APRIL 12, 2023

PORTFOLIO 1

DATA 2410

LATIFA AJRAM
S349520
Oslo Metropolitan University

1 Introduction

I dette prosjektet har jeg designet og implementert 'simpleperf' min egen forenklede versjon av iPerf ved hjelp av sockets. den skal brukes til å teste nettverkytlesen og estimere end-to-end throughput. Simpleperf kjører på et virtuelt nettverk som er administrert av mininet inne i en virtuell maskin.

Min deltagelse på lab timene der vi brukte iperf og mininet har vært til stor hjlep for å forstå applikasjonen. Samt har jeg brukt min tidligere web client/ web server applikasjonen. Forskjellen mellom simpleperf og cleint/server applikasjoenen er at data sendes i bunter på 1000 bytes.

Det er spennede å se hvor nøyaktig og pålitelig målingene mine med simpleperf er sammnlignet med i perf.

I denne rapporten skal jeg først beskrive Simpleperf og topologien jeg brukte for å evaluere simpleperf. Under performance evaluations skal jeg nevne og forklare verktøy som jeg brukte i dette eksprimentet (ping, iperf, Fairness measures). Videre skal jeg evaluere min simpleperf ved å måle latency og throughput i en rekke tester. Til slutt skal jeg starte med testene og diskutere resultatene.

2 Simpleperf

Simpleperf er en implementasjon av en enkel TCP-server som lytter etter tilkoblinger fra klienter, og måler båndbredden til data som overføres fra klientene. Verktøyet kjører i servermodus og i klientmodus.
Først importerer koden noen moduler: ipaddress,socket,argparse,time,threading,sys, math,enum og tabulate.
Deretter defineres en enum-klasse kalt "VolumeFormat" som angir forskjellige enheter for datavolum(B,KB,MB)
Deretter defineres 4 funksjoner:

- Format_data_volume(): regner om datavolum
- check_port(val): sjekker om en gitt port er en gyldig port
- check_time(val): sjekker om en gitt tid er større en null
- check_ip(address): sjekker om en git IP-adresse er en gyldig IPv4 -adresse.

I servermodus mottas TCP-pakker og sporer hvor mye data som ble mottatt fra tilkoblede klienter, ved å behandle en tråd for klient tilkoblingen. Funksjonen 'handle_client(conn,addr,format)' håndterer klientilkoblinger og måler båndbredden til data som overføres. Denne funksjoner tar tre parametere:

- 'conn': det en en socket-tilkobling som representerer tilkoblingen til klienten
- 'addr': inneholder IP-adressen og portnummeret til kilenten som er tilkoblet.
- 'format': spesifiserer formatet som båndbredden skal vises i (Bytes/s,KB/s eller MB/s)

Inne i funksjon 'handle_client ' defineres noen variabler som 'start-time','antall_bytes' og bandwidth. Deretter startes en løkke som mottar data fra klienten til det ikke er mer data igjen. Hver gang en pakke med data mottas, økes variabelen 'antall_bytes' med størrelsen på pakken, og variabelen 'end-time' oppdateres med nåtiden. Deretter beregner funksjonen tiden det tok å motta data(i sekunder) ved å ta differansen mellom'end_time' og 'start-time', og lagrer resultater i variabelen 'duration'. Til slutt beregner funksjonen båndbredden til dataene som er overført fra klienten ved å dividere'antall_bytes' med 'duration',o konvertere fra bytes til KB eller MB, avhengig av 'format '-parameteren.(ved bruk av format_data_volume() funksjonen)

Når det ikke er mer data å motta,sender funksjonen en bekreftelsesmelding('ACK') til klienten, og skriver ut resultatene til skjermen.
Inne i server() metoden lytter server på en spesifisert iP-adresse og portnummer for tilkoblinger og oppretter en ny tråd for hver klient som kobler seg til.Hver klient kjører i egen tråd, det betyr at flere klienter kan koble seg til server samtidig og utøfre testing uten å påvirke hverandre.

