til tallet 0, og det siste tallet i det andre ordet blir til et 1 tall (Kurose, et al., 2013, s. 203).

P4

a. Suppose you have the following 2 bytes: 01011100 and 01100101. What is the 1s complement of the sum of these 2 bytes?

a) Ved å legge til de to bytene får man 11000001. Tar man 1s complement så blir det 00111110.

b. Suppose you have the following 2 bytes: 11011010 and 01100101. What is the 1s complement of the sum of these 2 bytes?

b) Hvis du legger til de to bytene får man 01000000, da vil 1s complement bli 10111111.

c. For the bytes in part (a), give an example where one bit is flipped in each of the 2 bytes and yet the 1s complement doesn't change.

c) Første byte blir 01010100 og den andre byten blir 01101101

P5

Suppose that the UDP receiver computes the Internet checksum for the received UDP segment and finds that it matches the value carried in the checksum field. Can the receiver be absolutely certain that no bit errors have occurred? Explain.

Mottakeren kan IKKE være helt sikker på at ingen bit error har oppstått. Fordi det har noe med hvordan summen av packeten er regnet sammen. Hvis bitene (alt i lag) var 2 ord med 16 bits, og bitene var 0 og 1, selv om disse blir flippet til 1 og 0 så vil summen fremdeles være det samme. 1s complementet som mottakeren regner ut vil være det samme. (cpentalk, u.å.)

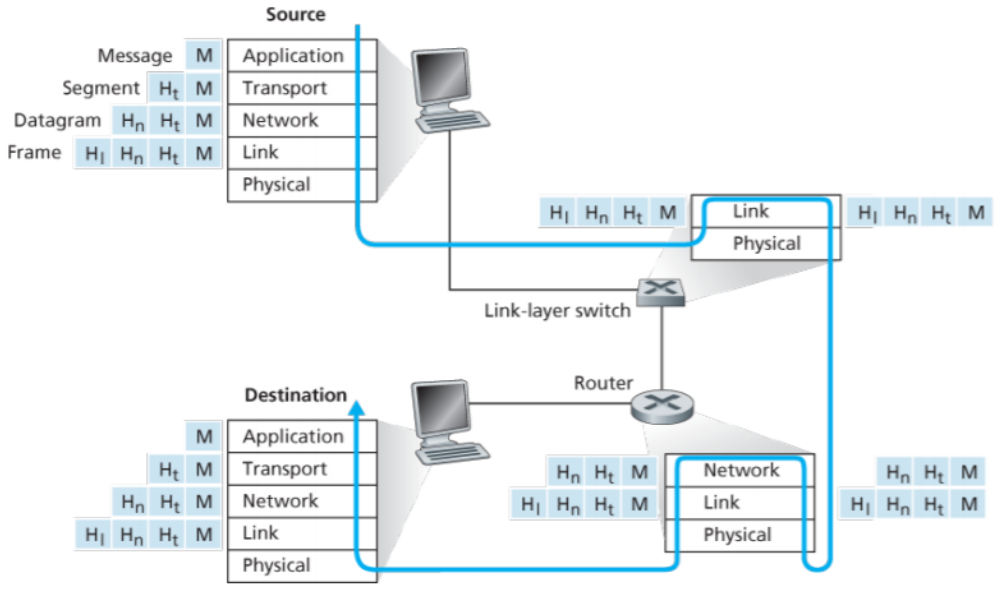
Kapittel 4

R1

Let's review some of the terminology used in this textbook. Recall that the name of a transport-layer packet is segment and that the name of a link-layer packet is frame. What is the name of a network-layer packet? Recall that both routers and link-layer switches are called packet switches. What is the fundamental difference between a router and link-layer switch?

Terminologien på en *network-layer packet* er "datagram" (Kurose, et. al., 2013, s. 51).

Hovedforskjellen mellom en ruter og en link-layer svitsj er at link-layere er mest brukt i 'access networks', mens rutere er mest brukes i nettverk kjernen (Kurose, et. al., 2013,, s. 4) . Link-layer sender et datagram gjennom en mengde rutere fra kilden til destinasjonen. Det er den som sørger for utlevering av datagram mellom nodene (Kurose, et. al., 2013, s. 52).



