

IKN Obl. AFl 1

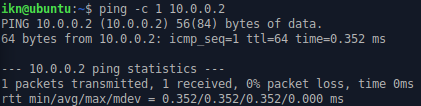
Gruppe 12 – 05-04-2018



|  |  |
| --- | --- |
| Navn | Studienr |
| Frederik Kastrup Mortensen | **201607221** |
| Søren Schou Mathiasen | 201605264 |
| Martin Haugaard Andersen | 201605036 |
| Alexander Lichtenstein Davidsen | 201608479 |

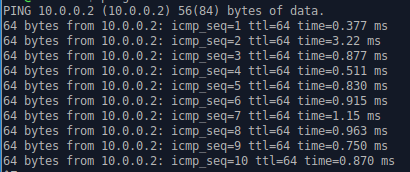
# Øvelse 3

1. Mål den tid der går fra en ping kommando startes i H1 til ping-respons fra H2 modtages i H1.

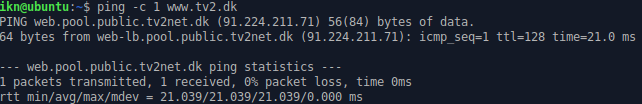


1. Mål minimum-/maksimum-/gennemsnits-forsinkelsestider og standardafvigelsen for 10 på hinanden følgende ping-kommandoer, udført som i punkt 1.

Minimum = 0,377ms  
Maksimum = 3,22ms  
Gennemsnit = 1,0463ms

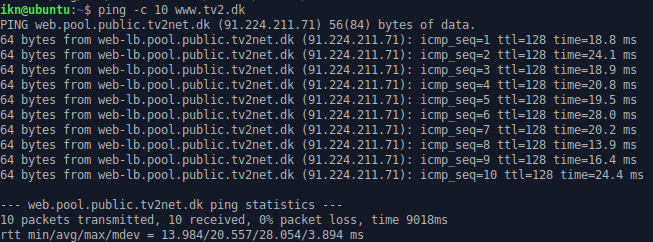


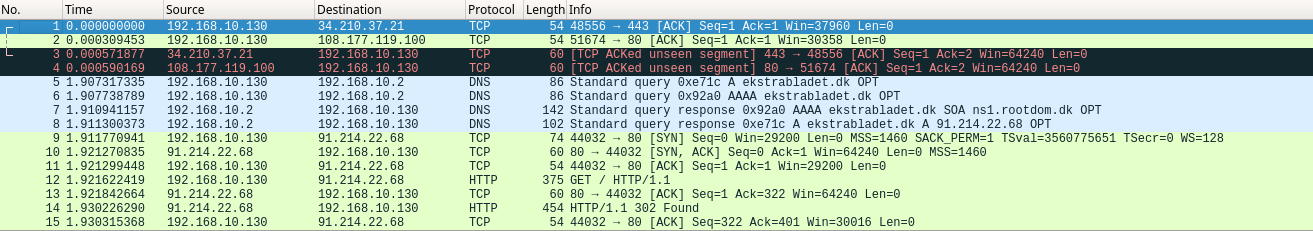
1. Mål den tid der går fra kommandoen ping –c 1 www.tv2.dk startes i H1 til ping-respons fra web-serveren www.tv2.dk modtages i H

