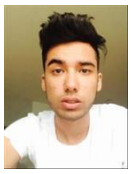
Projektnavn: CDIO 2
Afleveringsfrist: *Tirsdag d. 14/4 - 2015*
Denne rapport er afleveret via Campusnet (der skrives ikke under)
Denne rapport indeholder sider inkl. denne side.

Studienummer: s144839
Navn: Liban Jama Awil Dirir

Studienummer: s144859
Navn: Kenneth Skovsgaard Frandsen



Studienummer: s144841
Navn: Nael Rashdeen

Studienummer: s144849
Navn: Delvaux Paulin Richard Niyonzima

Studienummer: s133275
Navn: Peter Asmussen

Studienummer: s123503
Navn: Thomas Liljegren

Studienummer: s123172
Navn: Martin Vieth

Indledning

CDIO 2 handler om at lave to programmer der hver skal kunne to ting. Det første program skal være en FTP-klient der kan håndtere at sende en fil, samt at sende kommandoer og modtage svar fra et ZYBO-board. (Vi har i vores projekt brugt en FTP på Mac, blandet med FTP på ZYBO)
Det andet program skal have muligheden for, at forbinde til en vægt eller en simulator. Idet at der skal hentes en fil fra en FTP-server og sendes kommandoer, er der blevet lavet en menu hvor hvor man som bruger har muligheder for at samarbejde med programmet.
Der er fra et kundesynspunkt blevet lavet en afvejnings procedure, hvor man skal identificere sig som operatør, hvorefter man kan foretage afvejninger af et produkt valgt af operatøren ud fra en fil som hentes fra FTP serveren.
Derudover skal man kunne dokumentere de afvejninger der er blevet foretaget med vægten og gemme dem i en log, idet at der kan opstå fejl, så man senere kan backtrace, hvem der har lavet dem og hvornår, da loggen vil indeholde tidspunkt og id på operatøren.

Kravspecifikation

Der skal laves to programmer. Det første program skal være en FTP- klient der kan håndtere, at sende en fil, samt at sende kommandoer og modtage svar fra en FTP server. Det andet program skal have egenskaben i, at forbinde til en vægt eller en simulator. FTP serveren kunne køre på Zybo board.

Systemet skal overholde følgende krav:

- Sende en kommando til FTP server.
- Overfører en fil fra FTP serveren til en indtastet sti.
- Brugeren/operatøren skal kunne identificere sig.
- Brugeren/operatøren skal kunne angive varenummer.
- Man skal kunne opslå varenumre i databasen, disse skal kunne udskrives.
- Brugeren/operatøren skal verificere produktet
- Man skal kunne tarére
- Man skal kunne opfylde varer
- Man skal kunne afmåle nettovægten
- Man skal kunne foretage brutto kontrol.

Evt udbygning

- Man skal kunne afskrive forbrugte råvarer i databasen, og indføre afvejningen i loggen

Wireshark

Wireshark er et open source netværkspakke analyseprogram. Hovedsageligt bruges det til netværksfejlfinding, analyse, samt software protokol udvikling. Derudover er det verdens største netværks protokol analysator. Wireshark fungerer ved, at fange netværkspakker og forsøger, at vise de pakker så detaljeret som muligt. Rettere sagt, kan det kan ses som et måleprogram der analyserer hvad der foregår i et netværk.

Det er nyttigt i mange situationer, og nogle af de hyppigste eksempler er vist herunder:

- Netværks administratorer bruger det til, at fejlfinde netværksproblemer.
- Netværks sikkerheds ingeniører bruger det til, at måle sikkerhedsproblemer.
- Udviklere bruger det til at debugge protokol implementeringer.
- Folk bruger det til, at forstå interne netværksprotokoller. [1]

Vi har brugt wireshark til at se på eksisterende FTP kommunikation mellem en server og klienten. Dette gjorde vi for at se hvilke beskeder blev sendt samt hvilke respons som sendes tilbage.
Vi har brugt dette til at sikre os at vores FTP klient sender de rigtige beskeder i den rigtige rækkefølge.
Vi har vedlagt en wireshark fil hvor man kan se kommunikationen mellem FTP server og FTP klient.

