IKN Journal 1: Øvelse 3, 6 og 7

Gruppe: 50

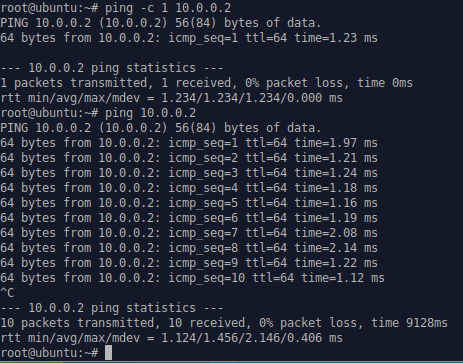
|  |  |
| --- | --- |
| Navn | Studienummer |
| Fatima Kodro | 201609565 |
| Søren Bech | 201604784 |
| Daniel Pat Hansen | 201601915 |

Øvelse 3

**1+2.       Mål den tid der går fra en ping kommando startes i H1 til ping-respons fra H2 modtages i H1. Mål minimum-/maksimum-/gennemsnits-forsinkelsestider og standardafvigelsen for 10 på hinanden følgende ping-kommandoer**

På Figur 1 ses terminalen efter ping er blevet kaldt en gang. Dette er 1,23 ms. Efter der er blevet hentet 10 packets kan min/avg/max aflæses. Dette aflæses som:

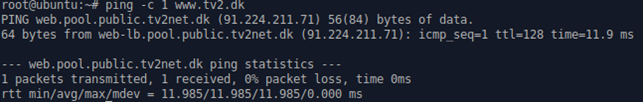
* Min: 1,124
* Avg: 1,456
* Max: 2,146



Figur 1 Ping kommando fra H1 til H2

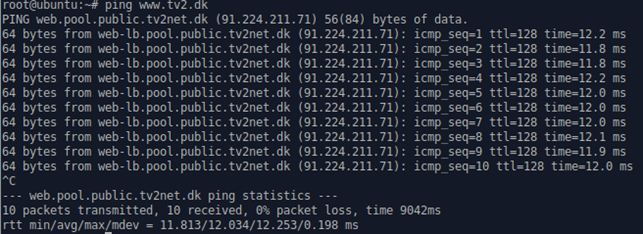
**3.       Mål den tid der går fra kommandoen ping –c 1 www.tv2.dk startes i H1 til ping-respons fra web-serveren www.tv2.dk modtages i H1.**

På figur XXX ses terminalen efter der bliver pinget til [www.tv2.dk](http://www.tv2.dk). Det ses at tiden er lig med 11.9 ms



**4.       Mål minimum-/maksimum-/gennemsnits-forsinkelsestider og standardafvigelsen for 10 på hinanden følgende ping-kommandoer, udført som i punkt 3.**

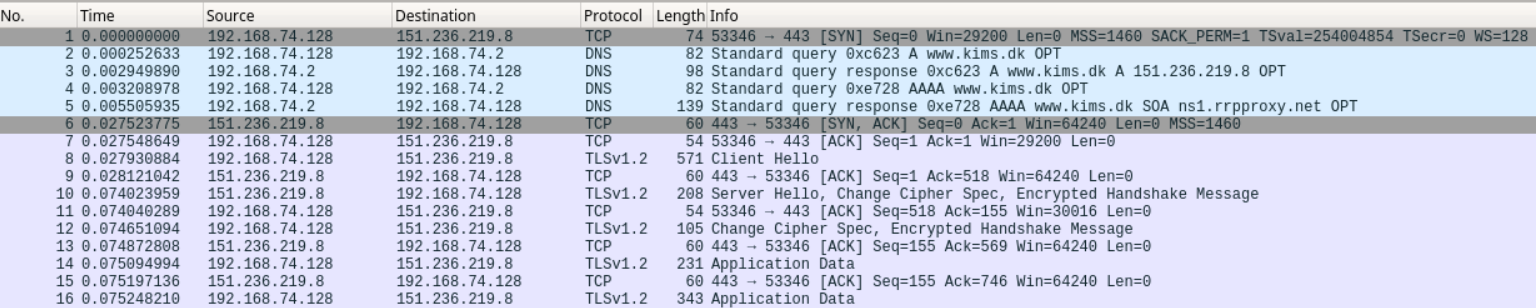
Figur XXX viser minimum/gennemsnit/maksimum efter der bliver pinget til [www.tv2.dk](http://www.tv2.dk). Det ses at der vises 11.813/12.034/12.253 ms