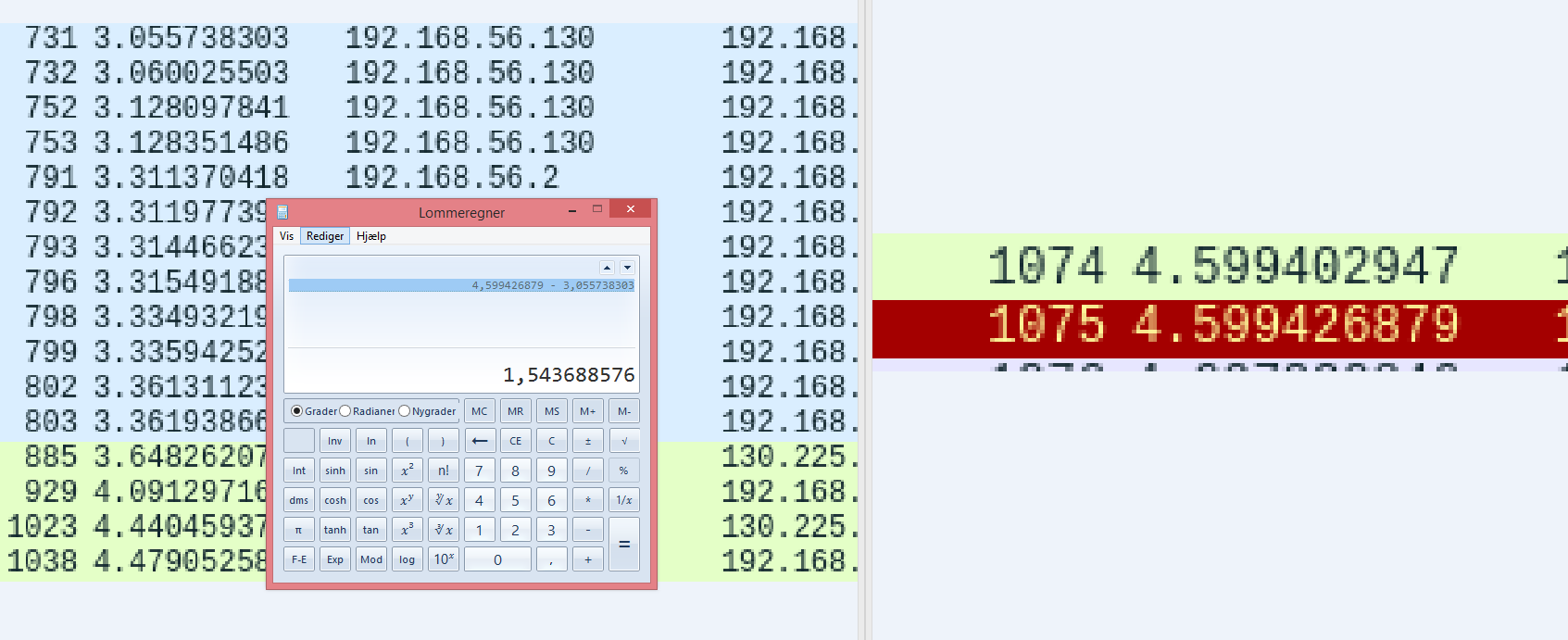


1. Mål minimum-/maksimum-/gennemsnits-forsinkelsestider og standardafvigelsen for 10 på hinanden følgende ping-kommandoer, udført som i punkt 3.

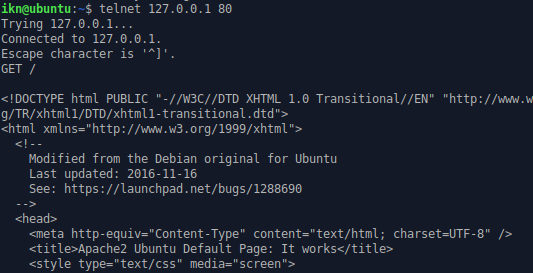
Minimum = 13,984ms

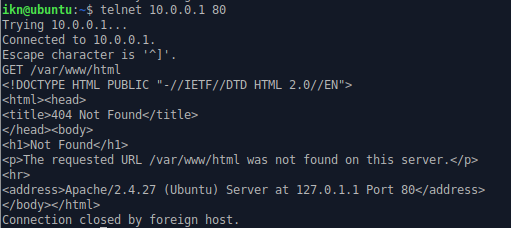
Maksimum = 28,054ms  
Gennemsnit = 20,557ms  


1. Mål den tid der går fra en web-side i en dansk web-server ønskes modtaget i H1, til web-serveren responderer. Der er her brugt websiden Ekstrabladet.dk.   
   Fra no 5 til no 9. tager det 0,01389ms.   
   
2. 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. Anvend din Web Browser som client. Webserveren, som indeholder den australske web-side, befinder sig sandsynligvis fysisk meget længere væk end den danske web-sides web-server gør.
3. 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?
4. Undersøg vha. wireshark hvad der sker, når denne web-side hentes vha. en Web Browser:   
   kurser.iha.dk/eit/it-dkt1/test.htm  
   Klienten sender kurser.iha.dk til en DNS server. Herefter får vi så IP-adressen tilbage som er 130.225.184.5. Herefter sendes der et http get request, som giver et html respons. 



1. Vi etablerer en LAN-forbindelse mellem en apache webserver og en web klient. Her vises lokalforbindelsen fra H1. Der bedes om at udskrive en tekst på skærmen svarende til indholdet af en html fil.

  
Forbindelse fra H2 til H1 server.



11:

Test protokollerne: HTTP 0.9, HTTP 1.0 og HTTP 1.1.vha. telnet med fokus på oprettelse/nedlukning af TCP-connect

|  |
| --- |
|  |
|  | GET / |
|  | Closed by foreign host med det samme |
|  |  |
|  | GET / HTTP/1.0 \*2 gange enter\* |
|  | Closed by foreign host med det samme |
|  |  |
|  | GET / HTTP/1.1 |
|  | Host:10.0.0.1 \*2 gange enter\* |
|  | Closed by foreign host efter noget tid (omkring 5 sekunder). Dermed kan der sendes flere requests. For at finde ud af hvilken version af Apache serveren benytter, skrives der ”Apache2 -v”.  https://cdn.discordapp.com/attachments/431151379952959490/431157985977565195/unknown.png |

12. Der skal anvendes Firefox web browser som web-klient i H2 sammen med en Apache web-server i H1. Der tilføjes 3 billeder til web-serveren, som herefter vises i web-browseren.

