Hogeschool Utrecht

Bachelor scriptie

Onderzoek naar het testen van datasnelheden tussen 40 Gb/s en 100 Gb/s

Auteur:
Arthur van der Weiden
Studentnummer: 1619815
Opleiding: HBO-ICT,
Technische informatica

Bedrijfsbegeleiders:
Tristan Suerink
Ronald Starink

Examinatoren:
Eerste examinator:
Marten Wensink
Tweede examinator:
Harry Beerlage (begeleider)

Computer technologie
Nikhef

April, 2016

Managementsamenvatting

Aanleiding

Het datacenter van Nikhef verwerkt per seconde een grote hoeveelheid data. Om de snelheid van de servers te kunnen testen wordt op dit moment een tool gebruikt die problemen heeft met het testen van snelheden die groter zijn dan 40 Gb/s. Omdat bij Nikhef datasnelheden van meer dan 100 Gb/s bereikt kunnen worden, is dit een probleem aangezien deze datasnelheden niet betrouwbaar getest kunnen worden. Deze tests worden uitgevoerd ter controle van de datasnelheid. Om dit probleem op te lossen moet een onderzoek gedaan worden naar de mogelijkheid om datasnelheden boven de 40 Gb/s te testen.

Conclusie en aanbevelingen

Het is mogelijk om een testsuite te ontwikkelen die datasnelheden van meer dan 40 Gb/s kan testen met behulp van de library DPDK. Door systeemaanroepen te omzeilen kunnen datasnelheden van 70 Gb/s getest worden.

Het is nog onduidelijk of de testsuite datasnelheden van 100 Gb/s kan bereiken. Dit kwam doordat het niet lukte om de driver van de netwerkkaart die beschikbaar gesteld was voor het testen van deze snelheden in combinatie met DPDK te gebruiken. Daarom wordt aanbevolen om contact op te nemen met de fabrikant van deze netwerkkaart met de vraag waarom dit niet mogelijk was, of om hulp te vragen bij het combineren van DPDK en de driver. Als de fabrikant geen oplossing heeft, is het ook mogelijk om zelf een driver te programmeren. Ook kan gekeken worden naar een andere netwerkkaart die deze snelheden kan bereiken en een DPDK driver kan gebruiken. Ook is het mogelijk om de testsuite te gebruiken met twee netwerkkaarten van 40 Gb/s en twee netwerkkaarten van 10 Gb/s. Op deze manier kan getest worden of de testsuite datasnelheden van 100 Gb/s kan bereiken.

Ook moet, om een beeld te krijgen van de bandbreedte, een test op twee machines gestart worden. De reden hiervan is dat het niet is gelukt om de ontvanger van de datapakketten feedback te laten sturen naar de zender van de datapakketten. Dit kwam door tijdgebrek

en andere functionaliteiten die prioriteit over deze functionaliteit hadden. Als Nikhef deze functionaliteit toch wil, zal deze later toegevoegd moeten worden.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Hoofdstuk 1

Inleiding

In deze scriptie wordt het onderzoek naar het meten van datasnelheden tussen 40 Gb/s en 100 Gb/s beschreven. Dit onderzoek is uitgevoerd bij Nikhef. Nikhef is het Nationaal Instituut voor Subatomaire Fysica in Amsterdam, waar fundamenteel onderzoek naar materie wordt gedaan. Om dit onderzoek te kunnen ondersteunen heeft Nikhef een ICT-afdeling waar onder andere onderzoek gedaan wordt naar nieuwe technieken. Ook worden op deze afdeling de servers onderhouden.

1.1 Aanleiding

Het datacenter van Nikhef verwerkt per seconde een grote hoeveelheid data, daarom zijn bij Nikhef datasnelheden tot 100 Gb/s noodzakelijk. Wanneer nieuwe hardware ingekocht wordt, waarmee deze snelheden haalbaar zouden moeten zijn, wil Nikhef controleren of de, door de leverancier beloofde snelheden ook daadwerkelijk behaald worden. Op dit moment wordt deze controle uitgevoerd met een tool die problemen heeft om vanaf één machine snelheden van meer dan 40 Gb/s te testen. Omdat bij Nikhef datasnelheden van meer dan 100 Gb/s bereikt kunnen worden, is dit een probleem aangezien deze datasnelheden niet betrouwbaar getest kunnen worden. Om dit probleem op te lossen moet een onderzoek gedaan worden naar de mogelijkheid om datasnelheden boven de 40 Gb/s met slechts één machine te testen. Nikhef wilt deze snelheden met één machine kunnen testen omdat de controles van de hardware dan betrouwbaarder zijn.

1.2 Doel van Nikhef

Nikhef verlangt een testsuite waarmee de bandbreedte tussen twee machines getest kan worden. De twee machines zullen hardware bevatten waar Nikhef het theoretisch maximum van wilt controleren. Met deze controle wil Nikhef erachter komen of de specificaties van de hardware, het theoretische maximum van de hardware, daadwerkelijk overeen komen met de praktijk. De testsuite die ontwikkeld gaat worden, moet dit dus mogelijk gaan maken.