I clientmodus, etableres en TCP-tilkobling med simpleperf-serveren og sendes data i bunter på 1000 bytes. Dette skjer i client() funksjonen i en evig loop , antall byte sendt oppdateres samt tiden som har gått siden klienten startet. Hvis intervalleter definert, vil funksjonen skrive ut informasjon om båndbredde og antall bytes sendt etter hvert intervall. Når tiden som er satt som 'duration 'er gått ,sender klienten en 'FINISH'-melding til serveren og avslutter tilkoblingen. Til slutt skrives det ut antall byte sendt , båndbredde og tiden det tok å sende datapakkene.client () funksjonen tar inn følgende argumenter:

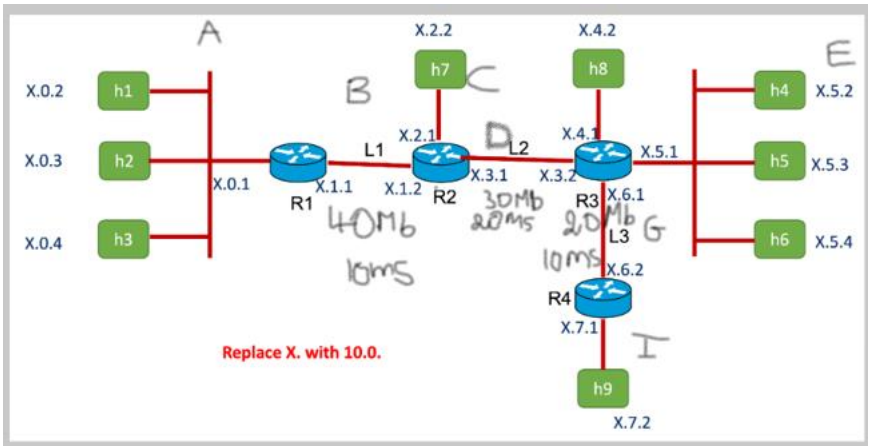
- Serverip: IP-adressen til serveren klienten skal koble seg til
- Port: Portnummeret til serveren klienten skal koble seg til
- Format:Streng som bestemmer formatet for båndbredden som skal vises(KB,MB,B).
- Duration:tiden klienten skal sende datapakker ,blir definert med -t falg (i sekunder)
- Intrerval: for utskrift av statistikk
- Num: Antall byte som skal sendes

Til slutt sjekker koden om navnet på det gjeldende programmet som kjøres er hovedprogrammet ved hjelp av if__name__== main(). Hvis dette er tilfellet starter koden å behandle argumentene som er sendt til programmet via kommandolinjern.

Dette gjøres ved å opprette en argparse-parser-objekt med en beskrivelse av programmet og legge til forskjellige argumentgrupper. Deretter henter koden ut verdiene for de ulike argumentene fra argparse-objektet og lagrer dem i variabler som skal brukes videre i koden.
Hvis «-n» er spesifisert etterfulgt med mengde data som skal sendes og enheten (KB,B,MB), splittes de og konverteres til bytes.
Hvis «-c» er spesifisert ,starter koden klientfunksjonen med de gitte argumentene.

Hvis «-s» er spesifisert starter koden server funksjonen med de gitte argumentene. Hvis parallelargumentet er spesifisert med en verdi mellom 2 og 5 ,startes flere klienttråder, ellers starter den bare en klient. Koden kaller til slutt hjelpefunksjoner for å sjekke gydlighet for portnummer,ipadresse og tid.

3 Experimental setup



Figur 1 Topology

Topologien (Figur 1) beskriver en nettverksoppsett som består av flere forskjellige subnett, rutere og hosts. Jeg beskriver subnett A og B og C samme prinsip gjelder for alle andre subnetts:

- Subnett A består av tre hosts(h1,h2,h3), en bryter (s1) og en ruter(r1).disse er alle koblet sammen via bryteren. Hver host har en IP-adresse som tilhører 10.0.0.0/24 subnettet, og

standardruten er via 10.0.0.1. r1 har IP-adressen 10.0.0.1/24 på interface (r1-eth0), og er koblet til s1 via dette grensesnittet.

