APRIL 1, 2023

PORTFOLIO 1

DATANETTVERK OG SKYTJENESTER

AMNA DASTGIR
S364520 I
Oslo Metropolitan University

1 INTRODUCTION

iPerf er et verktøy for å måle nettverksytelse ('network throughput'). I dette prosjektet er det blitt utviklet og implementert en forenklet versjon av iPerf ved bruk av sockets, hvor verktøyet skal hete simpleperf. Simpleperf nettverket som ble skrevet skulle kjøres på et virtuelt nettverk som ble håndtert av mininet, inni en virtuell maskin. Dette ble så brukt dette til å måle ytelsen til et nettverk.

Verktøyet som ble utviklet, simpleperf, er ment for å sende og motta pakker mellom en klient og en server ved bruk av sockets. Det er derfor utviklet en kode slik at det kan kjøres i to modus: en server-og en klient modus.

Relevant kompetansemål hentet fra emneinformasjon på Oslomet sin side:

> *«Konfigurere datanettverk, bruke nettverksverktøy for å studere nettverkstrafikk, programmere sockets, bruke automatiseringsverktøy for å rulle ut applikasjoner og deres underliggende infrastruktur, bruke skybaserte tjenester som utviklingsplattform, bruke overvåkingssystemer for å overvåke ytelse og stabilitet til applikasjoner og driftsmiljøer»*

2 SIMPLEPERF

Helhetlig definerer koden en simpleperf-klient som sender data i en angitt tidsperiode, eller en mengde byte med flere parallelle tilkoblinger, etter å ha blitt koblet med en server. For hver kobling, dersom det er flere tilkoblinger, skrives det ut sammendragsstatistikk.

Slik tidligere nevnt kan programmet kjøre i enten klientmodus eller servermodus, og disse modusfunksjonene utnyttes til å bestemme operasjonsmodusen og undersøke ulike kommandolinjeargumenter. Argparse-modulen brukes for analyse av argumentene, hvor i vårt tilfelle vil '-s'-flagget aktivere servermodus og '-c'-flagget aktivere klientmodusen. For å angi forskjellige parametere kan i tillegg andre argumenter også overføres til programmet, og eksempler på disse er dataoverføringshastighet, portnummer og server-IP-adresse.

Funksjonen er i stand til å motta flere parametere slik som portnummeret til servere, IP-adressen til serveren, overføringsformat, parallelle tilkoblinger, utskriftsintervallet og antall byte som det er ment å overføre. Koden bygger seg ut på at noen parametere har standardverdier eksempelvis ved at dersom tidsvarighet ikke defineres, eller angis, vil data i 25 sekunder sendes av klienten. Funksjonen benytter socket og threading modulene i trengsel for å opprette en klient-socket og utallige tråder for parallelle tilkoblinger.

Starten av koden importerer nødvendige moduler og pakker, i tillegg til at det lages funksjoner som vil håndtere formateringen av byte-størrelsene, og for å lage et format med riktig innrykk i utdata slik som

oppgaven forespør. Det defineres også globale verdier som benyttes som standardverdier for server-IP-adressen, portnummeret og andre innstillinger.

Den neste funksjonen er kodet for å konvertere bytestørrelser til et ønsket format, til kilobytes eller megabytes for eksempel, og funksjonen heter 'format_bytes'. 'tabs' er funksjonen som er ment til å legge riktig inntrykk i output-data for å danne en fin utskrift.

*** SERVERMODUS ***

Når koden kjøres i en server modus, vil simpleperf motta TCP pakker og spore mengde data som skal mottas fra en tilkoblet klient. Det vil da kalkuleres og fremvises båndbredde ('bandwidth') med utgangspunkt i hvor mye data som skal mottas, og hvor mye av tid som har gått med i koblingen. Den første funksjonen er 'server' med hovedoppgave i å danne en socket og lytte, eller vente, på tilkoblinger. Da en tilkobling først er på plass opprettes det av funksjonen en tråd for hver klient som velger å koble seg til denne åpne serveren. Dette skjer med funksjonen 'handle_server' som hører etter tilkoblinger til serveren blir sluttet, ellers så vil 'server'-funksjonen kjøre i en evig løkke.

'Handle_server'-funksjonen vil samle inn data fra klienten samtidig regne tiden det tar for mottaksprosessen, mengde bytes, hastigheten til overføringen av data i enheten 'megabits per sekund'. Når alle koblingene er avsluttet vil resultatene fra alle koblingene samles og skrives ut i fin-format som tidlig ble definert i koden.