Figur "1.24" fra "*Computer Network: A Top-Down Approach, 6th Edition*. Pearson"

Bildet viser til de ulike lagene i en ruter og i en link-layer svitsj, og hvordan dette gir dem forskjellig funksjonalitet.

R2

What are the two most important network-layer functions in a datagram network? What are the three most important network-layer functions in a virtual circuit network?

De to viktigste network-layer funksjonene i et datagram nettverk er å forflytte datagram fra en host til en annen. Den andre funksjonen er at den inneholder IP-protokoller som definerer feltene i datagrammet, og hvordan end-systemet og rutere behandler disse (Kurose, et. al., 2013, s. 51)

Funksjonene:

1. *Forwarding*. En pakke forflyttes fra ruterens input link til den aktuelle output linken i ruteren
2. *Routing.* Gjennom en såkalt "Routing algoritme", skal network-layeren bestemme stien pakken skal ta når den skal flyte fra sender til mottaker.

   (Kurose, et. al., 2013, s. 308)

Eksempler på virtual-circuit nettverk er blant annet ATM og frame relay. De viktigste network-layer funksjonene i et virtual-circuit network er:

1. Opprettholder *Connection state information*. Hver gang en ny forbindelse oppstår gjennom ruteren, må en ny tilkobling føres inn i ruterens 'forwarding tabell', og hver gang en tilkobling er over må oppføringen fjernes fra tabellen.
2. *Signaling messages* og *signaling protokoller*, brukes til å sørge for korrekt utveksling av meldinger, og for å sette opp eller terminere en virtual-circuit.

   (Kurose, et. al., 2013, s. 314-316)

R3

We made a distinction between the forwarding function and the routing function performed in the network layer. What are the key differences between routing and forwarding?

Forskjellen mellom routing og forwarding er forwarding handler om en pakkes forflytning fra input til output i en ruter, mens routing handler om bestemmelsen av hvilken sti pakken faktisk skal ta (Kurose, et. al., 2013, ch. 4).

Kapittel 6:

R1

What does it mean for a wireless network to be operating in "infrastructure mode?" If the network is not in infrastructure mode, what mode of operation is it in, and what is the difference between that mode of operation and infrastructure mode?

Når er trådløst nettverk er i "infrastructure mode", er hver trådløse vert, via en basestasjon koblet til et større nettverk. Det er noe som heter ad-hoc modus, som vil si en trådløs nettverk struktur der enheter direkte kan kommunisere med hverandre. Og det er ad-hoc et nettverk opererer i hvis den ikke opererer i "infrastructure mode". Når man ikke har en infrastruktur, må verten selv sørge for tjenester som adressetildeling, ruting osv (Techopedia, u.å), (Yang, 2015).

R2

What are the four types of wireless networks identified in our taxonomy in Section 6.1? Which of these types of wireless networks have you used?

De 4 typene er:

1. Single-hop - infrastruktur basert
2. Single-hop - uten infrastruktur
3. Multi-hop - infrastruktur basert
4. Multi-hop - uten infrastruktur

De som er brukt er de to første (Yang, 2015).

R3

What are the differences between the following types of wireless channel impairments: path loss, multipath propagation, interference from other sources?

Path loss er nedgang i signalstyrke, og skyldes av når det elektromagnetiske signalet dempes i det det beveger seg gjennom materie.

Multipath propagation referer til fiksing av signaler i den elektromagnetiske bølgen, når den reflekterer av gjenstander som får den til å ta stier mellom avsender og mottaker av ulik lengde.

Interference from other sources hender når den andre kilden også overfører i det samme frekvensområdet som det trådløse nettverket.

R4

As a mobile node gets farther and farther away from a base station, what are two actions that a base station could take to ensure that the loss probability of a transmitted frame does not increase?