- Subnett B er koblet til subnett A via ruter r1. r1 har også et interface (r1-eth1) som er koblet til r2. r1 har IP-adressen 10.0.1.1/24 på dette grensesnittet(interface), mens r2 har IP-adressen 10.0.1.2/24 på interface r2-eth0. Kobllingen mellom r1 og r2 har en båndbredde på 40 Mbps, delay på 10 ms og en kørstørrelse på 67 pakker.
- Subnett C består av r2 og en host (h7). r2 har interface(r2-eth1) som er koblet til h7, med IP-adressen 10.0.2.1/24 på dette grensesnittet.

L1, L2, L3 angir delay i millisekunder som en pakke vil oppleve når den passerer koblingen mellom tilknyttede enheter. For eksempel en delay på 10 ms i koblingen mellom r1 og r2 bety at det tar 10 ms for en pakke å reise fra r1 til r2.

Kø størrelse (windows size) refererer til mengden av data som sendes fra en sender til en mottaker før det kreves en bekreftelse (ACK) fra mottakeren.

4 Performance evaluations

4.1 Network tools

Ping og iperf er begge verktøy som jeg brukte i Mininet:

iPerf:

I dette eksperimentet har jeg brukt iPerf til å generere pakker som skal sendes ved hjelp av TCP eller UDP fra en datamaskin til en annen.det beregnes hastigheten på overføringer. For at testen skal lykkes må begge maskiner kjøre iPerf med riktig konfigurasjon på sin ende av forbindelse f.eks: iPerf -s(på server side) iPerf -c (server-IP.adresse) (på client side).
IPerf er en applikasjon som generer data og sender den til transport protokoll så fort som mulig, der skal transportprotokoll levere den til destinasjon. Man kan også starte iPerf test ved å bruke UDP,der er vi ikke begrenset med transport mekanism, med UDP sendes data så fort som mulig uten error,congestion eller flow kontroll.

Ping:

Med ping sendes det en ICMP-forespørsel til en annen enhet i nettverket og måler svartiden for å bekrefte om enheten er tilgjengelig og hvor lang tid det tar å få svar(RTT). Jeg brukte den i eksperimentet for å teste nettverksforbindelsen og til å feilsøke problemer med nettverkskommunikasjon.

Simpleperf: min forenklede versjon av IPerf.

Fairness.py:

Dette er en selvutviklet kode (fra lab-assignement 1) som beregner Jain's fairness index(JFI), inneholder en funksjon kalt jainsall som tar an liste av throughput verdier og returnerer a JFI. Jeg brukte denne for å avgjøre om brukere(hosts) mottar en rettferdig del av systemsressurene

4.2 Performance metrics

For å evaluere min simpleperf skal jeg måle latency og bandwidth mellom routere og mellom hosts.

Link Latency refererer til tiden det tar for en datapakke å reise fra en enhet til en annen over en nettverksforbindelse. Måles i millisekunder og kan påvirkes av en rekke faktorer som avstanden mellom enhetene, mengden trafikk på nettverket og kvaliteten på nettverkssutstyret.

Bandwidth definerer hvor mye data som muligens kan reise i et nettverk i løpet av en tidsperiode. Måles i Mbps.Den beskriver nemlig hvor bred kommunikasjonskanalen er og hvor mye data som kan passere gjennom den på en gang.

4.3 Test case 1: measuring bandwidth with iPerf in UDP mode.

4.3.1 Results

Hosts	Bandwidth	
	resultat	forventet
h1-h4	31,2	30
h1-h9	21,0	20
h7-h9	21,0	20

4.3.2 Discussion

h1-h4

For å måle båndbredden mellom h1 og h4 vil jeg velge BW = 30. Slev om andre linken mellom r1 og r2 overfører data raskere(40Mb/s) vil den være avhengig av den linken som er tregest.

Hvis vi velger 40 vil ikke r2 klare å videresende bits like raskt som den mottar dem(link r1 og r2), i vårt tilfelle vil bits kun forlate r2 med en hastighet på 30Mb/s (links mellom r2 og r3), og dermed gi end til end throughput på 30Mb. Hvis bits fortsetter å ankomme r2 fra r1 med hastighet på 40 Mb og forlater r2 mot r3 på 30 Mb, vil backlogen av bits på r2 som venter på å bli overført videre til destinasjon (h4) vokse enormt, og dette fører til at det oppstår flaskehalser(bottlenecklink).