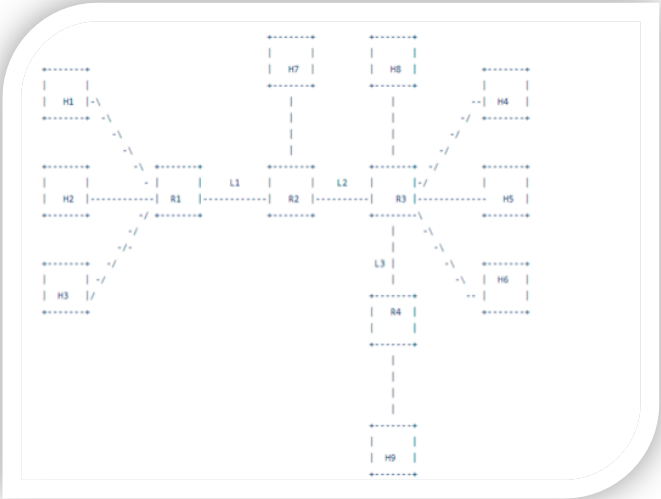
*** KLIENTMODUS ***

Kort fortalt vil klientfunksjonen koble til serveren etter å ha laget en klientsocket. For hver tråd vil en kobling dannes i tillegg til at det skal ventes til at alle trådene skal fullføre. For håndtering av klientdelen av koblingen er funksjonen 'handle_client' implementert. Med den sendes data til serveren ved et spesifikt intervall med tid, eller til en definert menge med byte skal sendes. Funksjonen måler tiden for overføringen av data og beregningen av mengden data. Til slutt vil det skrives ut resultater for det spesifikke intervallet.

Det importeres nødvendige moduler. Deretter defineres klientfunksjonen som parametere har portnummer, serverens IP-adresse, varigheten til testen, antall parallelle tilkoblinger, format for oppsummeringsresultatene, intervaller der statistikken skrives ut og mengde byte som skal overføres. Det skal ventes til at alle tråder er ferdig med å fullføre før det så skrives ut en oppsummering av hver tilkobling. 'handle_client' funksjonen, som tar inn argumentene klientsocket, IP-adressen til serveren, portnummeret, hvor lenge testen varte, formatet for oppsummeringsresultater, intervaller der statistikken skal skrives ut, mengde parallelle koblinger, antall byte som skal føres over og den endelige listen med oppsummeringsresultater. Lengre nede, rundt blokk 5 i koden, skal klientens IP-adresse hentes, og portnummeret, og der skal det skrives i en melding, en tyding på at klienten er koblet til serveren. Så er koden for konverteringen av antall bytes som skal beregnes til et heltall under. Så skal data sendes til serveren med et utgitt tidsintervall, eller til en gitt mengde bytes er overført. Tiden det tar for overføringen av

data og beregninger av mengden data utføres. Dersom antallet parallelle tilkobler er lik 1 så skal det skrives ut statistikk for det gitte intervallet. Dersom mengde bytes som skal overføres ikke er oppgitt så skal det sendes data i 25 sekunder, eller i tiden som oppgis, og til slutt vil dataene skrives ut.

3 EXPERIMENTAL SETUP



Den virtuelle nettverkstopologien som brukes til å evaluere det implementere simpleperf-verktøyet, er et komplekst nettverk som består av 4 subnet, der hvert subnet inneholder flere host, rutere og switcher sammenkoblet med lenker med spesifiserte egenskaper som bandwidth, delay, og maksimal kø-størrelse. Subnettene er koblet sammen med rutere, og danner en hierarkisk nettverkstopologi. Nettverket inneholder totalt 9 host, 4 rutere og 3 switcher. Hver host er tildelt en IP-adresse, og hver ruter er konfigurert til å sende videre IP-pakker mellom subnettene.

Topologien består av fire undernett:

- **Subnet A:** Dette subnettet har 3 hosts(h1, h2 og h3), en switch og en ruter(r1). IP-adresseområdet til dette undernettet er 10.0.0.0/24
- **Subnet B:** Dette subnettet består av 2 rutere(r1 og r2) koblet sammen med en kobling. IP-adresseområdet til dette undernettet er 10.0.1.0/24
- **Subnet C:** Dette subnettet har 1 hosts(h7) og en ruter(r2) koblet sammen med seg med en kobling. IP-adresseområdet til dette undernettet er 10.0.2.0/24
- **Subnet D:** Dette subnettet består av 2 rutere(r2 og r3) koblet sammen med en kobling, og en host (h8). IP-adresseområdet til dette undernettet er 10.0.3.0/24.