[1] https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html

Implementering:

getPASV() tager i mod svaret fra PASV kommandoen, og udregner porten til den nye data socket.

```
public void getPASV(){
    if(getEar().getLine().contains("227 Entering")){
        Main.setActive(false);
        PASV = getEar().getLine().substring(getEar().getLine().indexOf('(')+1, getEar().getLine().indexOf(')'));
        String[] box = PASV.split(",");
        portNumber = Integer.parseInt(box[4])*256 +Integer.parseInt(box[5]);
        System.out.println(portNumber);
    }
}
```

PASV kommando[2]

PASV kommandoen returnerer (h1,h2,h3,h4,p1,p2) som er en ip og en port:
Ip delen er (h1,h2,h3,h4) og (p1,p2) er porten.
Det hele er skrevet 8bit, derfor er de 2 tal der udgør porten ikke brugbare endnu.
Den kan dog beregnes med denne formel $(p1 * 256) + p2$ som laver det om til 16 bit.
fx (192.168.2.2,221,51) derved kan man se at ipen er 192.168.2.2.

Porten bliver udregnet ved man ganger $(221 * 256) + 51 = 56627$.
Denne port skal så bruges for at der kan oprettes forbindelse til serveren da det er den port data bliver kommunikeret igennem.

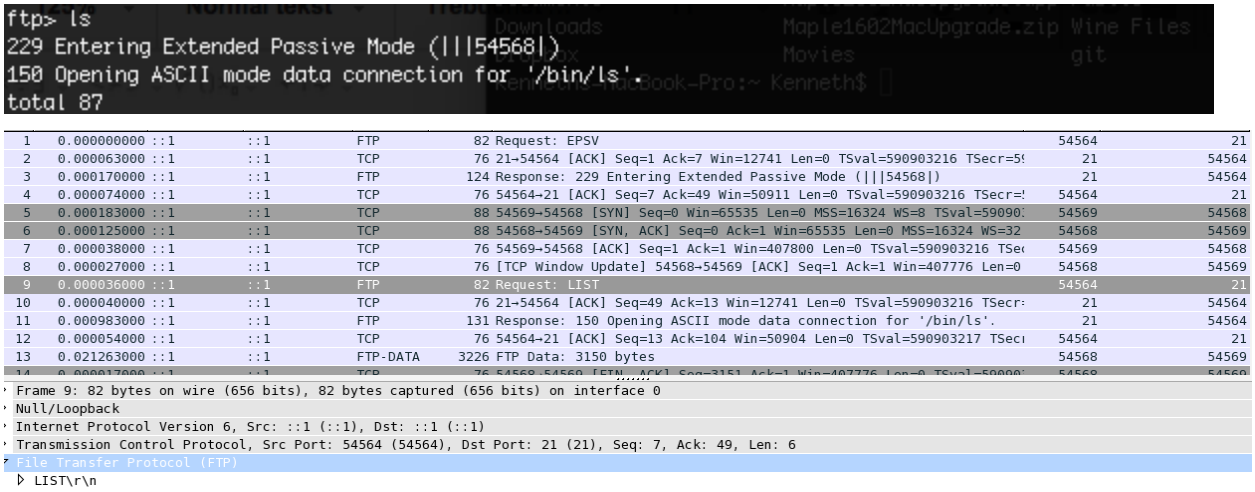
LIST kommando[3]

LIST kommando bruges til at vise hvilket filer der er på serveren i et bestemt mappe(sti).
Når kommandoen sendes til en bestemt mappe returnerer den en liste over hvilke filer og mapper som mappen indeholder.
LIST kan også bruges til at få tilsendt specifikke oplysninger om enkelte filer, ved at skrive den relative eller komplette sti til filen.
LIST sender altid sit svar kommando tilbage i ASCII. (TYPE A)

Igennem UNIX-terminalen virker det ikke at skrive LIST men til gengæld kan man skrive ls. Dette gør, at man får de samme muligheder. I Wireshark kan man se at det bliver lavet om til LIST når man kigger på pakkerne

```
ftp> LIST
?Invalid command.
ftp>
```

[2] https://www.ietf.org/rfc/rfc959.txt
[3] https://www.ietf.org/rfc/rfc959.txt



Vi har i vores implementation sat følgende opsætning op hver gang der sendes en LIST kommando.