1.3 Leeswijzer

In hoofdstuk 2 wordt de achtergrond van het onderzoek besproken. Hierin staat beschreven wat Nikhef doet en hoe de structuur van Nikhef in elkaar steekt. Verder staat in dit hoofdstuk de achtergrond van de opdracht beschreven, zijn hier de hoofd- en deelvragen te vinden en staat hier beschreven wat de verwachte resultaten van het onderzoek zijn. In hoofdstuk 3 staat beschreven wat de specifieke eisen aan de testsuite zijn. Ook staat hierin beschreven welke libraries/tools onderzocht zijn. In hoofdstuk 4 staat het vooronderzoek naar iPerf beschreven. iPerf is de tool die op dit moment bij Nikhef gebruikt wordt om dataverkeer te testen. De gebruikte versies bij dit onderzoek zijn: iPerf 2 en iPerf 3. Hoofdstuk 5 gaat over de besturingssysteemrechten die de verschillende libraries/tools nodig hebben om optimaal te kunnen werken. Vervolgens wordt in hoofdstuk 6 het onderzoek naar de beste library/tool beschreven. In hoofdstuk 7 is het ontwerp van de testsuite te vinden. Hoofdstuk 8 beschrijft de werking van de beste library/tool voor het onderzoek. Hoofdstuk 9 bespreekt de resultaten van het onderzoek. In dit hoofdstuk staat beschreven in welke volgorde het onderzoek is uitgevoerd. Verder worden hier de keuzes die tijdens het onderzoek gemaakt zijn verantwoord. De projectorganisatie is te vinden in hoofdstuk 10. In dit hoofdstuk is de communicatie verantwoord, staat beschreven aan welke eisen is voldaan en wordt de planning verantwoord. In hoofdstuk 11 is de conclusie te vinden. De aanbevelingen zijn te vinden in hoofdstuk 12 en in hoofdstuk 13 is een evaluatie van het project te vinden.

Hoofdstuk 2

Achtergrond

2.1 Nikhef

De afstudeeropdracht is uitgevoerd bij Nikhef. Nikhef is het Nationaal Instituut voor Subatomaire Fysica in Amsterdam-Oost, waar voornamelijk onderzoek gedaan wordt naar subatomaire fysica. De afstudeeropdracht is uitgevoerd op de afdeling Computer Technologie (hierna CT). Op deze afdeling zijn ongeveer 20 werknemers werkzaam. CT heeft een ondersteunende rol in Nikhef, en het doel van deze afdeling is om de wetenschappers hun onderzoek zo soepel mogelijk te laten uitvoeren. De afdeling houdt zich onder andere bezig met het netwerk en vernieuwingen op ICT gebied. Op de afdeling wordt ook onderzoek gedaan naar nieuwe en betere ICT mogelijkheden. Ook houdt een deel van de afdeling zich bezig met het Grid. Dit is een netwerk van computers over de hele wereld die complexe berekeningen uit kunnen voeren (*Nikhef*, 2016).

2.2 Opdracht en kwestie

2.2.1 Opdracht

De opdracht is om een onderzoek uit te voeren naar de mogelijkheden voor een testsuite, waarmee datasnelheden tussen de 40 Gb/s en 100 Gb/s getest kunnen worden. Verder is de opdracht het ontwerpen en ontwikkelen van de testsuite als proof of concept. Allereerst wordt onderzocht welke libraries/tools beschikbaar zijn die dit mogelijk maken. Vervolgens kan met die library/tool onderzocht worden of deze snelheden mogelijk zijn. De testsuite moet ervoor zorgen dat de hardware die gecontroleerd moet worden, gecontroleerd kan worden met slechts twee machines. Van deze twee machines heeft een machine de rol ontvanger van de data en de andere machine de rol van de verzender van de data. De tool die Nikhef nu gebruikt, iPerf, heeft meerdere machines nodig om een test uit te kunnen voeren, dit komt doordat het iPerf niet lukt om met één machine genoeg data te laten genereren om de juiste hoeveelheid data te kunnen versturen.

2.2.2 Kwestie

Het testen van de transfersnelheid gebeurt op dit moment met iPerf. Met iPerf kan de maximale bandbreedte van een netwerk getest worden. iPerf kan echter niet vanaf één machine data genereren en versturen wanneer de snelheid hoger dan 40 Gb/s is. Omdat in de toekomst netwerkkaarten van 100 Gb/s en machines die deze snelheden aankunnen regelmatig voor zullen gaan komen (*Green*, 2016) zullen deze snelheden wel getest moeten kunnen worden. Nikhef wil deze snelheden vanaf één machine kunnen versturen, in paragraaf 2.2.4 wordt uitgelegd waarom. De tests zullen worden uitgevoerd om de specificaties van aangekochte hardware te kunnen controleren. Dit controleren is van belang, omdat fabrikanten als doel hebben hun netwerkkaarten te verkopen en dus de specificaties mooier kunnen laten lijken dan ze in werkelijkheid zijn. Nikhef wil dus een controle uitvoeren, zodat geen dure hardware gekocht wordt, die niet aan de beloofde specificaties voldoet. Deze kwestie wordt door Nikhef opgepakt, omdat de netwerken nauwkeuriger getest moeten worden en nog geen onderzoeken, en tools, naar het testen van deze snelheden bekend zijn.