Dette virtuelle nettverket er designet for å likne en kompleks hierarkisk nettverkstopologi, der flere subnett er koblet sammen gjennom mellomliggende rutere for å muliggjøre kommunikasjon mellom hosts i forskjellige subnett.

4 PERFORMANCE EVALUATIONS

4.1 NETWORK TOOLS

Følgende verktøy er blitt utnyttet til ytelsesevaluering:

- iPerf: et verktøy for å måle nettverksytelse ved å generere trafikk mellom to endepunkter. Det har blitt brukt til å teste gjennomstrømningen (throughput) og båndbredden (bandwidth) i nettverket.

- Ping: et grunnleggende nettverksdiagnoseverktøy som sender ICMP-pakker til en mål-host og måler tiden det tar rundt for en tur. Det har blitt brukt til å teste tilkoblingen og ventetiden i nettverket.
- Mininet CLI: mininet kommandolinjegrensesnitt(CLI) tillater brukere samhandle med et Mininet-nettverk, starte og stoppe hosts og konfigurere nettverksgrensesnitt.
- Xterm: en terminalemulator som lar brukere åpne flere terminalvinduer i et Mininet-nettverk. Dette har blitt brukt til å samhandle med forskjellige host og rutere i nettverket, og kjøre kommandoer på dem.

4.2 PERFORMANCE METRICS

Den viktigste ytelsesberegningen som simpleperf-verktøyet måler er nettverksgjennomstrømning, som er hastigheten som data overføres mellom klienten og serveren med. Verktøyet regner gjennomstrømningen i Mbps (megabits per sekund) basert på den totale mengden data som sendes eller mottas og tiden det tar å overføre dataene. I tillegg til gjennomstrømming gir verktøyet også informasjon om mengden data som overføres og varigheten av overføringen. Denne informasjonen kan brukes til å bestemme effektiviteten til nettverket og identifisere eventuelle flaskehalser eller områder for forbedring.

4.3 TEST CASE 1: MEASURING BANDWIDTH WITH IPERF IN UDP MODE

For å utføre nettverkstesten på den angitte topologien, følger vi de neste trinnene:

1. Lag nettverkstopologien med den medfølgende konfigurasjonen, dette starter Mininet-nettverksemulatoren.
2. Åpne et Xterm-vindu på server-host og kontroller IP-adressekonfigurasjonen ved å bruke «iconfig»-kommandoen. Start iperf-serveren i UDP-modus, spesifiser IP-adressen som skal bindes til og utdatafilen for å lagre resultatene.
3. Åpne Xterm-vinduet på klient-host og start iperf-klienten i UDP-modus, spesifiser serverens IP-adresse og båndbredden (10M).
4. Gjenta trinn 2 til 3 for hvert host-par som skal testes
    a. H1 og H4
    b. H1 og H9
    c. H7 og H9

4.3.1 RESULTS

Resultatene som er oppnådd for hver koblings latens- og gjennomstrømningsmåler er oppsummert i følgende tabell:

Testtype	Klient-server par	Koblingsforsinkelse (ms)	Gjennomstrømning (Mbps)
UDP	H1-H4	1.943	10.5
UDP	H1-H9	3.263	9.80
UDP	H7-H9	0.915	10.5

Når det sees på resultatene, kan det observeres at tilkoblingslatensen for h7-h9 er den laveste ved 0.915 ms, mens koblingslatensen for h1-h9 er høyest ved 3.263 ms. Dette tyder på at avstanden mellom h7 og h9 er kortere enn mellom h1 til h9. Som vi ser i nettverkstopologien er det to hopp (rutere) fra h7 til h9, mens det fra h1 til h9 er tre hopp. Dette faktumet gjenspeiles også når det gjelder gjennomstrømning der det kan observeres at den høyeste verdien ble oppnådd i h1-h4 og h7-h9 testene, med et resultat på 10.5 Mbits/sek mens h1-h9 testen oppnådde en lavere gjennomstrømning på 9.80 Mbits/sek.