De to tingene å gjøre er:

1. Å øke overføringskraften
2. Å redusere overføringshastigheten

(Yang, 2015)

R9

Describe how the RTS threshold works.

RTS terskelen kan settes av hver trådløse stasjon, som gjør at RTS/CTS sekvensen kun brukes når data rammen som skal overføres er lengre enn terskelen. Dette resulterer i at det kun er store rammer RTS/CTS-mekanismen brukes til.

(Yang, 2015)

Kapittel 7:

R1

Reconstruct Table 7.1 for when Victor Video is watching a 4 Mbps video, Facebook Frank is looking at a new 100 Kbyte image every 20 seconds, and Martha Music is listening to a 200 kbps audio stream.

Først tar vi Victor Video: Her tar vi 4 megabits per sekund og ganger det med 60. Vi ganger det med 60 for å så kunne gange det med 67. Mer forklart vil dette si at siden det er megabits per sekund og vi skal overføre til 67 min så må vi gange med 60 siden det er 60 sekund i et minutt. Så ganger vi med 67 siden det bytes som skal overføres i 67 min. Når vi gjør dette så får vi dette regnestykket 4x60x67=16080. Her vet vi nå at vi har 16080 mbps men så konverterer vi til noe mindre for eksempel gigabyte. Her har jeg brukt google sin konverter (data transfer) til å få det om til gigabyter som blir 2.01 gbytes.

Facebook Frank: Her har 100 kbyte hver 20 sekund. Her må vi ta 100 kbyte og dele med 20, 100/20=5, her blir det 5 kbyte per sekund. Så som i forrige oppgave tar vi og ganger med 60 og 67, 5x60x67= 20100. Så ved google sin konverterer så gjør vi det om til megabytes som blir 20.1 mbytes.

Martha Music: Her har vi 200 kilobits per sekund. Vi gjør det samme som i første oppgave og tar og ganger det. 200x60x67 = 804000. Så tar vi og konverterer til mbytes. Dette blir då 100.5 mbytes.

	Bit rate	Bytes transferred in 67 min
Victor Video	4 mbps	2.01 GB
Facebook Frank	5 kbytes per sekund	20.1 MB
Martha Music	200kbps	100.5 MB

(Kurose, et al., 2013, s. 588).

R2

There are two types of redundancy in video. Describe them, and discuss how they can be exploited for efficient compression.

De to forskjellig typer redundancy i video er spatial og temporal. Begge to kan være utnyttet for kompresjon. Temporal redundancy reflekter over gjentagelse av bilder til påfølgende bilder. Et eksempel på dette er hvis vi har et bilde og et senere/påfølgende er helt like er det ikke vits å kode det påfølgende kode om igjen. Det er bedre og mer effektivt hvis det oppdager når det koder at følgende bilde er helt lik som det andre. Spatial redundancy er redundancy som er innenfor en gitt bilde. Intuitivt, bilde som består av det meste hvite mellomrom som har en høy grad av redundancy og kan komprimeres uten at bildekvaliteten må svekkes så mye(Kurose, et al., 2013, s. 589).

R3

Suppose an analog audio signal is sampled 16,000 times per second, and each sample is quantized into one of 1024 levels. What would be the resulting bit rate of the PCM digital audio signal?

PCM står for Pulse Code Modulation. PCM er en metode som blir brukt til å digital representere prøver (samples) analog signaler. Det er standard form for digital audio til i datamaskiner, CD og digital telephony. I en PCM stream , er amplitude av analog signaler samplet (sampled) jevnt i uniform intervaller, og hver prøve er kvantifisert til det nærmeste verdi innenfor et gitt utvalg av digitale trinn(“Pulse-code Modulation” .2021). Så må vi kvantifisere 1024 nivåer. Her må vi få 1024 ved å bruke en potens av to. Ved litt kalkulasjon så blir det 1024 = 2 ^ 10. Så tar vi 10 gange de 16 000 prøvene og får 16 000 x 10 = 160 000. Som vil resultere i en rate av 160 kbps (Kurose, et al., 2013, s. 590-591).

R4

Multimedia applications can be classified into three categories. Name and describe each category.