2.2.3 Doelstellingen

Nikhef heeft als doel om datasnelheden van meer dan 40 Gb/s te kunnen testen, om de specificaties van de aangekochte hardware te kunnen controleren. Het doel van de afstudeeropdracht is daarom om een onderzoeksrapport op te leveren met de mogelijkheden voor het testen. Verder moet een testsuite ontwikkeld worden als proof of concept. Het onderzoeksrapport zal gaan over de mogelijkheden betreffende het testen van datasnelheden van meer dan 40 Gb/s tot mogelijk 100 Gb/s. Deze doelstellingen zijn afgeleid uit de problemen die zijn beschreven in paragraaf 2.2.2. De doelstelling zal behaald zijn wanneer met een proof of concept bewezen wordt dat datasnelheden van meer dan 40 Gb/s bereikt kunnen worden. Deze doelstelling lost de kwestie op door het mogelijk maken van het testen van datasnelheden van meer dan 40 Gb/s.

2.2.4 Afbakening

Tijdens het uitvoeren van de opdracht zal een onderzoeksrapport opgeleverd worden (bevat in dit document). Ook wordt een proof of concept gerealiseerd die zal bewijzen of de gewenste snelheden haalbaar zijn. Het proof of concept zal alleen bewijzen of de snelheden vanaf één machine verstuurd kunnen worden. De reden dat de data vanaf slechts één machine verstuurd mag worden is dat de timing van het starten van de test dan beter is dan wanneer de test vanaf meerdere machines gestart moet worden. Het proof of concept zal dus niets bewijzen over een specifieke netwerkkaart of specifieke hardware, noch zal het onderzoeksrapport focussen op het behalen van deze snelheden met een specifieke netwerkkaart.

2.3 Theoretisch kader

2.3.1 Literatuur

De belangrijkste literatuur die onderzocht dient te worden is de documentatie van de verschillende libraries/tools. Uit deze documentatie zal blijken welke library/tool het nuttigst is voor het project. Deze documentatie laat namelijk zien welke libraries/tools (eenvoudig) uit te breiden zijn. Verder zullen wetenschappelijke artikelen en andere rapporten opgezocht worden waaruit de potentie voor het behalen van hogere snelheden van de verschillende libraries/tools zal moeten blijken. Deze bronnen zullen gebruikt worden om een goede keuze tussen de verschillende libraries/tools te kunnen maken.

2.3.2 Begrippen

Datapakketten: Datapakketten zijn netwerkpakketten die, in het kader van dit onderzoek, gegenereerd worden. De snelheid en grootte van deze datapakketten zal ervoor zorgen dat de verbinding volstroomt met data.

Data Plane Development Kit: Data Plane Development Kit (DPDK) is een library die het mogelijk maakt om invloed op een netwerk uit te oefenen.

Gb/s: Gb/s staat voor Gigabit per seconde en is de eenheid van de snelheid van het dataverkeer.

Kernel Network-interface-card Interface: Een KNI is een interface die tussen de Linux kernel en de user-mode applicatie geïnstalleerd wordt en ervoor zorgt dat de user-mode applicatie, kernel-mode instructies uit mag voeren (*Moergestel*, 2012).

Netwerkprotocol: Een netwerkprotocol is een verzameling afspraken op een netwerk. Communicerende nodes zullen zich aan deze afspraken moeten houden, zodat de communicatie goed kan verlopen (*Kurkose & Ross*, 2003).

SCTP: SCTP, of Stream Control Transmission Protocol, is een netwerkprotocol met de garantie dat de verstuurde data aankomt. SCTP werkt met datapakketten (*Kurkose & Ross*, 2003).

TCP: TCP, of Transmission Control Protocol, is een netwerkprotocol met de garantie dat de verstuurde data aankomt. TCP werkt met datastromen (*Kurkose & Ross*, 2003).

TX/RX: TX en RX zijn afkortingen die staan voor Transmit en Receive. De termen TX-burst of RX-burst betekenen: het aantal datapakketten dat per burst verstuurd of

ontvangen kan worden (*Pktgen ontwikkelaars*, 2015).

UDP: UDP, of User Datagram Protocol, is de directe tegenhanger van TCP en biedt geen aankomst garantie voor de verstuurde datagrammen. UDP werkt met datagrammen (*Kurkose & Ross*, 2003).

2.3.3 Modellen

Om de eisen van het project te verdelen voor de ontwikkelfase zal MoSCoW worden gebruikt. MoSCoW is een model waarbij de eisen verdeeld worden in *Must-haves*, *Should-have*, *Could-haves*, *Won't-haves*. Deze verdeling verwijst naar de prioriteiten van de verschillende eisen. De verdeling is gesorteerd op prioriteit met de *Must-haves* als hoogste prioriteit en de *Won't-haves* als laagste prioriteit (*Osch*, 2014). Om een beeld te krijgen van de uiteindelijke functionaliteiten en werking van de testsuite zijn een use case diagram, een sequentiediagram en een communicatiediagram gemaakt. Deze diagrammen zijn te vinden in bijlage F. Een use case diagram helpt de ontwerper van een systeem de uiteindelijke functies van het te ontwikkelen systeem in kaart te brengen (*Velthoven*, 2013). Een sequentiediagram helpt de ontwikkelaar van een systeem bij het bepalen van de volgorde waarin bepaalde functies van het systeem uitgevoerd dienen te worden (*Bell*, 2004). Een communicatiediagram wordt gebruikt voor het in kaart brengen van de communicatie tussen de verschillende onderdelen van een systeem (*Nishadha*, 2012).

2.3.4 Hoofdvraag

De hoofdvraag van dit onderzoek is de volgende:

Wat is de beste library of tool om een testsuite te ontwikkelen waarmee datasnelheden tot ongeveer 100 Gb/s getest kunnen worden waarbij de data vanaf slechts één machine verstuurd wordt?