4.3.2 DISCUSSION

I den første testen mellom h1 og h4 var latens- og gjennomstrømningsmålingene vellykket, noe som indikerte på at nettverksbanen mellom h1 og h4 har god ytelse og lav latens. Den andre testen mellom h1 og h9 resulterte med indikasjoner om at nettverksbanen mellom h1 og h9 er lengre, som fører til redusert gjennomstrømning og høyere latens. Til slutt, i den tredje testen mellom h7 og h9, tydet resultatene på at nettverksbanen mellom h7 og h9 har god ytelse, og lav latens, lik den første testen. Resultatene viser viktigheten av å måle nettverksytelsesmålinger som gjennomstrømning, ventetid og pakketap for å forstå kvaliteten på tjenesten som tilbys av nettverket. Ved å utføre disse testene i en virtuell minintet-topologi er det mulig å simulere og måle ytelsen til forskjellige nettverksveier og identifisere områder som trenger forbedring eller optimalisering for å gi bedre QoS.

4.4 TEST CASE 2: LINK LATENCY AND THROUGHPUT

For å utføre nettverkstesten på den angitte topologien, følger vi de neste trinnene:

1. Lag nettverkstopologien med den medfølgende konfigurasjonen, dette starter Mininet-nettverksemulatoren.
2. Åpne et terminalvindu i hver ruter og kontroller iP-adressekonfigurasjonen ved å bruke kommandoen «iconfig». Bruk «ping»-kommandoen for å måle ventetiden mellom ruterne for hver kobling. Alternativet «-c» spesifiserer antall pakker som skal sendes (25 pakker), og kommandoen «| tee» lagrer utdataene til en tekstfil. Utdataene viser tur-retur-tiden (rtt) for hver pakke og minimums-, gjennomsnitts-, maksimums- og standardavviksverdier for alle pakker.
3. Bruk verktøyet simpleperf.py for å måle nettverksgjennomstrømningen for hver kobling. Skriptet setter opp en klient-server-modell, der serveren lytter på en bestemt port, og klienten sender datapakker til serveren. Skriptet beregner mengden data som mottas og dataoverføringshastigheten for et bestemt tidsintervall. Bruk de riktige alternativene i server- eller klientmodus med tanke på at testen må ha en varighet på 25 sekunder. Lagre resultatene i tekstfiler når testen er ferdig for å senere analyse.
4. Gjenta trinn 2 og 4 for koblinger L1, L2 og L3 i nettverkstopologien.

4.4.1 RESULTS

Resultatene som er oppnådd for hver koblings latens-og gjennomstrømningsmåler er oppsummert i følgende tabell:

Link	Latens (ms)	Gjennomstrømning (Mbps)
L1	22.299	37.44
L2	43.035	28.22
L3	22.125	18.92

For link L1 kan vi se at latensen er ganske lav, i snitt rundt 22 ms, mens gjennomstrømningen er relativt høy, i snitt rundt 37.44 Mbps. Dette antyder på at Link L1 er relativt rask og pålitelig. På den andre side, for Link L2, kan vi se at latensen er høyere i gjennomsnitt rundt 43 ms, mens gjennomstrømningen er lavere, i gjennomsnitt rundt 28.22 Mbps. Dette indikerer at Link L2 er tregere og mindre pålitelig enn Link L1. Til slutt, for Link L3, kan vi se at latensen igjen er relativt lav, i snitt rundt 22 ms, mens gjennomstrømningen er lavere enn de andre koblingene, i snitt rundt 18.92 Mbps. Dette kan antyde at Link L3 er noe lik Link L1 når det gjelder latens, men har lavere gjennomstrømning.

4.4.2 DISCUSSION

Basert på de oppnådde resultatene kan vi konkludere med at ytelsen til koblingene i den virtuelle mininet-topologien varierer betydelig. Link L1 viser den beste ytelsen når det gjelder latens og gjennomstrømning, noe som indikerer at den kan håndtere en høy mengde data med lav latens. Link L2 har derimot høyere latenstid og lavere gjennomstrømning, noe som indikerer at den kanskje ikke er i stand til å håndtere store mengder data like effektivt som Link L1. Til slutt viser L3 liknende latens som Link L1, men har lavere gjennomstrømning, noe som antyder at den kan være egnet for applikasjoner som krever lav latens, men ikke nødvendigvis høy gjennomstrømning.