Vi har tre kategorier av multimedia applikasjoner som er, streaming av lagret video/lyd, muntlig video/lyd-over-IP og streaming live video/audio.

1. Streaming lagret video/lyd

I streaming lagret video/lyd applikasjon, en forhåndsinnspilt video/lyd filer er plassert i en server. Så spør kunden serveren om en video/audio til å spille .

Det er tre hovedfunksjoner for streaming lagret videoer. Hovedfunksjonene er Streaming, Interaktivitet og kontinuerlig «playout».

Under Streaming som sagt før spør kunden om tillatelse fra serveren til å spille av videoen som vil bli spilt av om et par sekunder. Neste steg blir då at serveren sender videoen til klienten. Hovedfordelen med streaming er at du trenger ikke å laste ned filen, men heller blir direkte sendt til deg.

Interaktivitet lar brukeren pause en video enkelt og greit, ikke bare det, men og spole frem og tilbake. Dette er mulig hvis videoen er allerede filmet ferdig, ikke under en live stream.

Kontinuerlig «playout» er når klienten må få data fra selve serveren i tide for å kontinuerlig «playout» (Kurose, et al., 2013, s. 591).

2. Muntlig video/lyd-over-IP

Denne applikasjonen tillater brukeren til å ha sanntid video/lyd over internettet, eksempel med dette i våres tid er Messenger, Skype, Zoom, Viber og Microsoft teams (Kurose, et al., 2013, s. 592).

3. Streaming live video/audio

Dette enkelt og sagt tillater brukerne til å se direktesendinger for eksempel av fotballkamper, pressekonferanser osv. Dette brukes via internett, men kan skje feil ved at skjer “delay” med flere sekunder som kommer ved å bruke streaming av live video/audio (Kurose, et al., 2013, s. 593).

R14

Section 7.3 describes two FEC schemes. Briefly summarize them. Both schemes increase the transmission rate of the stream by adding overhead. Does interleaving also increase the transmission rate?

Først så står FEC for «Forward Error Correction».

1. Den første FEC ordningen

I denne ordningen sammen med original informasjon , blir og redundant informasjonen og sent. I denne situasjonen blir ikke originalen informasjonen tapt eller forsvunnet, siden original informasjonen kan bli gjenopprettet fra redundant informasjonen. Altså at redundant informasjonen kan bli brukt til å rekonstruere tilnærminger eller helt like versjoner av noen av de tapte "packets". For eksempel sender det et redundant enkodert "chunk" etter hver n "chunks". Den redundant "chunk" er hentet ved et eksklusiv OR-ing den n originale "chunks". Ved denne måten hvis en "packet" av gruppen n + 1 "packet" er mistet, vil mottakeren klare å rekonstruere den tapte "packet". Men hvis en eller to "packets" i en gruppe er mistet. då kan ikke mottakeren rekonstruere de tapte "packets".

2. Den andre FEC ordningen

Denne ordningen er til å sende en nominal audio stream, lav-resolusjon, lav bitrate audio stream som sender det som redundant informasjon. For eksempel hvis senderen skaper et nominal audio stream og en til tilsvarende lav-resolusjon, lav bitrate audio stream. Den lave bitrate stream er referert til som den redundant stream. På denne måten, hver gang det oppstår en ikke sammenhengende "packet" tap, kan mottakeren skjule dette tapet ved å spille den lave bitrate enkodert "chunk" som kommer med den "subsequent packet".

Den første og andre FEC ordningene øker transmisjonens rate av streamen ved å legge til redundant informasjon. (Kurose, et al., 2013, s. 618)

Kapittel 8 (frivillig)

R1.

What are the differences between message confidentiality and message integrity? Can you have confidentiality without integrity? Can you have integrity without confidentiality? Justify your answer.

Meldingskonfidensialitet (confidentiality) og -integritet er begge to ønsket for sikker kommunikasjon over nettet. Meldingskonfidensialitet betyr at det kun er sender og mottaker som skal kunne forstå innholdet i den sendte meldingen. Meldingsintegritet handler om at innholdet i kommunikasjonen ikke er endret (Kurose et al. 2013 s.677, 688).