2.3.5 Deelvragen

De deelvragen die bij deze hoofdvraag horen zijn de volgende:

1. **Hoe werkt de tool, iPerf, die op dit moment gebruikt wordt voor het testen van serversnelheden?**
   Het doel van deze deelvraag is om als vooronderzoek te dienen. De functie van deze deelvraag is descriptief, omdat deze deelvraag de werking van de tool iPerf verklaart. De onderzoeksmethoden die bij deze deelvraag gebruikt gaan worden zijn observatie en een literatuuronderzoek. Deze onderzoeksmethoden zijn gekozen, omdat op deze manier de deelvraag zo volledig mogelijk beantwoord kan worden.

2. **Wat zijn de specifieke eisen aan de testsuite, naast de te bereiken testsnelheden?**
   Deze deelvraag moet beantwoord worden voordat de rest van het onderzoek kan beginnen. Deze eisen bepalen namelijk welke library/tool als beste naar voren komt. Deze deelvraag is prescriptief, hij helpt namelijk bij het ontwerpen van de testsuite. Om antwoord op deze deelvraag te krijgen zal een interview met de bedrijfsbegeleider gehouden worden. Deze onderzoeksmethode is gekozen, omdat de bedrijfsbegeleider de opdrachtgever is en daarom de eisen aan de opdracht heeft bedacht.

3. **Wat zijn de besturingssysteem rechten die de library/tool nodig heeft om goed te kunnen functioneren?**
   Deze deelvraag gaat verder in op de eisen die door de vorige deelvraag aan de testsuite gesteld worden. Deze deelvraag is prescriptief, omdat de rechten van belang zijn tijdens het ontwerpen van de testsuite. Bij deze deelvraag wordt een literatuuronderzoek uitgevoerd. Uit de documentatie zal blijken welke rechten de library/tool nodig heeft om goed te functioneren.

4. **Wat is de beste library/tool om te gebruiken bij het ontwikkelen van de testsuite?**
   Deze deelvraag dient om een keuze te maken tussen de gevonden libraries/tools. Het antwoord op deze deelvraag is namelijk de library/tool die gebruikt zal worden tijdens het vervolg van het onderzoek. Deze deelvraag is descriptief, omdat tijdens het beantwoorden van de deelvraag de verschillende libraries/tools met elkaar vergeleken zullen worden. Om antwoord te krijgen op deze deelvraag wordt gekeken naar de antwoorden op de vorige deelvragen. De eisen die aan de library/tool gesteld worden, zullen worden gehaald uit het antwoord van deelvraag 2.

5. **Hoe werkt de library/tool die als beste uit het voorgenoemde onderzoek komt precies?**
   De beste library/tool wordt bepaald aan de hand van een aantal eisen. Wat deze eisen precies zijn, is te vinden in hoofdstuk 4. Om een testtool te kunnen ontwikkelen met behulp van de library/tool moet onderzocht worden hoe deze library/tool precies werkt. Om daar achter te komen dient deze deelvraag beantwoord te worden. Deze deelvraag is descriptief, omdat deze deelvraag de werking van de beste tool verklaard. Bij deze deelvraag wordt gebruik gemaakt van de onderzoeksmethoden observatie en literatuuronderzoek. De reden van deze onderzoeksmethode is, dat in de literatuur van de library/tool de werking van de library/tool beschreven is.

6. **Welke aspecten heeft het proof of concept nodig om geslaagd te zijn?**
   Deze deelvraag zal helpen bij het ontwerpen van het proof of concept. In die zin dat het proof of concept dan aan alle eisen zal voldoen om geslaagd te zijn. Of, als dit niet binnen het kader mogelijk is, een verder onderzoek gestart kan worden naar alternatieve oplossingen. Deze deelvraag wordt beantwoord met behulp van

het interview beschreven bij deelvraag 2. Deze onderzoeksmethode is gekozen, omdat de bedrijfsbegeleider tevens de opdrachtgever is en daarom de eisen aan de opdracht heeft bedacht.

2.4 Aanpak

In deze paragraaf wordt de aanpak van het onderzoek beschreven. Hierin staan de onderdelen: onderzoeksmethoden en het theoretisch kader.