Nettverkstopologien spiller en kritisk rolle i å bestemme ytelsen til nettverket. I dette tilfellet kan det å legge til flere rutere i nettverket føre til høyere ventetid og lavere gjennomstrømning. Vi må vurdere nettverkstopologien nøye når vi designer og konfigurerer nettverk for å optimalisere ytelsen. På en annen side er det viktig å merke seg at ytelsen til koblingene i et virtuelt miljø kan avvike fra ytelsen i et virkelighetsscenario på grunn av ulike faktorer som virtualiseringskostnader, nettverksoverbelastning og maskinvarebegrensninger.

4.5 TEST CASE 3: PATH LATENCY AND THROUGHPUT

For å utføre nettverkstesten på den angitte topologien, følges de neste trinnene:

5. Lag nettverkstopologien med den medfølgende konfigurasjonen, dette starter mininett-nettverksemulatoren.

6. Åpne et Xterm-vindu på serverhosten og kontroller IP-adressekonfigurasjonen ved å bruke «iconfig»-kommandoen.
7. Åpne Xterm-vinduet på klienthosten og bruk «ping»-kommandoen med serverens IP-adresse og alternativet «-c» for å spesifisere antall pakker som skal sendes (25 pakker). Utdataene viser tur-retur-tiden (rtt) for hver pakke og minimums-, gjennomsnitts-, maksimums- og standardavviksverdier for alle pakker. Bruk kommandoen « tee» for å lagre utdataene til en tekstfil.
8. Bruk verktøyet simpleperf.py for å måle nettverkets gjennomstrømning mellom klienten og server. Skriptet setter opp klient-server-modell, der serveren lytter på en bestemt port, og klienten sender datapakker til serveren. skriptet beregner mengden data som mottas og dataoverføringshastigheten for et bestemt tidsintervall. Bruk de riktige alternativene i server- eller klientmodus med tanke på at testen må ha en varighet på 25 sekunder. Lagre resultatene i tekstfiler når testen er ferdig for senere analyse.
9. gjenta trinn 2 til 4 for hvert host-par som skal testes:
    a. h1 og h4
    b. h7 og h9
    c. h1 og h9

4.5.1 RESULTS

Resultatene oppnådd fra testene for hvert host-par er oppsummert i følgende tabell:

Sti	Gjennomsnittlig RTT (ms)	Gjennomstrømning (Mbps)
H1 -H4	62.347	26.14
H7-H9	61.958	16.53
H1-H9	83.004	15.94

Fra tabellen kan det observeres at banen h1-h4 har lavest latens og høyest gjennomstrømning, mens h1-h9 har høyest latens og lavest gjennomstrømning. Dette indikerer at nettverkstopologien ikke er optimaliser for h1-h9-banen, noe som kan ha en negativ innvirkning på applikasjonsytelsen.

4.5.2 DISCUSSION

Forskjellen i latens og gjennomstrømning mellom de tre banene kan tilskrives antall hopp (rutere) mellom server- og klientverter og båndbreddekapasiteten til koblingene. H1-h4-banen har tre hopp, mens h1-h9-banen har fire hopp som forklarer resultatet av dens lavere båndbreddekapasitet. Resultatene tyder også på at optimalisering av nettverkstopologien kan forbedre applikasjonsytelsen og brukeropplevelsen. Dette kan innebære å legge til flere direkte koblinger mellom host, øke båndbreddekapasiteten til flere koblinger, eller finne alternative veier med færre hopp.

4.6 TEST CASE 4: EFFECTS OF MULTIPLEXING AND LATENCY

For å utføre nettverkstesten på den angitte topologien, følger vi de neste trinnene:

1. Lag nettverkstopologien med den medfølgende konfigurasjonen, dette starter Mininet-nettverksemulatoren.
2. Åpne Xterm-vinduer på serverhostene og kontroller IP-adressekonfigurasjonen ved å bruke «iconfig»-kommandoen. Bruk verktøyet simpleperf.py for å sette opp servermodus ved å bruke de riktige alternativene. Verktøyet beregner mengden data som mottas og dataoverføringshastigheten for et bestemt tidsintervall. Lagre resultatene i tekstfiler når testen er ferdig for senere analyse.
3. Åpne Xterm-vinduer på klienthoster og sett opp klienmodusen ved å bruke det enkle verktøyet og de riktige alternativene. Hvis mulig, utfør kommandoene samtidig på hver klient for å garantere multipleksing av datasending. Lagre resultatene i tekstfiler når testen er ferdig for senere analyse.
4. Åpne nye Xterm-vinduer på klienthoster og bruk «ping»-kommandoen med serverens IP-adresse og alternativet «-c» for å spesifisere antall pakker som skal sendes (25 pakker). Denne kommandoen må utføres raskt etter at klienter begynner å sende data. Utdataene viser tur-retur-tiden (rtt) for hver pakke og minimums-, gjennomsnitts-, maksimums- og standardavviksverdier for alle pakker. Bruk kommandoen « tee» for å lagre utdataene til en tekstfil.
5. Gjenta trinn 2 til 4 for hvert host-par som skal testes
    a. H1 og H4, H2 og H5
    b. H1 og H4, H2 og H5, H3 og H6
    c. H1 og H4, H7 og H9
    d. H1 og H4, H8 og H9