1. PASV \r\n sendes til serveren
    a. Vi bruger respons til at klargøre en socket til data modtagelse
2. TYPE A \r\n
    a. Sætter dataene der sendes gennem socket til at være ASCII
3. LIST \r\n
    a. Dataene overføres gemmes i en buffered reader, fra denne udskriver vi hver linje som vi har modtage

```
LIST
227 Entering Passive Mode (127,0,0,1,252,203)
64715
200 Type set to A.
150 Opening ASCII mode data connection for '/bin/ls'.
226 Transfer complete.
total 12
```

Her er et lille udsnit af det der udskrives af LIST

```
drwxr-xr-x  12 clausstaffe  staff      408 Oct  9  2012 software
-rw-r--r--   1 clausstaffe  staff       16 Apr 14 15:28 store.txt
```

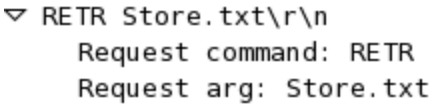
RETR Kommando[4]

RETR kommandoen bruges til at hente en kopi af en fil på serveren ved hjælp af en sti man har givet den.

[4] https://www.ietf.org/rfc/rfc959.txt

1	0.000000000	::1	::1	FTP	92	Request: SIZE Store.txt	54564	21
2	0.000052000	::1	::1	TCP	76	21→54564 [ACK] Seq=1 Ack=17 Win=12730 Len=0 TSval=592549992 TSecr=5	21	54564
3	0.000211000	::1	::1	FTP	85	Response: 213 332	21	54564
4	0.000032000	::1	::1	TCP	76	54564→21 [ACK] Seq=17 Ack=10 Win=50718 Len=0 TSval=592549992 TSecr=	54564	21
5	0.000074000	::1	::1	FTP	82	Request: EPSV	54564	21
6	0.000034000	::1	::1	TCP	76	21→54564 [ACK] Seq=10 Ack=23 Win=12729 Len=0 TSval=592549992 TSecr=	21	54564
7	0.000087000	::1	::1	FTP	124	Response: 229 Entering Extended Passive Mode (|||55059|)	21	54564
8	0.000026000	::1	::1	TCP	76	54564→21 [ACK] Seq=23 Ack=58 Win=50712 Len=0 TSval=592549992 TSecr=	54564	21
9	0.000170000	::1	::1	TCP	88	55060→55059 [SYN] Seq=0 Win=65535 Len=0 MSS=16324 WS=8 TSval=59254	55060	55059
10	0.000071000	::1	::1	TCP	88	55059→55060 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16324 WS=32	55059	55060
11	0.000032000	::1	::1	TCP	76	55060→55059 [ACK] Seq=1 Ack=1 Win=407800 Len=0 TSval=592549992 TSe	55060	55059
12	0.000014000	::1	::1	TCP	76	[TCP Window Update] 55059→55060 [ACK] Seq=1 Ack=1 Win=407776 Len=0	55059	55060
13	0.000031000	::1	::1	FTP	92	Request: RETR Store.txt	54564	21
14	0.000027000	::1	::1	TCP	76	21→54564 [ACK] Seq=58 Ack=39 Win=12729 Len=0 TSval=592549992 TSecr=	21	54564
15	0.000110000	::1	::1	FTP	146	Response: 150 Opening BINARY mode data connection for 'Store.txt'	21	54564
16	0.000030000	::1	::1	TCP	76	54564→21 [ACK] Seq=39 Ack=128 Win=50703 Len=0 TSval=592549993 TSec	54564	21
17	0.000068000	::1	::1	FTP-DATA	408	FTP Data: 332 bytes	55059	55060
18	0.000031000	::1	::1	TCP	76	55060→55059 [ACK] Seq=1 Ack=333 Win=407464 Len=0 TSval=592549993 TS	55060	55059
19	0.000001000	::1	::1	TCP	76	55059→55060 [FIN, ACK] Seq=333 Ack=1 Win=407776 Len=0 TSval=592549	55059	55060
20	0.000067000	::1	::1	TCP	76	[TCP Out-Of-Order] 55059→55060 [FIN, ACK] Seq=333 Ack=1 Win=407776	55059	55060
21	0.000022000	::1	::1	TCP	76	55060→55059 [ACK] Seq=1 Ack=334 Win=407464 Len=0 TSval=592549993 TS	55060	55059
22	0.000005000	::1	::1	FTP	100	Response: 226 Transfer complete.	21	54564
23	0.000015000	::1	::1	TCP	76	[TCP Dup ACK 21#1] 55060→55059 [ACK] Seq=1 Ack=334 Win=407464 Len=(	55060	55059
24	0.000020000	::1	::1	TCP	76	[TCP Dup ACK 20#1] 55059→55060 [ACK] Seq=334 Ack=1 Win=407776 Len=(	55059	55060
25	0.000022000	::1	::1	TCP	76	54564→21 [ACK] Seq=39 Ack=152 Win=50700 Len=0 TSval=592549993 TSec	54564	21
26	0.000457000	::1	::1	TCP	76	55060→55059 [FIN, ACK] Seq=1 Ack=334 Win=407464 Len=0 TSval=592549	55060	55059
27	0.000044000	::1	::1	TCP	76	55059→55060 [ACK] Seq=334 Ack=2 Win=407776 Len=0 TSval=592549993 TS	55059	55060
28	0.000129000	::1	::1	FTP	92	Request: MDTM Store.txt	54564	21
29	0.000045000	::1	::1	TCP	76	21→54564 [ACK] Seq=152 Ack=55 Win=12728 Len=0 TSval=592549993 TSec	21	54564
30	0.000072000	::1	::1	FTP	96	Response: 213 20150414093537	21	54564
31	0.000039000	::1	::1	TCP	76	54564→21 [ACK] Seq=55 Ack=172 Win=50698 Len=0 TSval=592549994 TSec	54564	21