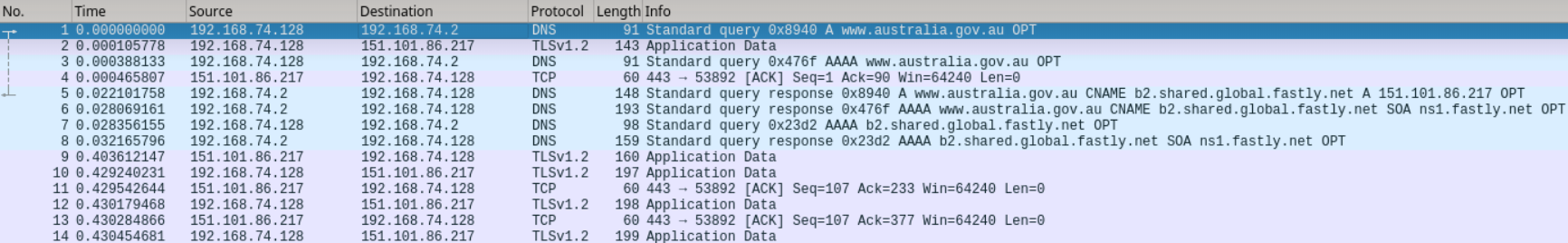
**5.       Mål den tid der går fra en web-side i en dansk web-server ønskes modtaget i H1, til web-serveren responderer:**

Der bruges [www.kims.dk](http://www.kims.dk/) som hjemmeside

På figur XXX ses et udsnit af wireshark, hvor den første DNS er en forespørgsel efter kims.dk. Herefter modtages respons fra kims.dk.



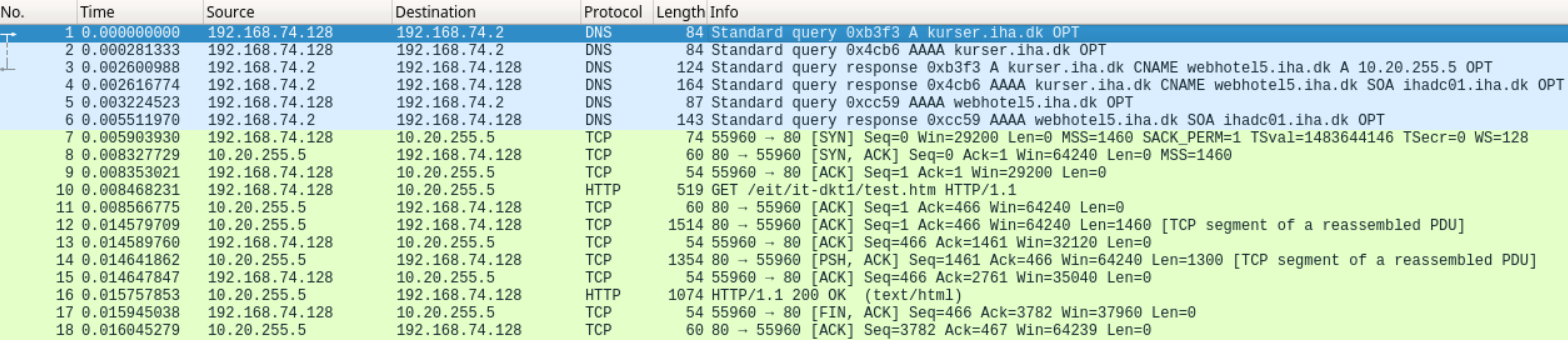
**6.       Mål vha. samme målemetode den tid der går fra en australsk web-side (som er tilfældigt valgt) ønskes modtaget i H1 web-serveren, til web-serveren responderer.**