Kan man ha konfidensialitet uten integritet? Vi mener det er mulig, konfidensialitet garanterer ikke integritet. Et eksempel er at Person A sender en kryptert melding til person B. Inntrengeren C har ikke mulighet til å se hva som står i meldingen, men C har mulighet til å endre meldingen, sånn at B ikke mottar det A sendte.

Kan man ha integritet uten konfidensialitet? Vi mener dette også er mulig. Integritet garanterer ikke konfidensialitet. Et eksempel er at person A vil sende en melding til B. Inntrengeren C kan se hva som står i meldingen, men har ingen mulighet til å endre det.

R2.

Internet entities (routers, switches, DNS servers, Web servers, user end systems, and so on) often need to communicate securely. Give three specific example pairs of Internet entities that may want secure communication.

I denne oppgaven bruker vi googles definisjon på en entitet "An entity is a thing or a concept that is singular, that is unique, that is well defined, and that is distinguishable." (Shelley, 2021). Når vi da tenker oss internett entiteter, så tenker vi rutere, servere og datamaskiner, men vi tenker også mennesker og organisasjoner som utfører handlinger på nettet. Etter refleksjon av spørmålet kom vi fram til at nesten alle entiteter på nettet vil ha sikker kommunikasjon. Vi satte sammen en liste av de tre situasjonene hvor vi mente sikker kommunikasjon er viktigst.

1. Online shopping og bank transaksjoner:
   Når man betaler for noe over nettet er det viktig å ha sikker kommunikasjon mellom de to systemene betalingen føres mellom. I moderne tid har det blitt utviklet kryptovalutaer, som løser dette problemet ved å bruke blokkjedeteknologi.
2. E-post og andre internett baserte meldingstjenester:
   Over e-post og andre eldingstjenester er ofte sikkerheten viktig, fordi det utveksles hemmelige ting, og viktige ting, som ikke bør bli lest av andre enn de to meldingen blir send mellom.
3. Passord relaterte operasjoner:
   Når man logger inn på en nettside på nettet vi man gjerne at det skal være sikker kommunikasjon mellom systemet passordet blir skrevet inn på, og systemet som mottar passordet.

R3.

From a service perspective, what is an important difference between a symmetric-key system and a public-key system?

I symmetrisk-nøkkel systemer så vet både sender og mottaker den hemmelige krypteringsnøkkelen, mens i offentlig-nøkkel systemer så finnes det en offentlig krypteringsnøkkel, og en privat hemmelig dekrypteringsnøkkel, som bare er tilgjengelig for mottakeren av meldingen (Kurose et al. 2013 s. 678/683).

R4.

Suppose that an intruder has an encrypted message as well as the decrypted version of that message. Can the intruder mount a ciphertext-only attack, a known-plaintext attack, or a chosen-plaintext attack?

Om inntrengeren har både det den krypterte versjonen av meldingen, og den ukrypterte (plaintext), så kan den utføre et *chosen plaintext angrep.*
(Kurose et al. 2013 s.677, 688).

R5.

Consider an 8-block cipher. How many possible input blocks does this cipher have? How many possible mappings are there? If we view each mapping as a key, then how many possible keys does this cipher have?

For en 8 blokks cipher vil det være 2^8 mulige input blokker (=256). Det finnes 256! Mulige måter de kan arrangeres på (mappinger), om man ser på hver av mappingen som en egen key så har cipheren 256! (Kurose et al. 2013 s. 680/677).

R6.

Suppose *N* people want to communicate with each of *N* – 1 other people using symmetric key encryption. All communication between any two people, *i* and *j*, is visible to all other people in this group of *N*, and no other per- son in this group should be able to decode their communication. How many keys are required in the system as a whole? Now suppose that public key encryption is used. How many keys are required in this case?