4.6.1 RESULTS

Tabellen under oppsummerer baneforsinkelsen og gjennomstrømningsresultatene for de tre forskjellige eksperimentene:

Eksperiment	Hosts	Latens (ms)	Gjennomstrømning(Mbps)
1	H1 – H4	69.400	9.88
	H2 – H5	65.807	9.30
2	H1 – H4	78.510	7.82
	H2 – H5	73.471	6.77
	H3 – H6	67.310	7.37
3	H1 – H4	78.414	8.47
	H7 – H9	59.835	9.89
4	H1 – H4	71.208	20.53
	H8 – H9	26.806	15.90

I eksperiment 1, der bare h1-h4 og h2-h5 kommuniserer samtidig tyder resultatene på at nettverket ikke er overbelastet i det hele tatt og er i stand til å støtte kommunikasjonen mellom disse hostene samtidig uten store latens- eller gjennomstrømningsproblemer.

I eksperiment 2, hvor tre par host kommuniserer samtidig, øker koblingsforsinkelsene betydelig og gjennomstrømningshastighetene reduseres merkbart. Dette indikerer at nettverket opplever trafikkbelastning, noe som resulterer i forsinkelser og reduserte dataoverføringshastigheter. Dette eksperimentet demonstrerer effekten av økende trafikkbelastning på nettverksytelse.

I eksperiment 3, hvor h1-h4 og h7-h9 kommuniserer samtidig, øker koblingslatensene, men gjennomstrømningshastighetene forblir relativt høye. Dette tyder på at nettverket ikke er så overbelastet som i eksperiment 2, dette skjer fordi host 3 ikke kommuniserer med noen andre hosts. Forskjellen i linklatens mellom h1-h4 og h7-h9 skyldes forskjeller i ruteveien eller fysisk avstand.

I eksperiment 4, hvor h1-h4 og h7-h9 kommuniserer samtidig, øker koblingslatensene for h1-h4, men gjennomstrømningshastigheten er høy. Dette indikerer at data sendes av klienter som bruker forskjellige ruteveier som vi ser i topologien. Det faktumet om at latensen i h1-h4 er høy, skyldes at stier krysses på ruter R3 hvor data fra hver tilkobling multiplekses for ruting som introduserer forsinkelser.

4.6.2 DISCUSSION

Hvis vi foretar en sammenlikning med resultater på testtilfelle 2 kan vi se at når h1-h4 og h2-h5 kommuniserer samtidig, reduseres gjennomstrømming for h1-h4 mer enn en halv fra 26.14 Mbps til 9.88 Mbps. På den andre siden, når h1-h4, h2-h5 og h3-h6 kommuniserer samtidig, reduseres gjennomstrømming for h1-h4 mer enn en tredjedel fra 26.14 Mbps til 7.82 Mbps. Når h1-h4 og h7-h9 kommuniserer samtidig, forblir gjennomstrømming for h1-h4 relativt høy som testtilfellet 2, og dette fordi data ikke multiplekses over lenker.

Vi kan se at når alle tre host-parene kommuniserer samtidig, er den totale ventetiden høy og gjennomstrømningen for hver tilkobling er lavere sammenliknet med da de kommuniserte individuelt. Dette skyldes sannsynligvis nettverkstopp forårsaket av økt trafikk på nettverket, noe som fører til overbelastning og pakketap. Vi kan også se at jo flere host som kommuniserer samtidig, jo høyere er latensen og jo laver gjennomstrømning.