**7.       Er der forskel på tidsforsinkelses-målingerne i punkt 5 og punkt 6? Hvis der er forskel, hvad kan årsagen være til at der denne forskel?**

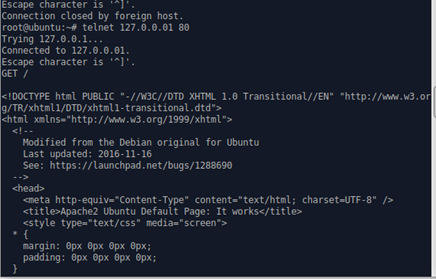
Det tager længere tid at anmode om at hente en pakke ved den australske hjemmeside. Dette er grundet at den australske server er længere væk end den danske server.

**8.       Undersøg vha. wireshark hvad der sker, når denne web-side hentes vha. en Web Browser:**

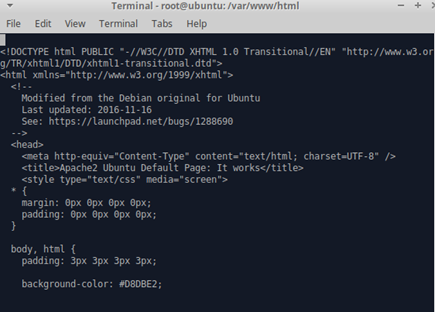


**9.       Installer en web-server (apache, nyeste stabile version) i din virtuelle maskine (H1).**

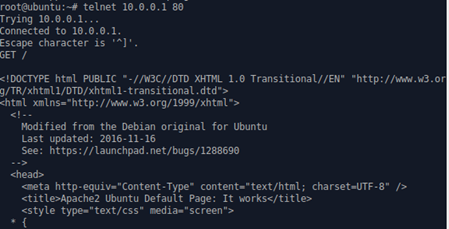
**10.   Etabler en LAN-forbindelse mellem en web-serveren (H1) og en web-client (H2) I første omgang anvendes en simpel, telnet baseret web-client i H2.**



Figur 1: H1 - GET /



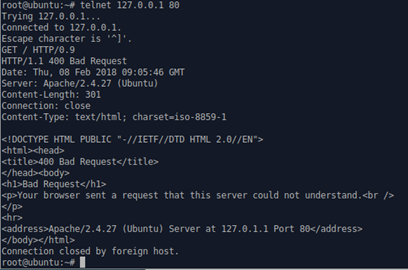
Figur 2: H1 - /var/www/html



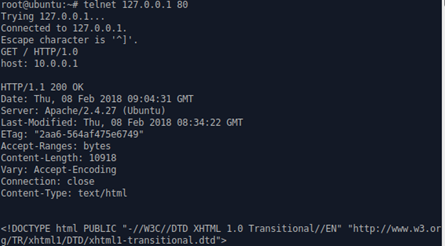
Figur 3: H2 – telnet 10.0.0.1 80

**11.   Test protokollerne: HTTP 0.9, HTTP 1.0 og HTTP 1.1.vha. telnet med fokus på oprettelse/nedlukning af TCP-connection og på persistent/non-persistent HTTP-kommunikation vha. HTTP-protokollen (uden/med pipelining).**

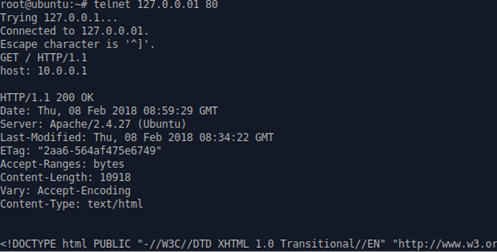
HTTP 0.9:

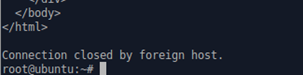
****

HTTP 1.0:

****

HTTP 1.1:

****

****

**Spørgsmål:**

**Lukkes TCP-forbindelsen straks når HTTP 0.9 anvendes?**

Ja

**Lukkes TCP-forbindelsen straks når HTTP 1.0 anvendes?**

Ja

**Lukkes TCP-forbindelsen straks når HTTP 1.1 anvendes?**

Nej, der går få sekunder

**Hvis TCP-forbindelsen ikke lukkes umiddelbart, lukkes den så automatisk lidt senere? Hvor lang tid går der?**

Den lukkes få sekunder efter.