Der er først handshaking mellem H1 og H2.

Derefter:

H2: GET / HTTP1.1

H1: HTTP/1.1 200 OK Indeholder response to GET kaldet. I dette er der 247 bytes i gzip format, som bliver til 411 bytes efter udpakning. Disse bytes er html teksten fra siden, og denne indeholder source navnet til alle tre billeder.

H2: Acknowledger at svaret er modtaget, og ser derefter at der er tre billeder

H2: GET /Hotdog1.jpg HTTP/1.1

H1 og H2: ACK og SYN

H2: GET /Hotdog2.jpg HTTP/1.1

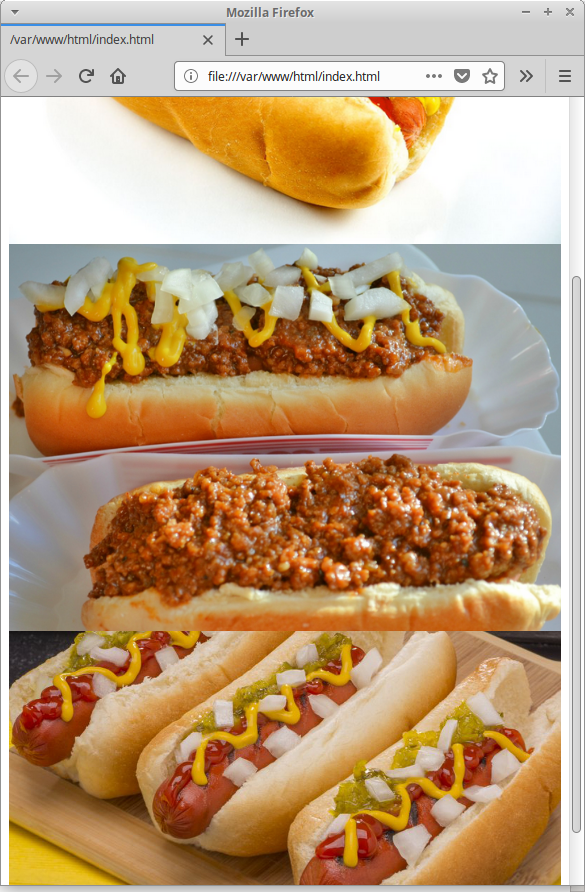
H1 og H2: ACK og SYN

H2: GET /Hotdog3.jpg HTTP/1.1

H1 og H2: ACK og SYN

Derefter sendes billederne over mange packets.

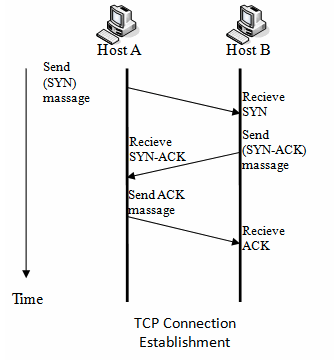
Cached:

Hvis billederne allerede er blevet overført til H2 en gang, og er i cache, sender H1 i stedet "Not modified" så billederne ikke sendes igen. Dette sker for hvert billede.

# Øvelse 6

Øvelsen består af et server/klient system, hvor en klient kan anmode om en fil, hvorved serveren sender filen tilbage. Dette skal foregå ved hjælp af TCP-protokollen.

## TCP

TCP står for transmission control protocol. Den bruges når der skal overføres data på en pålidelig måde, hvor hver eneste bit er vigtigt. Der tjekkes for fejl i transmissionen, så hvis data går tabt under transmissionen, vil det blive sendt igen. For at dette kan fungere, skal to værts applikationer skabe forbindelse til hinanden. Dette foregår gennem en handshake procedure der forekommer i transportlaget.

## Server

Der er i denne del af øvelsen udviklet en server, der skal modtage en tekststreng fra en klient. Tekststrengen indeholder et filnavn, der beskriver hvilken fil, klienten ønsker at modtage samt serverens IP adresse. Hvis filen ikke eksisterer, vil klienten få en fejlbesked. Hvis filen herimod eksisterer, vil serveren sender filen til klienten i bider af max 1000 bytes indtil hele filen er sendt.