Man kan se den bruger de første 12 linier på at skabe forbindelse og første begynder at hente Store.txt i linie 17 (FTP-Data). På linie 1 kan man se, at den beder om størrelsen. I vores program bruges SIZE også for at allokere et byte[] af tilstrækkelig størrelse.



I Wireshark har vi kigget i pakkerne for at se hvad en funktionel FTP-klient sender, når der taste f.eks. *get store.txt*.

Vi har i vores implementation sat følgende opsætning op hver gang der sendes en LIST kommando.

4. PASV\r\n sendes til serveren
    a. Vi bruger respons til at klargøre en socket til datamodtagelse.
5. TYPE I\r\n
    a. Vi sætter den til at sende binary data, dette gør vi kan sende billeder og har ikke haft nogen negativ effekt på vores txt fil overførsel.
6. SIZE /path/filnavn.txt\r\n
    a. Denne bruges til at finde størrelsen af filen, vi bruger dette til at oprette et byte array hvori files gemmes indtil vi skriver den ned på drevet
7. RETR /path/filnavn.txt\r\n
    a. Dette starter overførslen af filen, vi gemmer den på en brugerdefineret placering

Her er et eksempel på at vi henter store.txt gennem consolen i vores løsning.

```
RETR store.txt
227 Entering Passive Mode (127,0,0,1,254,6)
65030
200 Type set to I.
213 16
SIZE? :16
write a system path for file placement: fx /Users/clausstaffe
/Users/clausstaffe/storeasd.txt
150 Opening BINARY mode data connection for 'store.txt' (16 bytes).
226 Transfer complete.
DataAvail: 16
Please write a command
```

Store.txt & Logfile.txt

Vores vægt skal hente sit data fra et sted, hvor der står varenr, varenavn og varevægt. Dette bliver gjort fra Store.txt. Denne fil bliver delt, så hver række i filen er en substring. Vi har lavet vores Store.txt om til CVS, som er en type fil, der opdeler værdierne i en string når der er et komma. Det gør det yderligere nemmere at lave subsubstrings af de forskellige værdier for en vare. Vi manipulere så med en vares vægt og ændrer det i Store.txt. Derefter oprettes der en ny række i Logfile.txt, hvor der står at vægten fra en vare er blevet opdateret.