2.4.1 Methoden en deelresultaten

Het eindresultaat zal bereikt worden door de opdracht te verdelen in sprints. Tijdens deze sprints is het de bedoeling om de behaalde snelheid steeds te verhogen. Uiteindelijk zal op deze manier dus, mits dit mogelijk is, meer dan 40 Gb/s bereikt worden. In de eerste sprint zal de library/tool klaargemaakt worden om mee te kunnen ontwikkelen. Hiermee wordt bedoeld dat de broncode op de computer getest is met behulp van een testprogramma. Vervolgens wordt een programma ontwikkeld waarmee 1 Gb/s behaald kan worden. Door het ontwikkelen van dit programma zal kennis over de opbouw van de broncode opgedaan worden. Vervolgens wordt dit programma uitgebreid naar datasnelheden van 10 Gb/s. Voor deze snelheid is gekozen, omdat 10 Gb/s zeker haalbaar is, dit was immers lange tijd de standaard. Ook wordt hiermee de basis gelegd voor de uiteindelijke snelheden van meer dan 40 Gb/s. Deze snelheden worden in de laatste twee sprints ontwikkeld. In de vierde sprint wordt namelijk onderzoek gedaan naar het genereren en testen van snelheden van 40 Gb/s. Deze tussenstap is gekozen, omdat 40 Gb/s de huidige standaard is. Omdat dit nu de standaard is, is ook deze snelheid zeker haalbaar. Na het behalen van 40 Gb/s is de stap naar meer dan 40 Gb/s niet groot meer. Het laatste gedeelte van het onderzoek zal gaan over snelheden van meer dan 40 Gb/s. Wanneer deze snelheden behaald zijn is de kwestie opgelost. Nikhef maakt zelf geen gebruik van een bepaalde methode, daarom maakt het voor de organisatie niet uit welke methoden toegepast worden. Er is voor een vorm van scrum gekozen omdat de voortgang van het project eenvoudig bij te houden is (*Overheem*, 2004). De deelresultaten zijn: een programma waarmee datasnelheden van 1 Gb/s behaald kunnen worden, een programma waarmee datasnelheden van 10 Gb/s behaald kunnen worden, een programma waarmee datasnelheden van 40 Gb/s behaald kunnen worden en een programma waarmee datasnelheden van 40+ Gb/s behaald kunnen worden. Deze deelresultaten zullen een resultaat opleveren waarmee aan de doelstelling wordt voldaan. Het laatste deelresultaat is namelijk een testsuite waarmee datasnelheden van meer dan 40 Gb/s behaald kunnen worden. Omdat het waarschijnlijk eenvoudiger is om de behaalde snelheid steeds te verhogen, dan om in één keer meer dan 40 Gb/s te bereiken, is gekozen voor deze deelresultaten.

2.4.2 Criteria

In tabel 2.1 zijn de eisen verdeeld volgens de MoSCoW methode. Deze eisen zijn voortgekomen uit het interview met de opdrachtgever, zoals beschreven in paragraaf 3.1.1 en uitgewerkt in bijlage B, en de beschrijving van de opdracht. Omdat de eindgebruiker dezelfde persoon is als de opdrachtgever, werd met dit interview direct onderzocht wat de eisen van de eindgebruiker zijn.

Tabel 2.1: Eisen aan het eindproduct verdeelt volgens de MoSCoW methode

MSCW	**Onderdeel en eis**
Must-have	**Onderzoeksverslag:** Beste library/tool gekozen **Testsuite:** Meer dan 40 Gb/s Werkend onder bekende Linux distributies UDP en TCP testen
Should-have	**Onderzoeksverslag:** n.v.t. **Testsuite:** Tot 100 Gb/s Werkend onder andere Unix producten Negeren van TCP slow start Veranderen van eenheid uitvoer
Could-have	**Onderzoeksverslag:** n.v.t. **Testsuite:** Feedback van de ontvanger naar de zender
Won't-have	**Testsuite:** Windows ondersteuning

2.5 Verwachte resultaten

Omdat de ontwikkelaars van de netwerkkaarten hun producten zelf ook hebben getest, om dezelfde reden als dat Nikhef het zal gaan testen, is de verwachting dat snelheden van meer dan 40 Gb/s behaald kunnen worden. Of dit met slechts één machine kan blijft onzeker. Verder wordt verwacht, vanwege vooronderzoek van Nikhef, dat DPDK een goede kandidaat is om de benodigde snelheden te halen. DPDK is een groep libraries waar onder andere datapakketten mee gegenereerd kunnen worden. Daarom zal DPDK zeker onderzocht worden.

Hoofdstuk 3

Specifieke eisen en te onderzoeken libraries/tools

In dit hoofdstuk worden de eisen aan de testsuite besproken. Verder zal beschreven worden welke libraries/tools onderzocht zullen worden en wat de eisen aan de verschillende libraries/tools zullen zijn tijdens het onderzoek naar de beste library/tool. De testsuite zal uiteindelijk moeten aantonen of het theoretisch maximum van het netwerk wordt bereikt. Dat wil zeggen: als een netwerk volgens de specificaties 10 Gb/s kan halen dan moet de testsuite laten zien of dit echt zo is. Dit zal de testsuite doen door op de zender en de ontvanger de doorgangssnelheid te meten. Als dit zowel bij de zender als bij de ontvanger 10 Gb/s is, dan is het theoretisch maximum dus bereikt en is het netwerk in orde.

3.1 Eisen aan de testsuite

3.1.1 Onderzoeksmethode

Om erachter te komen wat de specifieke eisen aan de testsuite zouden zijn, is een interview gehouden met de opdrachtgever, die tevens de bedrijfsbegeleider is. Het volledig uitgewerkte interview is te vinden in bijlage B. De vragen voor het interview waren:

1. Wanneer is het proof of concept geslaagd?
   Deze vraag is gesteld om de minimale eisen aan het proof of concept te kunnen achterhalen. Daarmee wordt bedoeld: wat moet het proof of concept minimaal kunnen om als bewijs beschouwd te worden.

2. Op welke besturingssystemen moet de testsuite kunnen werken?
   Niet alle machines die getest gaan worden, kunnen dezelfde besturingssystemen draaien. Om erachter te komen welke besturingssystemen meestal gebruikt zullen worden, is deze vraag gesteld.

3. Welke protocollen moeten getest kunnen worden? (TCP/UDP/SCTP)
   Netwerken gebruiken verschillende protocollen voor de verschillende applicaties die mogelijk op de netwerken kunnen draaien. Dit betekent dat ook verschillende protocollen getest moeten kunnen worden, omdat het theoretisch maximum met alle protocollen behaald dient te worden en de testsuite deze protocollen moet kunnen gebruiken.

