DATAKOM AK-1

# P3

## Sockets

En socket er en mekanisme som gjør det mulig for applikasjoner å kommunisere over et nettverk lokalt eller over internett. En socket består av en IP-adresse og et portnummer som identifiserer den unike adressen til en datamaskin og en spesifikk applikasjon som kjører på den. Ved hjelp av sockets kan to applikasjoner koble seg til hverandre og sende og motta data, uavhengig av hvilket operativsystem de kjører på. Sockets brukes ofte i nettverksapplikasjoner som kommunikasjonsverktøy mellom klienter og servere.

## TCP

TCP står for Transmission Control Protocol og er brukt over internett og andre nettverk som form for kommunikasjon. TCP garanterer at informasjonen som sendes blir sendt i riktig rekkefølge og at det ikke går tapt ved kommunikasjon.

Ved bruk av TCP må det brukes en 3-way-handshake/"treveis håndtrykk" for oppkobling som består av en SYN-melding fra klienten til serveren, en SYN-ACK-melding fra serveren til klienten, og til slutt en ACK-melding fra klienten til serveren. Etter tilkoblingen kan data utvekslingen begynne ved bruken av TCP-segmenter som består av en datastrøm, et sekvensnummer og en bekreftelse på mottak av data.

Sekvensnummer:

Et sekvensnummer er et unikt nummer som tildeles hver byte med data som sendes over nettverket. Sekvensnummeret brukes av den mottakende enheten for å sette sammen dataene i sin opprinnelige rekkefølge.

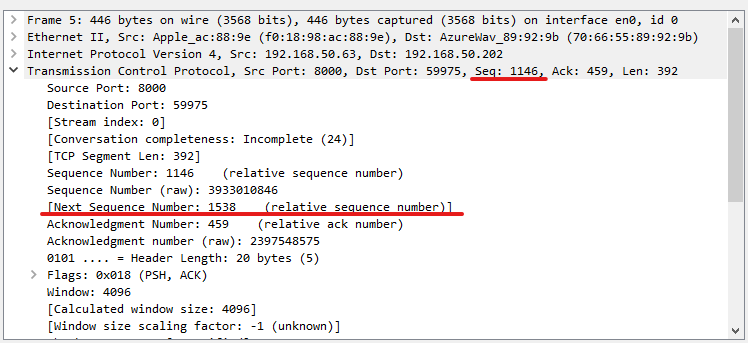
Når en avsender ønsker å sende data til en mottaker, blir dataene delt opp i en serie med pakker. Hver pakke er tildelt et sekvensnummer som indikerer posisjonen til dataene i strømmen. Når mottakeren mottar pakkene, bruker den sekvensnumrene for å sette sammen pakkene i den opprinnelige rekkefølgen.

Sekvensnummeret brukes også for å gi pålitelig levering av data. Når en pakke sendes, forventer avsenderen en bekreftelse fra mottakeren for å bekrefte at pakken er mottatt. Hvis avsenderen ikke mottar en ACK innen en viss tidsperiode, sender den pakken på nytt med samme sekvensnummer. Dette sikrer at mottakeren mottar alle pakkene i riktig rekkefølge, og at ingen data går tapt eller blir ødelagt under overføring. ACK bruker sekvensnummeret med lengden plusset sammen for å vise at det har mottatt data.

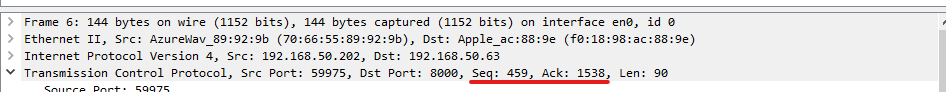
Diagram

Description automatically generated

Figure 1: Her ser vi sammenhengen mellom ack og seq nummer. Seq nummer viser den første byte og lengden kan hjelpe i å bestemme posisjonen til siste data byte. seq nummeret inkrementeres slik at man vet rekkefølgen og posisjonen til data. Tatt fra: <https://madpackets.com/2018/04/25/tcp-sequence-and-acknowledgement-numbers-explained/>