## Klient

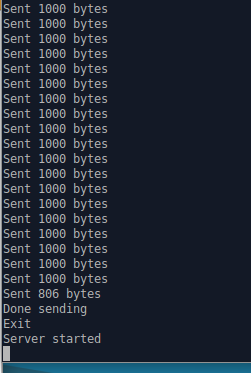
Klienten sender indledningsvis en tekststreng der indeholder et filnavn og serverens IP-adresse. Klienten modtager filens størrelse hvorefter klienten modtager filen fejlfrit fra serveren.

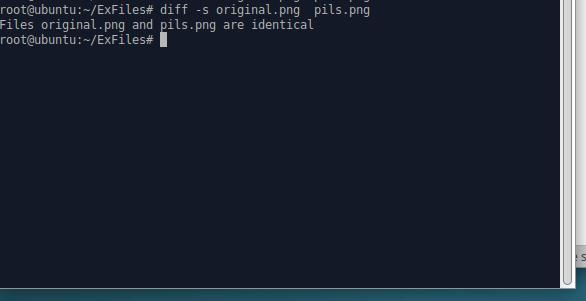
## Resultat

Der er sendt en jpg fil fra serveren til klienten. Den sendte fil sammenlignes med originalen, ved brug af diff funktionen.

Det ses her at klienten anmoder om at modtage filen kaldet ”pils.png”. Herefter modtager klienten filstørrelsen og begynder at modtage filen fra serveren. I nederste del af billedet ses det at klienten har modtaget filen ”pils.png” og gemt den i ”ExFiles” mappen.

## 

Her ses fil-transaktionen fra serverens synspunkt. Her kan det ses at serveren sender i pakker af 1000 bytes og at serveren fungerer iterativt. Efter serveren er færdig med at sende, er den her klar til at sende endnu en fil, som det ses i bunden af billedet.   
  
På nedenstående billede, sammenlignes den sendte fil med originalen, ved brug af diff funktionen. Det meddeles her at filerne er identiske.

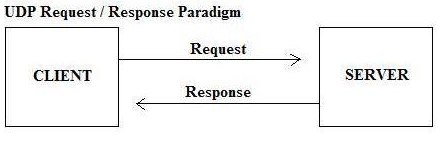


# Øvelse 7

Der skal udvikles et server-klient system der kan kommunikere gennem UDP protokollen.

UDP

UDP står for user datagram protokol. Denne protokol gør computer applikationer i stand til at sende beskeder til hinanden, uden tidligere kommunikation. Med andre ord, der er ikke noget handshake mellem to værter. Det er derfor ikke nødvendigt at opsætte kommunikationskanaler eller datastrømme. Dette betyder også at der ingen garanti er for, at beskeden når frem til klienten og omvendt.



## Server

I denne del af øvelsen skal der designes og implementeres en UDP-server med support for en klient ad gangen.

Serveren skal kunne håndtere beskeder der sendes fra en klient. Hvis serveren modtager et ”U”, skal informationen i filen /proc/uptime returneres til klienten.  
Hvis serveren modtager et ”L”, skal informationen fra /proc/loadavg returneres til klienten. Serveren skal kunne håndtere både store og små bogstaver fra klienten.

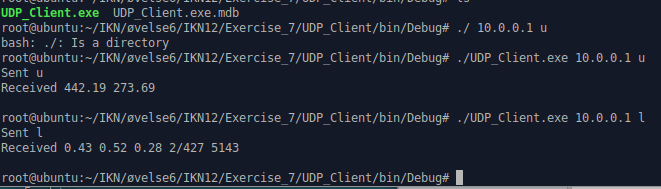
Serveren indlæser fra en af de filer, alt efter input fra klienten, i en string som derefter sender over til klienten gennem en UDP socket. Der er ingen garanti for at klienten modtager beskeden og klienten er måske ikke engang forbundet til serveren, når beskeden sendes.

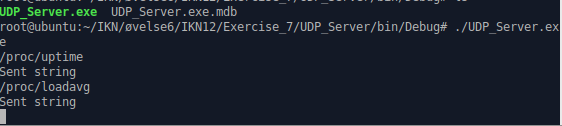
## Client

Der skal her designes og implementeres en UDP-klient, som kan sende en besked til en server. Beskeden skal være ”U”, ”u”, ”L” eller ”l”, svaret fra serveren skal her udskrives til operatøren af klienten.

## Resultater

Nedenunder ses det at der fra klienten sendes et **u** og et **l** til serveren og serveren sender hermed indholdet af /proc/uptime og /proc/loadavg tilbage til klienten henholdsvis.





Nedenunder ses det at hvis der sendes et stort U eller L, vil serveren stadig sende det rigtige information tilbage. Der sendes også et K, som serveren ignorerer.