**Hvad er fordelen ved at nedlukningen af TCP-forbindelsen udskydes?**

Der kan sendes flere pakker efter hinanden

**Er det Web-server eller WEB-client, der lukker TCP-forbindelsen?**

Web-server

**Hvilken version af Apache serveren anvendes?**

1.1

12.   Anvend Firefox web-browser som web-client i H2 sammen med Apache web-server i H1

Øvelse 6

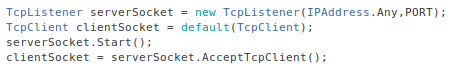
Opgaveformulering

Der bruges 2 virtuelle maskiner. En maskine, der fungerer som en server, og en anden maskine, der fungerer som en klient. Serveren har det ansvar at skulle kunne sende filer af vilkårlige typer/størrelser til klienten, ud fra en forespørgsel fra klienten. Forespørgslen foregår således at klienten sender en tekststreng, der udskrives på terminalen, til serveren. Denne tekststreng skal indeholde et filnavn + en eventuel stiangivelse til en fil, der eksisterer i serveren.

Serveren skal sende filen i segmenter på max. 1000 bytes ad gangen indtil filen er overført, hvis filen altså eksisterer, ellers skal der udskrives en fejlmeddelelse. Uanset om filen eksisterer eller ej, skal serveren kunne håndtere en ny forespørgsel, når den er klar på til at sende en ny fil, dvs. den skal være iterativ.

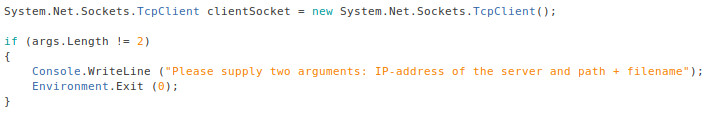
Kodeforklaring

**Server**

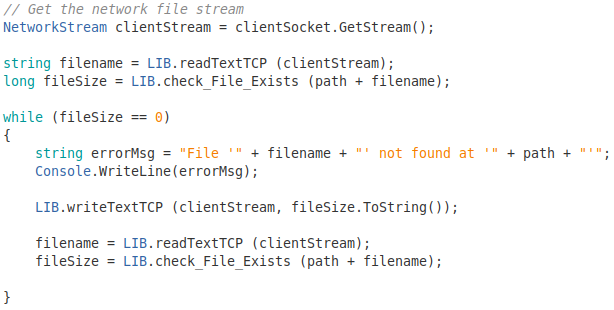
Forløbet er udført således, at serveren starter med at bruge klassen TcpListener til at lytte efter en TCP-forbindelse fra klienten som kan ses på følgende kodeudsnit: 

Serveren åbner en forbindelse med en port defineret som 9000, og venter herefter på en klient. Dette kald er blokerende indtil en klient forbindes til serveren.

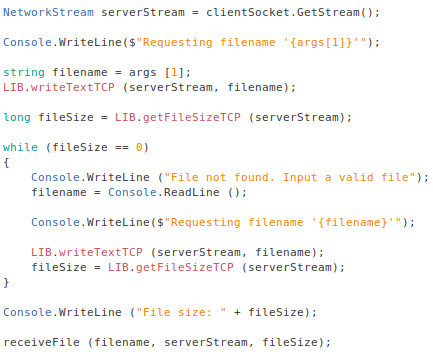
Følgende kode er på klientens side, hvor der oprettes en forbindelse til serveren:



Næste kodeudsnit viser serveren som læser en tekst streng fra klienten, og herefter tjekker om filen findes. Hvis den findes, vil filstørrelsen returneres, ellers vil 0 returneres. Returneres 0, vil serveren forespørge klienten om et andet filnavn:



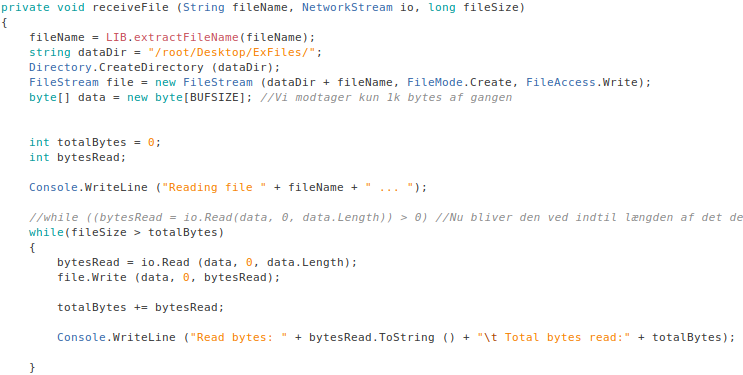
Følgende kodeudsnit er på klientens side, og viser at klienten sender en tekststreng til at forespørge om et filnavn. Hvis filen ikke findes, bliver det muligt at sende en ny forespørgsel:



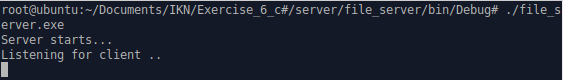
Følgende kodeudsnit viser der, hvor serveren sender filen til klienten. Filnavnet og størrelsen registreres og det gøres derefter klar til at blive sendt til klienten. BUFSIZE er lig med 1000, og inde i while-loopet sørges der for, at der hele tiden bliver sendt 1000 bytes ad gangen, indtil der til sidst er mindre end 1000 bytes tilbage at blive sendt. Her bliver de sidste bytes sendt og da der nu er 0 bytes tilbage, kører programmet ud af loopet:



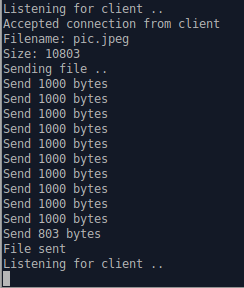
Følgende kode viser der, hvor klienten modtager filen fra serveren. Klienten anmoder om et filnavn, der findes i serveren, og herefter bliver størrelsen af filen registreret. Klienten sørger for at oprette et directory til at opretholde filen. Klienten skal kunne modtage 1000 bytes ad gangen fra serveren, og inde i while-loopet, bliver der sendt disse 1000 bytes, indtil der ikke er flere bytes at sende.



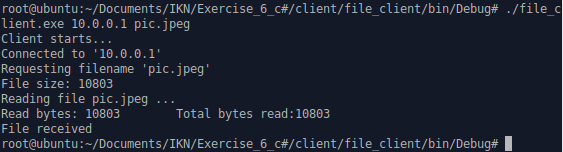
Resultater

På figur XXX ses at serveren venter på en TCP-connection fra klienten.

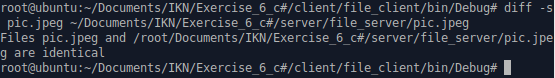
Figur XXX viser overførslen af en fil fra serveren. Filen er en jpeg-format på ~11 kb. Det ses at der sendes 1000 bytes ad gangen, indtil der til sidst kun er 803 bytes tilbage af filen.



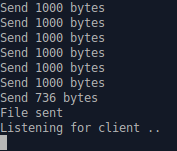
På figur XXX ses den måde klienten modtager denne jpeg fil på ~11kb. Øverst i terminalen anmoder klienten denne “pic.jpeg” fra serveren som er 10.0.0.1. Herefter sender overføres filen fra serveren og der ses at klienten til sidst modtager denne fil.