Resultatet viser at 37,5 Mbytes ble sendt med 31,2 Mb/s, det vil si at testen kunne overføre data med litt høyere hastighet enn det vi spesifiserte(30Mb), skyldes bedre nettverksforhold enn forventet.
UDP tilbyr ikke reliability, flow controll eller error-recovery, men åpner for rask overførsel mellom to hosts, hvor det er greit med noe datatap. Det blir ikke opprettet en connection mellom hosts, noe som medfører at det er en 'best effort' overføring.

på serversiden viste rapporten at kun 25,3 Mbytes ble mottatt med en båndbredde på 21,0Mb/s, det betyr at server(h4) kunne motta data i en lavere hastighet enn hastigheten de ble sendt i. dette kan skyldes nettverksstopp, jitter viste tapt datagram, det skyldes nettverksproblemer, under testen, som har ført til lavere båndbredde på serversiden.data vil bruke tid på å reise gjennom nettverket, denne tiden vil øke for hver ruter den passerer, den totale forsinkelsen vil være summen av forsinkelsen ved hver ruter (processing delay + queuing delay+ transmission delay og propagation delay).

Det at flere pakker ble mistet kan skyldes at en av routerne har hatt full buffer, En høy queuing delay fører til at båndbredden reduseres. Dermed selv om jeg valgte å sende data med 30Mb/s så kom ikke alle pakkene til sin destinasjon og noen ble tapt, samt hastigheten var redusert, det betyr hvis jeg hadde valgt lavere bandwidth vil data ankomme destinasjon med forventet bw og data blir mindre tapt.

h1-h9

For å måle båndbredden mellom h1 og h9 velger jeg bw=20, siden det er den som overstyrer andre linker som overfører bits med større hastighet 30 og 40Mb.

Resultatet viser at 25 Mbytes ble sendt med 21 Mb/s. Om vi valgte å kjøre testen med 30 M vil ikke r3 klare å overføre pakker med en hastighet større en 20 Mbps,fordi bw på link mellom r3 og r4 er satt til 20 Mbps.

På serversiden viser resultatet at data ankom server med en hastighet på 16,3 Mb/s, 3861 pakker ble mistet.Igjen er det mange faktorer som påvirker trafikken, overbelastning av routere, total forsinkelse ved hver router, gjør at hastigheten brukt på å sende data via linkene reduseres. Ved å velge lavere bandwidth enn den minste tilgjegelige(20) vil data overføres raskere og data blir ikke tapt.

h7-h9

For å måle båndbredden mellom h7 og h9 velger jeg også bw=20, enda fordi det er den laveste hastighet blant linkene som bits skal gjennom fo å nå sin destinasjon.

Resultatet viser at 25 Mbytes ble sendt med 21 Mb/s. Data sendes fra h7 gjennom r2,r3,r4 for å komme til h9, om vi velger bw på 30 ms vil ikke r3 klare å overføre data med mer enn 20Mbps.

jitter var på 1,042 ms som representerer forskjellen i forsinkelsen mellom datapakker, 5476 av total 17830 datagrams ble mistet,på serversiden indikerer rapporten at datagrams som ble mottatt kom med en hastighet på 14,5 Mbits/sec, dette skyldes igjen nettverksbelastning, og delays ved hver router. Det kan hende at r3 måtte vente på at tidligere pakker skal behandles før den kan sende ut neste pakke, dette kalles queuing delay.

Tap av data kan relateres til at trafikkintesiteten har økt. Hvis jeg hadde valgt mindre bw enn laveste tilgjengelige hadde data ankommet h9 med den hastigheten jeg valgte.

I Alle disse tre testene, har jeg øndersøkt å kjøre med lavere bandwidth, og resultatet viser at jo lavere bandwidth jeg velger desto mindre datatap blir det.Til konklusjon jeg burde ha valgt bandwidth lavere enn den laveste tilgjegelige båndbredden.

Når man bruker iperf i UDP-modus for å måle båndbredde, er det viktig å ha informasjon om nettverkstopologien, for eksempel, antall enheter, type av enhet, avstand mellom dem..etc. Men hvis vi ikke vet noe om nettverket, velger jeg å starte med en lav hastighet og gradvis øke den til maksimal hastighet er nådd, eller til forbindelsen begynner å miste pakker eller vise andre tegn på ustabilitet.