Sekvensnummeret til denne pakken er 1146, og neste sekvensnummer er 1538. Det betyr at mottakeren vet å putte pakken med sekvensnummer 1538 etter 1146 når den setter pakkene sammen igjen (siden lengden plus sekvensnummeret er like det nye sekvensnummeret). Det betyr også at hvis den ikke mottar pakken med sekvensnummer 1538 , blir den pakken sendt igjen. Kvitteringsnummer viser neste byte av sekvensnummer mottakeren forventer å motta.



Den neste pakken hadde sevensnummer 459, så da vet mottakeren at denne pakken kommer etter den som kom før.

## HTTP

HTTP står for HyperText Transfer Protocol og er brukt for å overføre data over internett. HTTP fungerer ved å sende forespørsler i form av meldinger fra klienten til serveren. Disse meldingene består vanligvis av en forespørselsmetode (for eksempel GET, POST eller PUT), en URL-adresse som identifiserer ressursen som skal hentes eller manipuleres, og andre metadata som kan brukes av serveren til å bestemme hvordan forespørselen skal behandles.

## Tråder

I programmering refererer tråder (også kjent som threads) til den minste enheten av behandling som kan planlegges av et operativsystem. En tråd er en enkel sekvens av instruksjoner som kan utføres samtidig med andre tråder i samme program. Tråder er spesielt nyttige i situasjoner der en oppgave må utføres samtidig med en annen oppgave.

## P3:

Oppgave 1, håndtere flere klienter:

Denne oppgaven går ut på å kunne lage en kalkulator klient og server som kan håndtere flere klienter samtidig. For denne oppgaven var det brukt TCP for transport av informasjon og ip + port for nettverkslag bindinger.

Først kan vi se på hvordan det er når man etablerer en kobling med serveren, her er en skjermdump fra Wireshark.

A screenshot of a computer

Description automatically generated

Figur 1: historikk av kommunikasjon mellom 1 klient og 1 server

Her vises hele historikken av server og tjener kommunikasjon. De første 3 linjer er brukt for å opprettholde bindingen med klient og tjener. Source kolonnen viser hvilken ip-addressen til sender og destination viser mottaker ip.

Graphical user interface, text, application, Word

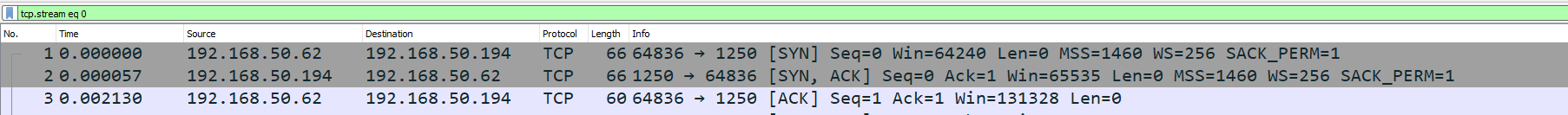
Description automatically generated

Figur 2:Her vises mac-adressen og porten til sender og mottaker som vist i Wireshark

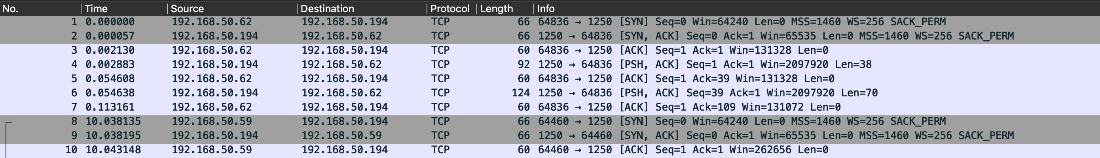
## Oppkobling

Det første TCP flagget på første linjen ([SYN]) som står for «synchronize» er en melding til serveren for å initiere en tilkoblingsforespørsel. Når SYN request-en når serveren, vil serveren respondere med en [SYN, ACK] segment tilbake til klienten. Dette er for å bekrefte til klienten at tilkoblingsforespørselen er bekreftet. Deretter må klienten sende en [ACK] segment til serveren for å bekrefte tilkoblingsforespørselen. Dette er kjent som «TCP 3-way handshake» eller «TCP 3-veis håndtrykk».