4. Welke iPerf opties moeten terugkomen in de testsuite?
   Omdat iPerf de huidige testtool van Nikhef is, zijn er waarschijnlijk opties die iPerf ondersteunt die de gebruikers graag terug zien komen in de nieuwe testsuite.

5. Zijn er nog andere eisen die niet behandeld zijn met deze vragen?
   Deze vraag was voor dekking van eventuele vergeten eisen.

6. Mag de testsuite root rechten vereisen?
   Deze vraag was, achteraf gezien, relatief irrelevant ten opzichte van de rest van de gestelde vragen, omdat iedereen die de applicatie gaat gebruiken root-rechten heeft. Echter omdat Nikhef veel stagiair(e)s heeft, kan het zo zijn dat iemand zonder root rechten ook de applicatie zou gaan gebruiken. Om erachter te komen of dit ook het geval was, werd deze vraag gesteld.

3.1.2 Eisen

De eisen die uit het interview naar voren kwamen waren:

- De testsuite moet werken onder de bekende Linux distributies, als: Ubuntu en CentOS;
- De protocollen die getest moeten kunnen worden zijn: TCP, UDP en het moet mogelijk zijn om verschillende poortnummers te gebruiken;
- De testsuite moet:
  - de maximale snelheid kunnen testen;
  - slow start kunnen negeren;
  - UDP kunnen testen zonder een server te starten.

Uit het interview kwam naar voren dat het proof of concept geslaagd is wanneer snelheden tussen 40 Gb/s en 100 Gb/s gegenereerd en dus getest kunnen worden. Dit is de enige eis aan het proof of concept, omdat het proof of concept alleen dient om te bewijzen dat de gebruikte library/tool snelheden van meer dan 40 Gb/s kan genereren en dus testen.

3.2 Onderzoek libraries/tools

Een library/tool wordt alleen onderzocht als hij voldoet aan zogenaamde "knock-out" eisen. Omdat de uiteindelijke testsuite op Linux moet werken, was de eerste knock-out eis dus dat de library/tool door Linux ondersteund wordt en zelf ook Linux ondersteunt. Ook moet de broncode openbaar zijn en open-source. Dat wil zeggen dat de code vrij aan te passen moet zijn. Verder moeten ze de mogelijkheid bieden tot het genereren van datapakketten of, in geval van tools, moeten ze dit al kunnen. Als een library/tool niet aan alle drie deze eisen voldoet, wordt hij niet verder onderzocht. De libraries/tools die niet aan de knock-out eisen voldeden zijn beschreven in paragraaf 3.2.1.

3.2.1 Te onderzoeken libraries/tools

De libraries/tools die onderzocht gaan worden zijn: iPerf/iPerf3 (een tool om de bandbreedte van een netwerk te testen), DPDK (een library die het mogelijk maakt een netwerk op verschillende manieren te manipuleren), Netcat (een tool om o.a. de bandbreedte van een netwerk te testen) en Web10G (een library die het mogelijk maakt om datapakketten te versturen met maximaal 10 Gb/s). Voor deze libraries/tools is gekozen omdat ze voldoen aan alle knock-out-out eisen. In paragraaf 3.2.2 wordt van iedere library/tool een korte uitleg gegeven. Voorbeelden van tools/libraries die gevonden zijn maar niet aan alle knock-out eisen voldoen zijn: NetStress, TotuSoft LAN Speed Test, NetIO-GUI en AIDA32. Hieronder wordt kort uitgelegd aan welke knock-out eisen deze tools niet voldoen.

NetStress
NetStress is een tool waarmee onder andere de bandbreedte van een netwerk getest kan worden. NetStress is alleen voor verschillende versies van Windows beschikbaar en de code van NetStress is niet beschikbaar, daarmee voldoet NetStress niet aan de eisen: De library/tool moet Linux ondersteunen en de broncode moet openbaar en open-source zijn (*Nuts about Nets*, 2016).
TotuSoft LAN Speed Test
TotuSoft is, net zoals NetStress, een tool waarmee de bandbreedte van een netwerk getest kan worden. Van deze tool zijn verschillende versies beschikbaar, de versie die bekeken is, is de gratis versie. Dit maakte echter verder niet uit, geen van alle versies ondersteunde namelijk Linux. Ook is de broncode niet beschikbaar en daarom kan ook deze tool niet uitgebreid worden (*Totusoft*, 2016).
NetIO-GUI
NetIO-GUI is een tool waarmee onder andere de bandbreedte van een netwerk getest kan worden. NetIO-GUI is een uitbreiding voor de command-line tool NetIO en heeft als toevoeging een grafische interface. Van NetIO-GUI is de broncode niet beschikbaar. Dit betekent dat deze tool niet uitgebreid kan worden (*Markus*, 2015).

AIDA32
Ook AIDA32 is een applicatie waarmee de bandbreedte van een netwerk getest kan worden, die echter ook alleen voor Windows beschikbaar is (*Miklos*, 2004). Dit betekent dat AIDA32 niet voldoet aan de knock-out eis: de library/tool moet Linux ondersteunen.

3.2.2 Uitleg libraries/tools

3.2.2.1 iPerf

iPerf is een tool waarmee de bandbreedte van een netwerk getest kan worden. Dit doet iPerf door datapakketten te genereren en te meten met welke snelheid dat genereren gebeurt en op de ontvangende machine te meten met welke snelheid de datapakketten binnenkomen.