UDP test på server rapporterer jitter(the variance of the delay between packets and any packet loss). Altså ved å overvåke pakketap og jitter kan vi estimere maksimal hastighet for den gjeldende nettverkstopologien. Denne tilnærmingen kan imidlertid være upraktisk for store nettverk.

Et annet alternativ er å bruke ping som viser oss total delay som blir brukt ved å sende/motta data (round trip time), jo lavere RTT jo raskere vil kommunikasjon være mellom to noder, eller trouceroute som viser hvor mange noder data går gjennom og hvor mye er delay på hver node. Med dette kan vi identifisere flaskehalser eller overbelastningstidspunkter. Dermed vil vi være mer nøyaktig på båndbreddemålingen.(Kurose, 2022)

4.4 Test case 2: link latency and throughput

4.4.1 Results

Hosts	Bandwidth/mbps		RTT(avr)/ms		Deviation	Packet loss
	resultat	forventet	resultat	forventet		
r1-r2	31,58	40	27,109	20	14,669	0%

r2-r3	23,31	30	45,115	40	6,392	0%
r3-r4	15,83	20	69,823	20	11,639	0%

4.4.2 Discussion

Når vi kjører ping mellom routere sjekker vi om en maskin kan nås på nettet og er i stand til å svare Det er nettopp det som skjedde i alle tilfellene vist ovenfor i tabellen.
Vi ser at alle pakker har kommet sin destinasjon. 0% packet loss.

RTT: er den tiden det tar for en datapakke å reise fra klient til server og deretter tilbake til klienten.I topologeien er delay i link r1-r2 lik 10 ms,derfor blir forventet RTT 20ms,
delay i link r2-r3 lik 20 ms,derfor blir forventet RTT 40ms,
delay i link r3-r4 lik 10 ms,derfor blir forventet RTT 20ms
Bw i link r1-r2 er satt til 40Mbps i topologien, derfor forventer jeg at hastigheten på overføring av data mellom r1 or 2 blir 40Mbps. Det samme gjelder for r2-r3 (forventet 30) og r3-r4(forventet 20)

Jo lavere RTT, jo raskere vil kommunikasjon være mellom to noder.

Resultatet viser at den gjennomsnittlige RTT i alle tilfeller er høyere enn forventet ,og at bandwidth er lavere , grunnet:

- ✓ Avstand, jo lengre avstand et signal må reise, jo lengre tid vil det ta for en forespørsel å nå sin destinasjon og å få respons tilbake.
- ✓ Overføringsmedium: for eksempel. Kobbertråd, fiber optiske kabler, kan påvirke hvor raskt en forespørsel mottas av en router og respons tilbake
- ✓ Nettverkstrafikk nivå, RTT øker vanligvis når et nettverk er overbelastet. F.eks: mellom r3 og r4 var den gjennomsnitilige RTT på 69,8 ms og forventet var 20 ms.
- ✓ Routers responstid, tiden det tar for router å svare på en forespørsel er avhengig av dens prosesseringskapasitet, antall forespøsler som håndteres og arten av forespørselen.

Målingene hadde en høy standard deviation, det betyr at network delay er variabel og uforutsigbar.

Standard deviation viser stabiliteten eller konsistensen av network delay mellom to noder.

4.5 Test case 3: path Latency and throughput

4.5.1 Results

Hosts	Bandwidth/mbps		RTT(avr)/ms	
	resultat	forventet	resultat	forventet
h1-h4	22,36	30	65,587	60
h7-h9	15,87	20	61,062	60
h1.h9	17,08	20	81,253	80

4.5.2 Discussion

h1-h4
I topologien er delay i link mellom h1 og h4 satt til 30ms (10 ms + 20ms). Så forventet RTT er 60ms.

Mellom h1 og h4 er det 3 routere, og vi ser at 30 MB skal overstyre hastigheten i den ruten, derfor foventer jeg at bw skal være 30 Mbps, mens resultatet viste 22,36.

RTT og bandwidth er påvirket av til sammen summen av de 4 typer delays.