FTP

FTP står for File Transport Protocol som er en TCP protokol der er beskrevet i RFC 959[5] fra 1985. Protokollen gør det muligt at transportere filer mellem hosts(server/klient) nemt og effektivt. Protokollen køre på Applikations laget. Måden dette gøres på er ved man har en server man opretter forbindelse til med sin klient. Herved har klienten adgang til de ting som administratoren har givet adgang til på forhånd. Når der er forbindelse mellem server og klient, er der mulighed for at skrive, overskrive, hente og slette på serveren.

Anonym FTP

Mulighed for at hente filer fra en FTP server uden nogen form for login findes også og dette kaldes “Anonym FTP” her skal man ikke give noget brugernavn eller kode for at kunne få adgang til filerne. Denne metode bruges mange steder på internettet. Dette gøres ved man

[5] https://www.ietf.org/rfc/rfc959.txt

giver adgang til at du kan hente filen, men der kan ikke rettes og uploades nogen ny fil, man kan kun hente.

Sikkerheden ved FTP er ikke god i sig selv da den sender brugernavn og password ukrypteret hvilket giver mulighed for at opsnappe det fx WireShark og derved kunne misbruge det.
Dog findes der andre udgave af FTP så som SFTP som krypterer dataen der sendes med protokollen. Her krypteres både den rå data samt login og password. I nogle situationer kan SFTP også være den bedste løsning da den ikke ville blive taget lige så let i firewallen.

Vi bruger ProFTPD(open source) på ZYBO-board, og en indbygget FTP på vores Mac-computere, som startes fra terminalen med linjen:

```
sudo -s launchctl load -w /System/Library/LaunchDaemons/ftp.plist
```

Port[6]

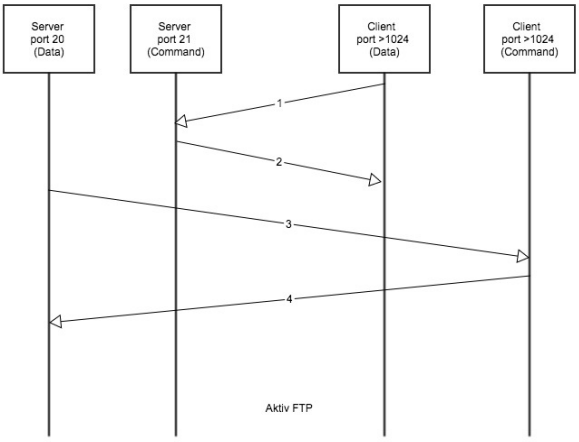
FTP køre som standard på port 20 og 21 som default på serveren, de kan dog ændres.
Port 20 (reserveret) bruges til data transfer som er her hvor man sender selve dataen.
Port 21 (reserveret) bruges til kontrol. Port 21 er den port man bruger til at sende requests igennem til FTP severen. I control beskeden til serveren fra klienten sends det hvilket port klienten lytter på. Dermed er det faktisk serveren der laver sin forbindelse til clinten(Aktiv FTP). Klienten giver bare adressen hvorpå tingene de skal leveres hvorefter den replayer når det er modtaget.

Passiv og aktiv FTP

Ser man på forskellen mellem Aktiv og passiv FTP ligger den størst forskel i at man i passiv kun får specificeret en port hvor i aktiv får man 2 på server siden, nemlig både Commando porten og Data porten.
Dette gør at det kan være svære at opsnappe hvilken port der bliver brugt i passiv FTP da man kun kan finde ud af hvilken port den skal modtage en kommando på. Og ikke ved hvilken port den sender ud på. Samt hvem der opretter Data forbindelsen i aktiv er det serveren der opretter den hvor i passiv er det klienten