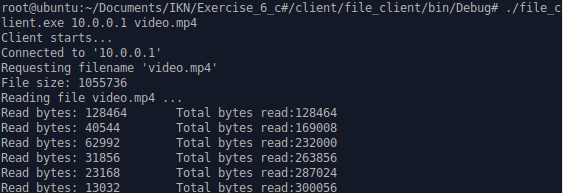
Som kvalitetskontrol, bliver der sikret, at den sendte fil og den modtagende fil er ens. Her bruges diff-kommandoen i terminalen som ses på figur XXX til at tjekke om de to filer er ens. Der ses her at der bliver udskrevet “are identical”, hvilket vil sige at de to filer er identiske.



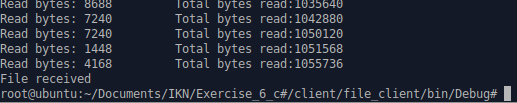
På figur XXX bliver der sendt en mp4-fil på ~1mb til klienten fra serveren. Dette bliver overført på samme måde som jpeg-filen, nemlig 1000 bytes ad gangen.



På figur XXX ses starten af, hvordan klienten modtager mp4-filen som blev sendt fra serveren.

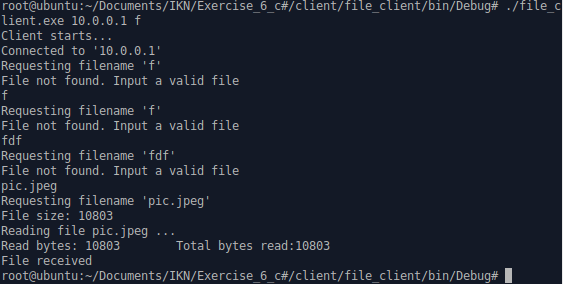


På figur XXX ses der, hvor klienten er færdig med at modtage mp4-filen som blev sendt fra serveren.

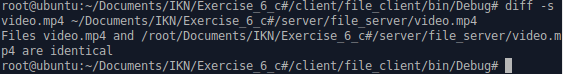


På figur XXX ses serveren, hvor den prøver at sende en fil der ikke findes, hvorefter det bliver muligt at sende en ny fil. Efter der er blevet skrevet et rigtigt filnavn, nemlig pic.jpeg, bliver filen overført til klienten, og derefter lytter den igen til klienten. Dette er fordi serveren er iterativ, så den bliver hele tiden ved med at lytte på forespørgsler fra klienten efter hver filoverførsel indtil serveren lukkes ned.



På figur XXX ses klienten, hvor den anmoder om ikke eksisterende filer. Derefter bliver det muligt for den igen at lave en forespørgsel. Selv efter den rigtige fil er blevet overført, er det muligt for klienten at sende en ny forespørgsel.

På figur XXX laves der en kvalitetskontrol, for at tjekke, om de to filer er identiske. Der bliver derefter skrevet “are identical” på terminalen, hvilken betyder at begge filer er helt identiske som forventet.



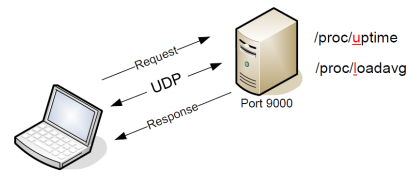
Øvelse 7

Opgaveformulering

Der bruges 2 virtuelle maskiner. En maskine bruges som server og en anden maskine bruges som klient.

Serveren har det ansvar, at skulle kunne modtage en forespørgsel fra klienten, bestående af enten et “U” eller “L”. Hvis serveren modtager et “U” skal den returnere /proc/uptime tilbage til klienten, som indeholder information om den samlede til, serveren har været kørende siden start. Hvis serveren modtager et “L” skal den returnere /proc/loadavg, som indeholder information om serverens aktuelle CPU-load.

Klienten skal kunne sende “u”, “U”, “l” eller “L” til UDP-serveren, og derefter modtage det som serveren returnere, som nævnt tidligere.



Kodeforklaring

Resultater