Om man bruker symmetrisk nøkkel kryptering vil hvert par av personer i systemet dele en nøkkel. Det er to måter man kan tenke seg at dette fungerer, den ene er at sender og mottaker har identiske nøkler. Den andre måten er at de to nøklene som hører sammen er unike nøkler. Om nøklene er unike vil man ha N*(N-1) nøkler. Om hvert par har identiske nøkler, så vil antallet være halvparten, altså N*(N-1)/2.

Om man bruker offentlig nøkkel kryptering, så vil hver bruker ha to nøkkelen, en offentlig og en privat, som vil seultere i 2N nøkler.

R12.

What does it mean for a signed document to be verifiable and non-forgeable?

Et signert dokument i den digitale verden avspeiler signerte dokumenter i virkeligheten. At de er verifiserbare og ikke-forfalskbare betyr derfor det samme som i den virkelige verden.

Verifiserbar betyr at det må være mulig å bevise at dokumentet ble signert av det individet det står at det er signert av. Ikke-forfalskbar betyr at det ikke skal være mulig for noen andre å produsere en denne signaturen (Kurose et al. 2013 s.693).

P1.

Using the monoalphabetic cipher in Figure 8.3, encode the message "This is an easy problem." Decode the message "rmij'u uamu xyj."

```
  public class MonoAlphabeticCipher {
     public static void main(String[] args) {

       //alfabetet og det krypterte alfabetet
       char[] plainText =
"abcdefghijklmnopqrstuvwxyz".toCharArray();
       char[] ciphertext =
"mnbvcxzasdfghjklpoiuytrewq".toCharArray();

       //ukryptert og kryptert input
       char[] plainTextInput = "this is an easy
problem".toCharArray();
       char[] cipherInut = "rmij'u uamu xyj".toCharArray();

       //krypterer det ukrypterte inputet
       for (int i = 0; i < cipherInut.length; i++) {
           for (int j = 0; j < ciphertext.length; j++) {
               if (cipherInut[i]==ciphertext[j]) {
                   cipherInut[i] = plainText[j];
                   break;
               }
           }
       }

       //dekrypterer det krypterte inputet
       for (int i = 0; i < plainTextInput.length; i++) {
           for (int j = 0; j < plainText.length; j++) {
               if (plainTextInput[i]==plainText[j]) {
                   plainTextInput[i] = ciphertext[j];
                   break;
               }
           }
       }

       System.out.println(plainTextInput);
       System.out.println(cipherInut);
   }
```

```
}
```

output i terminalen:

```
Kristians-MacBook-Pro:oppgaveP1_kap8 kristianskibrek$ javac MonoAlphabeticCipher.java
Kristians-MacBook-Pro:oppgaveP1_kap8 kristianskibrek$ java MonoAlphabeticCipher
uasi si mj cmiw lokngch
wasn't that fun
Kristians-MacBook-Pro:oppgaveP1_kap8 kristianskibrek$
```

Kommentar til programmet

Programmet ble skrevet på macOS big sur. Vi valgte å løse oppgaven ved å lage et program som krypterer en setning og dekrypterer en setning. Programmet har to alfabeter, et ukryptert og et kryptert, i form av arrayer av bokstaver. Setningene som skal krypteres, blir også gjort om fra stringer til arrayer av bokstaver. Vi bruker to for looper inni hverandre for å kryptere og dekryptere setningene. Vi valgte å gjøre det på denne måten for å få litt dypere innsikt i hvordan kryptering foregår, selv om denne krypteringen er så enkel, at i den virkelige verden ville den nok gitt mye forsvar. Vi brukte Java fordi det er det programmeringsspråket vi er mest kjent med.

P2.

Show that Trudy's known-plaintext attack, in which she knows the (ciphertext, plaintext) translation pairs for seven letters, reduces the number of possible substitutions to be checked in the example in Section 8.2.1 by approximately 10^9 .