3.2.2.2 DPDK

DPDK is een groep libraries die het mogelijk maken om een netwerk op verschillende manieren te manipuleren. Zo is het mogelijk om datapakketten te genereren met de maximaal mogelijke snelheid die de netwerkkaarten aankunnen. Door gebruik te maken van deze mogelijkheid kan een tool ontwikkeld worden met dezelfde functionaliteit als iPerf.

3.2.2.3 Netcat

Netcat is een tool waarmee verschillende soorten netwerktests uitgevoerd kunnen worden. Zo kan met deze tool onder andere de bandbreedte van een netwerk getest worden. Netcat doet dit door een bestand, gemaakt door de gebruiker, te versturen over het netwerk en de bandbreedte die beschikbaar was tijdens het versturen te meten.

3.2.2.4 Web10G

Web10G is een library die het mogelijk maakt om een tool te maken die datapakketten genereert met een snelheid van 10 Gb/s, waardoor het mogelijk wordt om de datasnelheid te meten en dus een netwerk te testen. Met deze mogelijkheid kan dus een tool gemaakt worden die dezelfde functionaliteit als iPerf heeft.

3.2.3 Eisen aan de libraries/tools

In deze paragraaf staan de eisen beschreven. Aan iedere eis hangt een puntenaantal. Met dit puntenaantal wordt uiteindelijk per library/tool bepaald welke library/tool gebruikt gaat worden voor het onderzoek. Voor iedere eis kan minimaal één punt en maximaal vijf punten behaald worden, tenzij dit anders wordt aangegeven bij het definiëren van de eis. Omdat sommige eisen belangrijker zijn dan anderen wordt per eis ook een weging meegegeven. Het puntenaantal wordt dan vermenigvuldigd met de weging en de som

van die vermenigvuldigingen wordt uiteindelijk gebruikt voor de keuze tussen de verschillende libraries/tools. De weging, toegewezen aan de eisen, hangt af van het belang van de eis ten opzicht van het volledige onderzoek. Dat wil zeggen, een eis die nodig is om de doelstelling te behalen zal een hogere weging hebben dan een eis die alleen het eenvoud van ontwikkelen moet waarborgen. De library/tool met de meeste punten na het vermenigvuldigen met de weging gaat uiteindelijk gebruikt worden voor het onderzoek.

3.3 Eisen

1. **Broncode moet duidelijk zijn**
   Omdat de broncode naar alle waarschijnlijkheid uitgebreid dient te worden, moet de broncode duidelijk zijn geschreven. De term duidelijk betekent in dit geval: de namen van de functies moeten direct aangeven waar de functies voor dienen, de namen van de variabelen moeten direct aangeven wat in de variabelen wordt opgeslagen en het liefst moet de broncode voorzien zijn van documentatie. De verdeling van de punten bij deze eis is als volgt: één punt wanneer van de library/tool geen documentatie beschikbaar is (geen API etc.), geen commentaar in de broncode bevat en de functienamen nietszeggend zijn en vijf punten wanneer de broncode voorzien is van een duidelijke API, de broncode voorzien is van commentaar en van de library/tool veel documentatie beschikbaar is. De weging van deze eis is 1, omdat de eis alleen het eenvoud van ontwikkelen waarborgt. Deze eis is dus niet belangrijk om de opdracht te kunnen afronden. Echter omdat de opdracht binnen een bepaalde tijd moet worden afgerond, is deze eis wel van belang, omdat hij bijdraagt aan de snelheid van het ontwikkelen.

2. **De library/tool moet potentieel de 40-100 Gb/s kunnen halen**
   Omdat de testsuite uiteindelijk deze snelheden moet kunnen bereiken, heeft het geen nut om een library/tool zonder potentie voor deze snelheden te bekijken of gebruiken. Deze eis had ook als knock-out eis gebruikt kunnen worden. Echter omdat het niet direct zichtbaar is, en dus een literatuuronderzoek naar dit onderwerp uitgevoerd moet worden, is deze eis niet als knock-out eis gebruikt. Het aantal punten wordt bij deze eis bepaald door de hoeveelheid literatuur die laat zien dat het mogelijk is. Als literatuur niets zegt over de prestaties van een library/tool betekent dat niet dat het onmogelijk is om meer dan 40 Gb/s te kunnen halen. Daarom is in zo een geval drie punten het minimale aantal punten voor deze eis. Op deze manier krijgen de minder bekende libraries/tools ook een kans. Vooraf wordt niet op bekendheid gefilterd, omdat minder bekende libraries/tools best de mogelijkheid kunnen bezitten om deze snelheden te bereiken. Deze eis heeft een weegfactor van 2, omdat deze eis helpt bij het waarborgen van het behalen van de doelstelling. De verdeling van de punten bij deze eis is als volgt: Als geen onderzoeken gedocumenteerd zijn: drie punten; Als onderzoeken gedocumenteerd zijn die geen hard bewijs leveren van 40 Gb/s wel de suggestie wekken dat het mogelijk

is, of dit van meer dan één machine verstuurd wordt: vier punten; Als onderzoeken gedocumenteerd zijn die bewijzen dat 40+ Gb/s mogelijk is: vijf punten. Wanneer literatuur gevonden wordt waarin expliciet staat aangegeven dat het niet mogelijk is om met de specifieke library/tool meer dan 40 Gb/s te behalen, dan krijgt de library/tool slechts 1 punt voor deze eis.