Her er en figur som viser bare den 3-way handshake delen.



Figur 3: Eksempel med enkel klient/tjener; her er 192.168.50.194 serveren og tilsvarende 192.168.50.62 er klienten.



Figur 4: Eksempel med flere samtidige klienter til en tjener; her er 192.168.59.194 serveren, mens 192.168.50.62 er klient 1 og 192.168.50.59 er klient 2.

## Overføring av data

Calendar

Description automatically generated with medium confidence

Figur 5: Overføring av data ved bruk av [PSH,ACK].

PSH-ACK-pakken i en TCP-tilkobling brukes til å indikere til senderen at mottakeren har mottatt alle dataene opp til et visst punkt i strømmen, og at mottakeren er klar til å behandle dataene. Imidlertid bekrefter ikke PSH-ACK-pakken mottakelsen av PSH-ACK-pakken selv.

Etter å ha mottatt PSH-ACK-pakken, trenger senderen å vite at mottakeren mottok PSH-ACK-pakken selv. For å gjøre dette sender mottakeren en ACK-pakke tilbake til senderen for å bekrefte at den mottok PSH-ACK-pakken. ACK-pakken bekrefter også mottakelsen av alle dataene opp til det punktet som er angitt av sekvensnummeret til PSH-ACK-pakken.

På figur 5 kan man se I den røde firkanten at hver pakke har et sekvensnummer og acknowledgment nummer. Første linje i den røde firkanten har flagget psh-ack har seq nr = 1, ack nr = 1 og lengde = 38. Klienten responderer med ack nr = 1 (seq nr som viser start byte)+ 38 (lengde til dataen) = 39. Med denne ack responsen vet serveren at hele data-en som ble sendt tidligere har blitt tatt imot.

Graphical user interface, text, application

Description automatically generated

Figur 6: her vises informasjon som blir sendt fra tjener til klient og motsatt. Dette er det som blir sendt i psh-ack pakkene. Markert med rød er klienten sin melding og blå markerer tjenerens melding

## Nedkobling

A picture containing diagram

Description automatically generated

Figur 7: Nedkobling av forbindelsen med 4 steg. Både serveren og klienten må sende et FIN segment som forventer en ACK segment tilbake

Når programmet skal avsluttes må klienten først sende et [FIN, ACK] segment til serveren og den forventer en ACK respons tilbake fra serveren. Etter serveren sender en ACK segment tilbake må den også sende et FINT segment for å vise at den er også klar til å avslutte koblingen. Klienten må deretter sende en ACK segment tilbake for å fullføre

## Flere koblinger samtidig

Table

Description automatically generated

Figur 8: Her vises en skjermdump der det er to klienter koblet med tjeneren 192.168.50.194

## Text Description automatically generated with low confidenceP3 webserver med html

Figur 9: kommunikasjon med en webtjener som returnerer en html. Her er klienten 192.168.1.249 og tjeneren er 192.168.1.249

Som vist i figuren, markerer den første blå firkanten oppkoblingen med 3VH som vanlig. Deretter markert med grønt punkt er en GET forespørsel til tjener fra klienten. Under det gule punktet ser vi status til forespørselen som er 200/OK som indikerer at det var en suksess. Selve html-en blir sendt med PSH-ACK markert med gul sirkel i figuren. Koblingen avsluttes til slutt med en Fin-ACK markert med en rød sirkel.