- ✓ Processing delay: tiden det tar for å undersøke pakkens header og finne ut hvor den skal
- ✓ Queueing delay: Tiden det tar å vente på at andre pakker skal bli sendt ut på linken
- ✓ Transmission delay: Tiden det tar for å skyve pakken ut på linken(veien mellom nodene)
- ✓ Propagation delay: Tiden det tar for å reise fra node til node

Altså det som kan ha påvirket latency er håndteringstid av datamaskiner mellom h1 og h4, det er 3 routere mellom dem . Når en pakke kommer til en router ,slår routeren opp i en forwarding table for å finne ut av hvilken utgående forbindelse pakken burde sendes videre på. Dette er processingsdelay. I tillegg kan det komme ekstra forsinkelse ved at en pakke må vente på at forrige pakke er ferdig med å bli sendt.

Imellomtiden blir pakken lagret i en ouput buffer (pakker kan ha ventet i en kø ved r1,r2 eller r3 eller i alle routere før de har blitt videresendt), dette er queuing delay. Når en ouput queue er full, skjer det pakketap(packet loss), det vil si at pakken blir forkastet og kommer ikke fram til sin destinasjon. Dette har ikke skjedd i vår test.

Pakker sendes fra h1 via r1,r2,r3 så destinasjonen er h4, så det tar tid å sende pakker fra en enhet til annen , denne tiden innebærer både transmission delay og propagation delay. Nemlig Den totale forsinkelsen vil da være summen av delay ved hver ruter og delay som skyldes at dataene må reise gjennom kablene mellom rutere.

h1.h9

Delay i link mellom h1 og h9 er satt til 40ms (10 ms + 20ms+10ms). Så forventet RTT er 80ms.
Vi ser at gjennomsnittet er 1,253 ms lengre en forventet RTT.

Vi ser ut fra topologien at 20 MB skal overstyre hastigheten i den ruten (den laveste), derfor foventer jeg at bw skal være 20 Mbps, mens resultatet viste 17,08. det skyldes igjen den totale summen av delay ved hver router(prosessingdelay+ queuing delay+ transmission delay+ propagation delay) .Avstanden mellom h1 og h9 er stor, og pakkene skal gå via 4 routere,som er koblet til alle andre hosts, noe som kan øke datatrafikk, dermed høyere latency og lav båndbredde enn forventet.

h7.h9

Delay i link mellom h7 og h9 er satt til 60ms (20 ms(r2,r3) 10ms(r3,r4)). Så forventet RTT er 60ms.
Vi ser ut fra topologien at 20 MB skal overstyre hastigheten i den ruten (den laveste), derfor foventer jeg at bw skal være 20 Mbps.

Vi ser RTT avg er 1,062 ms lengre enn forventet,skyldes nettverksbelastning eller forsinkelser i kommunikasjonen.med dette mener jeg den totale forsinkelsen ved hver router.
Sammenlignet med h1,h4 så er RTT mellom h7 og h9 mindre, selv om avstand mellom dem er lik.Det kan skyldes at nettverksforholdene mellom h1 og h4 er mindre optimale enn mellom h7 og h9.

4.6 Test case 4: effects of multiplexing and latency

Multipleksing er en teknikk som tillater at flere signaler blir sendt over en enkelt kanal, vi skal nå se på hvordan multipleksing påvirker både latency og båndbredde.

4.6.1 Results

Hosts	Bandwidth	RTT (avg)

	resultat	forventet	resultat	forventet
h1-h4	13,01	30	91,135	60
h2-h5	8,25	30	96,228	60

Hosts	Bandwidth		RTT (avg)	
	resultat	forventet	resultat	forventet
h1-h4	7,63	30	103,655	60
h2-h5	6,14	30	106,235	60
h3-h6	11,10	30	110,819	60

Hosts	Bandwidth		RTT (avg)	
	resultat	forventet	resultat	forventet
h1-h4	16,97	30	119,338	60
h7-h9	8,20	20	121,150	60

Hosts	Bandwidth		RTT (avg)	
	resultat	forventet	resultat	forventet
h1-h4	28,46	30	97,227	60
h8-h9	19,12	20	39,141	20

4.6.2 Discussion

- **h1-h4 / h2-h5**

Når data går samtidig mellom disse hosts via r1,r2,r3 blir RTT mye høyere enn det som er forventet ut fra topologien(r1-r2=delay er 10 ms, r2-r3= delay er 20 ms).så økt nettverks trafikk kan føre til høyere queuing delay og dermed en høyre RTT og lav bandwith.