4.7 TEST CASE 5: EFFECTS OF PARALLEL CONNECTIONS

For å utføre nettverkstesten på den angitte topologien, følger vi de neste trinnene:

1. Lag nettverkstopologien med den medfølgende konfigurasjonen, dette starter mininet-nettverksemulatoren.
6. Åpne Xterm-vinduet på server-hostene h4, h5 og h6. Sjekk IP-adressekonfigurasjonen ved å bruke «iconfig»-kommandoen. Bruk verktøyet simpleperf.py for å sette opp servermodus ved å bruke de riktige alternativene. Verktøyet beregner mengden data som mottas og dataoverføringshastigheten for et bestemt intervall. Lagre resultatene i tekstfiler når testen er ferdig for senere analyse.

2. Åpne Xterm-vinduer på klient-hostene h1, h2 og h3. Sett opp klientmodusen ved å bruke simpleperf. For klient h1, sett -P-alternativer til 2 parallellkoblinger, -t-alternativet til 25 sekunders testvarighet og -i-alternativet til 5 sekunders tidsintervall. Hvis mulig, utfør kommandoene samtidig på hver klient for å garantere multipleksing av datasending med parallelle forbindelser. Lagre resultatene i tekstfiler når testen er ferdig for senere analyse.

4.7.1 RESULTS

Resultatene oppnådd fra testene for hvert host-par er oppsummert i følgende tabell:

Sti	Gjennomstrømning (Mbps)
H1-H4 H1-H4	4.94 6.42
H2-H5	8.43
H3-H6	8.82

Fra tabellen viser resultatene at når hostene h1, h2 og h3 kommuniserer samtidig med hostene h4, h5 og h6, med to parallellkoblinger i h1, er gjennomføringen for h1-h4 lav på hver parallellforbindelse, gjennomstrømningen for h2-h5 og h3-h6 er relativt høy. Dette er fordi nettverket må håndtere mer trafikk og dermed opplever pakkene mer forsinkelse.

4.7.2 DISCUSSION

Vi kan observer at gjennomstrømningen til h1-h4- og h2-h5-forbindelsene er betydelig lavere enn gjennomstrømningen til h3-h6-forbindelsen. Dette kan skyldes at h1 åpner flere parallellforbindelser, mens h2 og h3 bare åpner en forbindelse. Å åpne flere tilkoblinger samtidig kan føre til høyere nettverkskostnader, da nettverksressursene må deles mellom flere tilkoblinger. Derfor er det mulig at h1 bruker mer nettverksressurser enn h2 og h3, noe som resulterer i lavere gjennomstrømming for tilkoblingene deres.

Hvis vi foretar en sammenlikning med resultater på testcase 3 eksperiment 2, kan vi se at når h1-h4, h2-h5 og h3-h6 kommuniserer samtidig, med parallelle forbindelser, reduseres gjennomstrømningen for h1-h4 nesten halvparten fra 7.82 Mbps til 4.94 Mbps og 6.42 Mbps for hver parallelltilkobling. Gjennomstrømning for h2-h5 og h3-h6 forblir relativt som eksperiment 2. Det er fordi data multiplekses over lenker og eksisterer med to forbindelser fra h1-klienten.

Vi konkluderer med at ventetiden til forbindelsene også kan bli påvirket av den økte nettverkstrafikken. Høyere latenstid kan resultere i langsommere responstider og generelt lavere ytelse for applikasjonene som kjører på hostene. Det er viktig å overvåke og administrere nettverksoverbelastning for å opprettholde optimal ytelse og forhindre nettverksfeil.

5 CONCLUSIONS

Rapporten, og prosjektet, tar for seg nettverksytelse og betydningen av å kunne måle ulike ytelsesparametere for å kunne oppfatte kvaliteten av tjenesten som tilbys av nettverket. De ulike resultatene og testene demonstrerer at nettverkstopologien, mengde hopp mellom server- og klienthoster, båndbreddekapasiteten i tilkoblinger, og samtidig kommunikasjon av mange host-par, kan spille en rolle i ytelsen til nettverket betydelig. Nøye vurdering av nettverkstopologien og identifikasjon av områder i trengsel om forbedring er essensielt for å kunne optimalisere ytelsen til nettverket. Det er viktig å være bevisst over faktumet om at ytelsen til nettverket i et virtuelt miljø muligens kan avvike fra ytelsen i et virkelighetsscenario grunnet diverse faktorer. I konklusjon kan det avgjøres: for å sikre pålitelighet og høy kvalitet av nettverkstjenester er forståelse og måling av nettverksytelse ekstremt viktig.

6 REFERENCES

- Topology fra professor Safiqul Islam
- Informasjon og visdom fra forelesninger generelt