I eksemplet har vi Alice, som sender meldingen "bob, i love you" til mottakeren bob. Inntrengeren Trudy har klart å dekode plaintext pairingen for a, l, i , e, b og o. De bruker monoalfabetisk kryptering (vist i forrige oppgave), hvor det er 26! mulige kombinasjoner (Kurose et al. 2013 s. 677), siden hun nå vet 7 av dem, er det bare 19! Kombinasjoner å prøve ut. Differansen mellom 26! og 19! (= 26 * 25 * 24...20) er 3315312000, som er over 3 ganger så mye som 10^9, men boka mener at disse tallene er omtrent like.

P3.

Consider the polyalphabetic system shown in Figure 8.4. Will a chosen- plaintext attack that is able to get the plaintext encoding of the message "The quick brown fox jumps over the lazy dog." be sufficient to decode all messages? Why or why not?

Med et polyalfabetisk system vil det ikke lenger være nok å ha setningen "the quick brown fox jumps over the lazy dog" (som inneholder alle bokstavene i det engelske alfabetet), fordi med polyalfabetisk kryptering så har hver bokstav flere versjoner (Kurose et al. 2013, s 678). Med krypteringen vist i eksempel 8.4, kan krypteringen forandre seg fra melding til melding, avhengig av verdien til de to nøklene, derfor kan det være at plaintexten Trudy har, ikke vil ha noen nytte i det hele tatt.

Referanseliste

cpentalk.com (u.å.) *Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host A and Host B each send a UDP segment to Host C with destination port number 6789.* hentet 16.05.2020 fra:

https://cpentalk.com/1955/suppose-process-socket-suppose-segment-destination-number?show=1956#a1956

cpentalk.com (u.å.) *Suppose that the UDP receiver computes the Internet checksum for the received UDP segment and finds that it matches the value carried in the checksum field. Can the receiver be absolutely certain that no bit errors have occurred? Explain.* hentet 16.05.2020 fra:

https://cpentalk.com/1949/receiver-computes-internet-checksum-absolutely-occurred

Brownell, N. (n.d). *What is WHOIS and How Is It Used?* Domain.com. https://www.domain.com/blog/2020/07/17/what-is-whois-and-how-is-it-used/ Hentet 03.2021

Chen, T. (2014, 14. mai). *what does whoIs have to do with cybersecurity?* Domaintools. https://www.domaintools.com/resources/blog/what-does-whois-data-have-to-do-with-cybersecurity Hentet 11.05.2021.

GeeksForGeeks. (2021, 19. februar). *Packet switching and delays in computer networks.* Geeksforgeeks.com. https://www.geeksforgeeks.org/packet-switching-and-delays-in-computer-network/ Hentet 03.2021.

GeeksForGeeks. (2021, 22. mars). *Differences between TCP and UDP.* Geeksforgeeks.com https://www.geeksforgeeks.org/differences-between-tcp-and-udp/ Hentet 11.05.2021.

Google. (n.d). *Our edge network is how we connect with ISPs to get traffic to and from users.* Google Peering. https://peering.google.com/#/ Hentet 03.2021.

Gryczka, P. (2014. Oktober). Computer networking chapter 3 review questions. Quizlet.com. https://quizlet.com/55365749/computer-networking-chapter-3-review-questions-flash-cards/ Hentet 13.04.2021.

Juniper Networks. (07.11.2020) *Understanding Voice Clients and Voice Traffic*. Hentet 16.05.2020 fra:

https://www.juniper.net/documentation/en_US/junos-space-apps/network-director3.5/topics/concept/wireless-voice-client-and-traffic.html

Jatti, B.(30.04.2020) *Is it possible for an application to enjoy reliable data transfer even when the application runs over UDP? If so, how?* Quora.com, hentet 16.05.2020 fra:

https://www.quora.com/Is-it-possible-for-an-application-to-enjoy-reliable-data-transfer-even-when-the-application-runs-over-UDP-If-so-how

Jackson, B. (2016, 21. september). *What is the difference between HTTP and HTTPS?.* Keycdn. https://www.keycdn.com/blog/difference-between-http-and-https Hentet 03.2021.