Når jeg kjørte testen , har jeg kjørt samtidig ping fra h1 og h2 til h4 og h5, så simpleperf test rett etter, Når datapakke ankommer r1 fra h1 og h2, undersøker r1 pakkens overskrift for å finne riktig utgående kobling for pakken og dirrigerer deretter pakken til denne koblingen. I dette tilfelle lenken som fører til r2. En pakke kan sendes på en link bare hvis det ikke er noen pakke som sendes for øyeblikket på lenken og hvis det ikke er andre pakker foran den i køen. Hvis koblingen for øyeblikket er opptatt,eller hvis det allerde er andre pakker i kø for koblingen,den nye ankommende pakke vil da bli med i køen.Det er nettopp dette som skjedde ,fordi ut fra resultatet var RTT h2-h5< RTT h1-h4, samt bandwidth er lavere mellom h2-h5 enn mellom h1.h4. Hvis det var noen prosessingsforsinkelse i rutere, kan dette også bidra til høyere RTT.

Siden bw=8,25 mellom h2 og h5 og forventet er 30 Mbps ,kan man tenke på at det er en flaskehals et sted i nettverket som begrenser hastigheten på overføringen
Til oppsummering ,resultatet viser treg overføring av data gjennom nettverket.

- **h1-h4 / h2-h5 / h3.h6**

Topologien viser at bw = 40 mellom r1 og r2, bw= 30 mellom r2 og r3 , derfor satt jeg forventet til 30 Mbps. Det samme med RTT, delay mellom r1-r2=10 ms og delay mellom r2-r3 er satt til 20 ms ,derfor forventet RTT blir 60 ms.
Sammenlignet med forrige test,så har multiplexing effekt på både RTT og bandwidth, jo mer hosts som kommuniserer sammen,desto lav hastighet og treg kommunikasjon blir det.(økt RTT). Vi ser at forventet kapasitet 30 Mbps ble delt mellom alle tilkoblingene.

Fairness measures eller metrics brukes I nettverksutvikling for å avgjøre om brukere eller applikasjoner mottar en rettferdig andel av systemressursene. La oss bruke bandwidth fra de konkurerrende

dataoverføringene til å beregne Jains Fairness Index(JFI) ved hjelp av Python programmet fra lab-assignement 1 (jeg legger ved filen)

Resultatet var: 0.94. JFI er maksimal når alle brukere (h1, h2, h3) får samme tildeling. Beste tilfellet er når JFI=1 og verste tilfelle er når JFI= 1/n (n er antall brukere). I vår test ser det ut til at h1, h2, h3 fikk en rettferdig ressurstildeling.

- **h1-h4 / h7-h9**

Topologien viser at bw = 40 mellom r1 og r2, bw= 30 mellom r2 og r3 , derfor satt jeg forventet til 30 Mbps. Bw mellom r3 og r4 er 20 derfor forventet er 20 Mbps

Det samme med RTT, delay mellom r1-r2=10 ms og delay mellom r2-r3 er satt til 20 ms ,derfor forventet RTT blir 60 ms.delay mellom r3 og r4 er 10 ms, derfor forventet RTT er 60 ms.

Igjen økt datatrafikk har ført til høyere RTT og lav båndbredde. Når pakker sendes fra h1 de går via r1, så r2, samtidig kommer pakker fra h7 og når r2, her vil det være en kø slik at de første pakker som kom r2 sendes videre til r3, og det samme skjer når pakker ankommer r3.Nemlig ventetiden ved utgangslink for transfer er avhengig av overbelastningsnivå til routeren.Hver gang pakker kommer til en router sjekkes bitnivå feil i pakkene, og jo rask router og prosessor er jo mindre tid tar det.

Vi ser at båndbreddet til det overførte signalet er mindre enn den totale båndbredden til kanalen (16,97 < 30 og 8,20 <20), så den samme kanalen deles med flere brukere, på denne måten kan kanalbåndbredden utnyttes mer effektivt, og ved å bruke samme kanal. I midlertid konkurerrer datastrømmer om båndbredden, dette førte til at dataoverføring fra h7 til h9 opplevde forsinkelse.