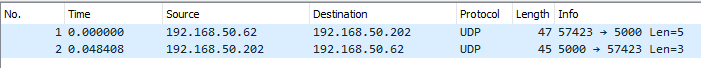
Graphical user interface, text, application

Description automatically generated

Figur 10: resultat av følg tcp-stream fra Wireshark. Den røde teksten er klienten sin forespørsel og den blåe teksten er tjenerens respons.

# P4

## UDP



Ingen oppkopling eller nedkopling

UDP er enklere og raskere en TCP. Dataene som skal overføres er delt inn i små pakker med en fast størrelse. Ingen bekreftelse eller feilretting. Dette betyr at hvis en pakke går tapt eller blir ødelagt under overføring, vil den ikke bli sendt på nytt, og mottakeren vil ikke bli varslet. Pakken blir gitt en port og en IP som applikasjonen bruker til å tildele pakken til riktig sted. Det er også viktig å nevne at UDP etablerer ikke en kobling før den sender pakker i motsetning til TCP som trenger en kobling (handshake). UDP har ikke noe form for feil håndtering. Dette gjør at UDP er en rask protokoll som brukes til å sende informasjon over nettverk.

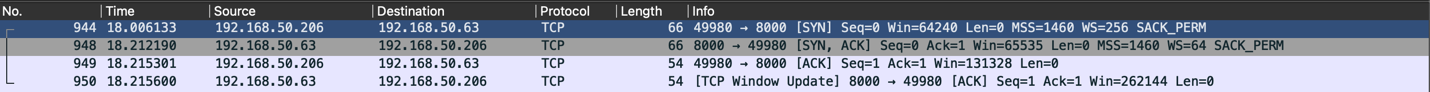
## TLS

TLS står for transport layer security og er en krypteringsprotokoll som brukes til å sikre kommunikasjon over nettverk. TLS er en etterfølger av SSL. TLS har mange versjoner og for denne øvingen har vi brukt TLS v1.3 siden det er alltid lurt å bruke nyeste versjon som skal være mindre sårbar enn tidligere versjoner.

Virkemåte: TLS bruker kryptering til å etablere en sikker kobling mellom tjener og klient. Her er 5 steger for å etablere en sikker kobling

1. Client Hello: klienten sender en melding til serveren som har med seg informasjon om blant annet hvilken TLS-versjon den bruker og krypterings metoder den støtter.
2. Server Hello: Serveren responderer med hvilken TLS versjon og krypterings metode som skal brukes for koblingen.
3. Sertifikat utveksling: Serveren sender sertifikatet til klienten. Serverens Public Key blir også sendt
4. Client Key Exchange: klienten lager en sesjons nøkkel som krypteres ved bruk av serverens Public Key fra forrige steg og sendes tilbake til serveren.
5. Symmetrisk kryptering: når serveren får sesjons nøkkelen fra klienten kan denne nøkkelen brukes til å kryptere og dekryptere all informasjon sendt mellom dem for resten av sesjonen.

Oppkobling TCP



Figur 11 TCP 3VH når klient kobler seg til tjeneren. Klient-ip: 192.168.50.206, tjener-ip: 192.168.50.63

Her kan vi se 3VH gjennom TCP når klienten (102.168.50.206) kobler seg til tjeneren (192.168.50.63). Klienten sender først en SYN melding, før tjeneren sender en SYN, ACK melding, for å vise til at den anerkjenner oppkoblingen. Så sender klienten en ACK melding til tjeneren igjen, for å vise til at den anerkjenner at serveren har anerkjent oppkoblingen. Til slutt sender tjeneren en beskjed til klienten, at bufferen er klar til å motta data, gjennom TCP Window Update.

Oppkobling TLS & TCP (Når første pakke blir sendt fra klient)

Et bilde som inneholder tekst

Automatisk generert beskrivelse

Figur 12 TCP blir brukt til å anerkjenne meldingene sendt frem og tilbake mellom klient og server under oppkoblingen til TLS

Etter koblingen er anerkjent gjennom TCP, så starter oppkoblingen til TLS når klienten sender sin første melding. TCP vil anerkjenne de forskjellige dataene som blir sendt gjennom TLS sin oppkobling.