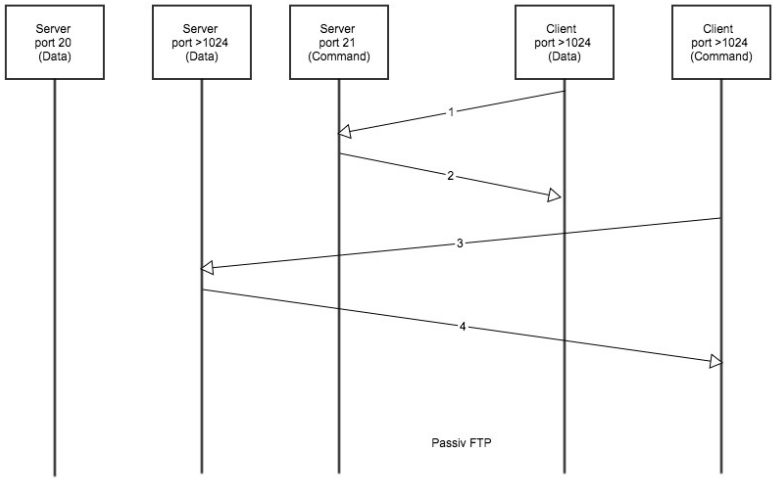
[6] https://www.ietf.org/rfc/rfc959.txt

Aktiv FTP[7]



1. klienten kontakter server og beder om en fil.
2. Server svar på beskeden
3. Serveren sender fil til den port som klientenen lytter på.
4. klientenen sender et ACK(Acknowledgement) sendes som regel i ASCII, tilbage til serveren for at vise klientenen har modtaget pakken og er klar til næste.

Passiv FTP[8]

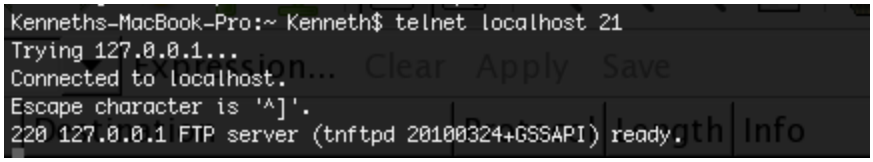


1. klienten kontakter server og beder om en fil.
2. Server svar på beskeden og fortæller hvilket port den sender dataen fra.
3. klientenen sender et ACK(Acknowledgement) sendes som regel i ASCII, tilbage til serveren for at vise klientenen har modtaget pakken og er klar til næste.

[7] http://www.cisco.com/web/about/ac123/ac147/ac174/ac199/about_cisco_ipj_archive_article09186a00800c85a7.html
[8] http://www.cisco.com/web/about/ac123/ac147/ac174/ac199/about_cisco_ipj_archive_article09186a00800c85a7.html

4. Serveren sender Dataen til klientenen i form af pakker.

En måde at tjekke på om den bruger port 21 er ved at have ip adressen til FTP serveren kan gøres gennem telnet hvor man i terminalen skriver: telnet ip port. Fx. “telnet 192.168.2.2 21”.



No.	Time	Source	Destination	Protocol	Length	Info	Src_port	Dest_port
1	0.000000000	127.0.0.1	127.0.0.1	TCP	68	54444→21 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=32 TSval=5898335	54444	21
2	0.000066000	127.0.0.1	127.0.0.1	TCP	68	21→54444 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16344 WS=32 TS	21	54444
3	0.000016000	127.0.0.1	127.0.0.1	TCP	56	54444→21 [ACK] Seq=1 Ack=1 Win=408288 Len=0 TSval=589833592 TSecr=	54444	21
4	0.000015000	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 21→54444 [ACK] Seq=1 Ack=1 Win=408288 Len=0 TS	21	54444
5	0.008706000	127.0.0.1	127.0.0.1	FTP	114	Response: 220 127.0.0.1 FTP server (tnftpd 20100324+GSSAPI) ready.	21	54444
6	0.000030000	127.0.0.1	127.0.0.1	TCP	56	54444→21 [ACK] Seq=1 Ack=59 Win=408224 Len=0 TSval=589833600 TSecr	54444	21

Der er brugt wireshark til at sniffe med, man kan se hvad der er blevet sendt frem og tilbage og at der kan man se at der er blevet svaret både gennem terminalen men også på Wireshark da der er fundet pakker der er sendt retur.

De “tilfælde” porte som både klienten og serveren kan bruge ligger i spektrummet fra 1024-65535 hvilket giver ca 64511 teoretiske ledig port nr de kan bruge.
Grunden til man ikke bruge port nr 0-1024 er fordi de er reserveret.[9]

[9] http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?&page=1