- **h1-h4 / h8-h9**

Data fra h8 til h9 passerer gjennom r3 og r4 og utfra topologien er forventet bandwidht 20. mens mellom h1 og h4 er det 30 . Samt delay mellom r3 og r4 er 10 ms så forventet RTT er 20. og mellom h1 og h4 er forventet RTT 60 ms.

Sammenlignet med test 1 (h1-h4,h2-h5) ser vi at bandwidht nærmer seg det vi forventet, data gikk raskere grunnet datastrømmer deler ikke samme kanal, men de treffes i r3. så på grunn av mindre nettverkstrafikk , ble ikke båndbredden redusert så mye.Imidlertid RTT er høyere enn forventet, det skyldes total delay ved r3. Hvis vi igjen sammenligner dette resultatet sammen med resultatet fra h1-h4/h7-h9, ser vi effekten av datatrafikk på delay og bandwidht, h1,h4,h7,h9 deler samme kanal, det vil være queing delay ved hver router. Mens i test h1-h4/h8-h9 trafikken deler ikke samme kanal.

4.7 Test case 5: effects of parallel connections

4.7.1 Results

Hosts	Bandwidth(Mbps)		
	resultat		forventet
h1-h4	7,33	7,18	30
h2-h5	8,73		30
h3-h6	8,04		30

4.7.2 Discussion

Vi ser at kapasiteten deles likt mellom alle hosts (30/4), så hver hosts tilkobling fikk sin rettferdige andel av bandwidth, så end-to-end throughput for hver tilkobling er avhengig av den laveste bandwidth i kanalen (30 Mbps).
Jains Fairness Index(JFI)= 0,99 ,dette vil si at alle tilkoblinger fikk en rettferdig ressurstildeling.
Her legger jeg skjermbildet av funskjonen som har bergenet JFI:

```
print (jainsall ([7.33,7.18,8.73,8.04]))
```

lenken har hastighet på R=30, det er til sammen 4 TCP-tilkoblinger i denne testen, Vi har to parallele tilkoblinger i h1-h4 , alstså hver tilkobling får omtrent samme overføringshastighet R/4.

parallel connections kan påvirke båndbredden ved å øke den totale mengden data som kan overføres i løpet av en gitt periode. Det ble brukt to tilkoblinger fra h1 til h4 samtidig, så vi forventer at data blir overført med høyere hastighet enn en enkelt tilkobling. Imidlertid brukte de to tilkoblingene fra h2 og h3 samme kanal, noe som førte til nettverkskø og dermed fordeling av den tigjengelige nettverksbåndbredde.(Kurose, 2022)

5 Conclusions

Alle resultatene viste at forventet latency og bandwidth ikke alltid stemmer overens med de faktiske verdiene som ble målt i testen. Etter å ha brukt simpleperf konkulderer jeg med at båndbredden mellom forskjellige hosts i nettverket kan påvirkes av nettverksbelasting, totale forsinkelser ved hver router, køing og tap av datapakker. Det er altså viktig å ta hensyn til disse faktorene når man planlegger og designer nettverk for å oppnå optimal ytelse.

Ved testen av paralelle tilkoblinger var det ikke så lett å starte alle tilkoblinger samtidig, noe som kan ha påvirket testen. Det var spennende å sammenligne resultater jeg fikk ved bruk av simpleperf med resutlater ved bruk av iPerf, jeg har faktisk brukt iperf for nysjerrighets skyld i å teste parallel connections og fant ut at hver host fikk sin rettferdige del av bandwidth, imidlertid var bandwidth mye lavere enn det jeg fikk med simpleperf.

For å øke sjansen for at resultatene fra min simpleperf er pålitelige og nøyaktige, bør det gjøres en grundig feilsøking i koden.I tillegg bør testing skje under forskjellige forhold og med ulike nettverkskonfigurasjoner, dette inkluderer å endre topologien ved å teste med forskjellige antall hosts eller rutingkonfigurasjoner.

6 References (Optional)

Kurose, J. F. (2022). *Computer networking : a top-down approach* (8th edition, global edition.). Pearson Education.