TLS Oppkobling forklart:

1. Client Hello: klienten sender en melding til serveren som har med seg informasjon om blant annet hvilken TLS-versjon den bruker og krypterings metoder den støtter.
2. Server Hello: Serveren responderer med hvilken tls versjon og krypterings metode som skal brukes for koblingen.
3. Certificate exchange: Serveren sender sertifikatet til klienten. Serverens Public Key blir også sendt
4. Client key exchange: klienten lager en sesjons nøkkel som krypteres ved bruk av serverens public key fra førrige steg og sendes tilbake til serveren.
5. Symmetrisk kryptering: når serveren får sesjons nøkkelen fra klienten kan denne nøkkelen brukes til å kryptere og dekryptere all informasjon sendt mellom dem for resten av sesjonen.

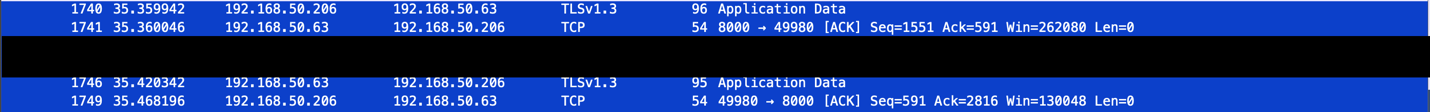


Figur 13 Kryptografi suite og sesjonsnøkkel under “Server Hello”, som gir beskjed til klienten hvilken kryptografi suite som er i bruk, og hvilken sesjonsnøkkel som brukes.

Sertifikat:

Gjennom samtale med student assistent, har vi kommet frem til at versjonen vi bruker av TLS, som er TLS v1.3, krypterer sertifikatet inn under application data, som gjør det så og si umulig å hente ut. Derfor har vi ikke fått frem nøyaktig hvilket sertifikat tjeneren identifiserer seg med. Hvis vi skulle ha hentet dette ut, måtte vi ha dekryptert Application Data, for så å dekryptere sertifikatet.

Overføring av data TCP og TLS

Figur 14 Overføring av data når en melding blir sendt mellom klient og tjener.

Det som skjer her, er at klienten (192.168.50.206) sender kryptert data til tjeneren (192.168.50.63) gjennom TLS. Tjeneren sender en ACK melding gjennom TCP på at den har mottatt dataen. Så sender tjeneren kryptert data tilbake til klienten, før klienten sender en ACK melding på at den har mottatt dataen fra tjeneren.