Kurose, James, Ross, Keith. (2013). *Computer Network: A Top-Down Approach, 6th Edition*. Pearson. Hentet fra https://eclass.teicrete.gr/modules/document/file.php/TP326/Θεωρία%20(Lectures)/Computer_Networking_A_Top-Down_Approach.pdf

MDN. (2021, 21. mars). *Web technology for developers-User-Agent*. MDN Web docs. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent Hentet 11.05.2021.

MDN. (2021, 21. mars). *Web technology for developers-CONNECT*. MDN Web docs. https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/CONNECT Hentet 09.05.2021.

MDN. (2021, 21. mars). *Web technology for developers-HTTP conditional headers*. MDN Web docs. https://developer.mozilla.org/en-US/docs/Web/HTTP/Conditional_requests Hentet 09.05.2021

Mitchell, B. (2021, 24. april). *A list of every IP address used by Google.* Lifewire. https://www.lifewire.com/what-is-the-ip-address-of-google-818153

McMahon, M. (2021, 14. februar). *What is an end system*? EasyTechJunkie. https://www.easytechjunkie.com/what-is-an-end-system.htm Hentet 10.05.2021.

Ornbo, G. (2019, 16. november). *Linux and Unix ping command tutorial with examples*. shapeshed.com. https://shapeshed.com/unix-ping/ Hentet 11.05.2021.

Powercert. (2016, 16. november). *TCP vs UDP Comparison*[video]. YouTube. https://www.youtube.com/watch?v=uwoD5YsGACg Hentet 03.2021.

Shelley R (22. februar 2021) What is an Entity § how they impact SEO business2community.com hentet 11.05.2021 fra https://www.business2community.com/brandviews/shelley-media-arts/what-is-an-entity-how-they-impact-seo-02388254
sr2jr (u.å.)

TechTerms. (2017, 21. juni). *WHOIS.* Techterms.com. https://techterms.com/definition/whois Hentet 03.2021.

The internet society. (1999, juni) *RFC 2616.* Datatracker.itef.org. https://tools.ietf.org/html/rfc2616 Hentet 03.2021.

Tutorialspoint. (n.d). *Error Detection & Correction*. Tutorialspoint.com. https://www.tutorialspoint.com/data_communication_computer_network/error_detection_and_correction.htm Hentet 03.2021

Ulseth, T. Dvergsdal, H. (2021, 6. januar). *Peer to peer architecture*. SNL.no. https://snl.no/peer-to-peer-arkitektur Hentet 11.05.2021.

Wikipedia. (2021, 2. mai.). *Wireless WAN*. Wikipedia.com https://en.wikipedia.org/wiki/Wireless_WAN Hentet 03.2021

Wikipedia. (2021, 1. mai). *Wireless LAN*. Wikipedia.com https://en.wikipedia.org/wiki/Wireless_LAN Hentet 03.2021.

Wikipedia. (2021. 15. mars). *Protocol (Diplomacy)*. Wikipedia.com https://en.wikipedia.org/wiki/Protocol_(diplomacy) Hentet 03.2021.

Wikipedia. (2021. 29. april). *Pulse-code modulation*. Wikipedia.com https://en.wikipedia.org/wiki/Pulse-code_modulation Hentet 05. 2021.

Wikipedia. (2021. 10. mai). *List of TCP and UDP port numbers*. Wikipedia.com https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers Hentet 05.2021.

Wikipedia. (2021. 17. april). *DHCPv6*. Wikipedia.com. https://en.wikipedia.org/wiki/DHCPv6#Port_numbers Hentet 05.2021.

WHOIS. (2017, juli). *about whois.* I cann | lookup. https://whois.icann.org/en/about-whois Hentet 11.05.2021.

Yang, D. (2015, juli). *Digital Networking: Chapter 6 Wireless and Mobile Networks*. Quizlet.com. https://quizlet.com/87071074/digital-networking-chapter-6-wireless-and-mobile-networks-flash-cards/#:~:text=What%20does%20it%20mean%20for,operating%20in%20%22infrastructure%20mode%22%3F&text=In%20infrastructure%20mode%20of%20operation,operates%20in%20ad%2Dhoc%20mode Hentet 13.04.2021.