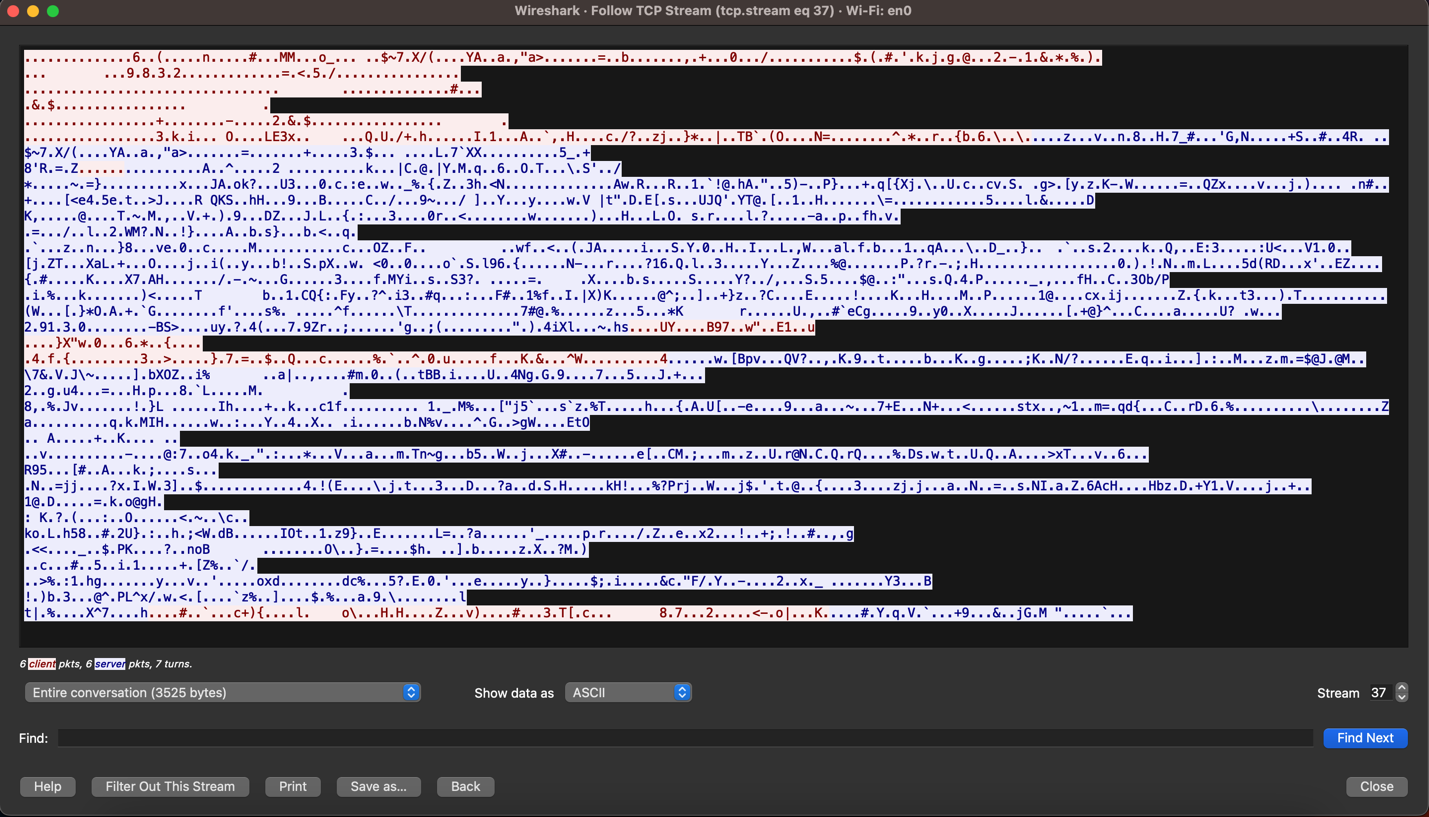
Figur 15 Lengde på Application Data fra klient. Se figur 14 for ACK melding.

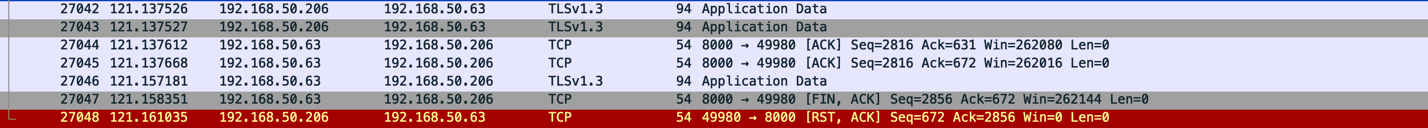


Figur 16 Lengde på Application Data fra server. Se figur 14 for ACK melding.

Et bilde som inneholder tekst, overvåke, skjerm, skjermbilde

Automatisk generert beskrivelseFigur 17 Kryptert applikasjonssdata som blir sendt fra klient til tjener under overføring av data gjennom TLS

Figur 18 Kryptert data som TCP anerkjenner, tilhørende TLS, som vist I bildet ovenfor. Hentet ut gjennom å følge TCP stream.

Nedkobling TCP og TLS

Figur Nedkoblingsdel av programmet

I nedkobling med TCP og TLS, sender klienten en finish message (FIN) til serveren. Serveren sender en ACK tilbake for å vise at den har mottatt FIN meldingen.