3. **De library/tool moet overzichtelijke uitvoer genereren.**
   Als de testsuite straks een onoverzichtelijke uitvoer genereert, dan zijn de tests onhandig om uit te voeren. Daarom moet de library/tool eenvoudige en overzichtelijke uitvoer genereren of de library/tool moet zo uit te breiden zijn dat eenvoudig de uitvoer bepaald kan worden. Het woord overzichtelijk moet worden gelezen als: een precieze bandbreedte of de mogelijkheid tot het berekenen van een precieze bandbreedte. Als de library/tool standaard geen uitvoer genereert, is vier punten het maximum aantal punten voor deze eis. De hoeveelheid hangt dan af van de hoeveelheid mogelijkheden voor het toevoegen van de uitvoer. Deze eis heeft een weegfactor van 0,5. Dat komt doordat, mocht de uitvoer van de library/tool minder overzichtelijk zijn, dan kan dit aangepast worden in de code van het programma.

3.4 Samenvatting

In dit hoofdstuk zijn de eisen aan de testsuite en aan het proof of concept beschreven. Om achter deze eisen te komen is een interview met de opdrachtgever/bedrijfsbegeleider gehouden. Het volledig uitgewerkte interview is te vinden in bijlage B. Daarna werden de te onderzoeken libraries/tools besproken en de eisen die aan de libraries/tools gesteld zullen worden. De libraries/tools die onderzocht zullen worden zijn: iPerf/iPerf3, DPDK, Netcat en Web10G. De libraries/tools die niet onderzocht gaan worden omdat ze niet aan alle knock-out eisen voldoen zijn: NetStress, TotuSoft LAN Speed Test, NetIO-GUI en AIDA32. Voor de eis *De library/tool moet potentieel de 40-100 Gb/s kunnen halen* kan maximaal 5 punten gehaald worden en geldt een weegfactor van 1. De eis *De library/tool moet potentieel de 40-100 Gb/s kunnen halen* is ook maximaal 5 punten waard. Voor deze eis geldt een weegfactor van 2. De laatste eis, *De library/tool moet overzichtelijke uitvoer genereren* is ook maximaal 5 punten waard en heeft een weegfactor van 0,5.

Hoofdstuk 4

Werking huidige testtool: iPerf

In dit hoofdstuk wordt de werking van de huidige testtool, iPerf, besproken. De werking van iPerf is onderzocht door middel van een literatuuronderzoek en een praktijkonderzoek. iPerf is de tool die Nikhef op dit moment gebruikt om de bandbreedte van een netwerk te testen. Dit onderzoek wordt uitgevoerd, zodat iPerf als referentie tool gebruikt kan worden. Als tijdens het onderzoek naar de beste library/tool blijkt dat iPerf meer potentie heeft dan in eerste instantie gedacht wordt, zal iPerf verder onderzocht worden.

4.1 iPerf

iPerf is een tool om de bandbreedte van een netwerk mee te testen. Dit doet iPerf door datapakketten te genereren met een snelheid die de netwerkkaart maximaal aankan. Door vervolgens te meten met welke snelheid de datapakketten bij de ontvangende node aankomen, creëert iPerf een beeld van de bandbreedte die het netwerk kan bereiken. De uiteindelijke testsuite zal dit ook op deze manier moeten gaan doen. Om de bandbreedte van een netwerk te kunnen testen moeten namelijk datapakketten gegenereerd worden. Dit genereren moet gebeuren op de maximale snelheid van het netwerkkaart, zodat het limiet van het netwerk duidelijk zichtbaar wordt. iPerf biedt mogelijkheden om de tests zoveel mogelijk te finetunen. Op deze manier kan een nauwkeuriger beeld van de bandbreedte gegenereerd worden. Een iPerf test genereert per seconde de behaalde bandbreedte en aan het einde van de test wordt de gemiddeld gemeten bandbreedte gegeven. Op deze manier kan de gebruiker dus zien of de gemeten bandbreedte in de buurt komt van het theoretisch maximum van het netwerk. Met iPerf kunnen zowel TCP als UDP en SCTP connecties getest worden.

4.1.1 Het starten van tests

Bij het testen van een TCP verbinding moet de venstergrootte ingesteld worden. De venstergrootte bepaalt de hoeveelheid bytes die in één keer ontvangen kunnen worden.

De ideale venstergrootte kan worden berekend met de formule:

$$V = B_{laagste} \times Rtt,$$

waarbij geldt: V is de venstergrootte, $B_{laagste}$ is de laagste bandbreedte in het netwerk en *Rtt* is de *round trip time*, dit is de tijd die het netwerk nodig heeft om een datapakket te versturen en vervolgens een ontvangstbevestiging te versturen (*Kurkose & Ross*, 2003). Als de opgevraagde venstergrootte niet ingesteld kan worden, bijvoorbeeld als het besturingssysteem een vaste venstergrootte heeft, dan zal iPerf hier een waarschuwing over geven en de venstergrootte aanpassen naar de venstergrootte die wel mogelijk is.
Bij UDP creëert iPerf een UDP stroom met een contstante transmissiesnelheid. De server detecteert het aantal verloren datagrammen en geeft dit weer in een percentage (uiteenzetting van verloren datagrammen tegenover verstuurde datagrammen). Ook zogenaamde out-of-order datagrammen (datagrammen die op een verkeerd moment binnenkomen) worden gedetecteerd. Het is met iPerf ook mogelijk om een multicast verbinding te starten. Bij een multicast verbinding is het mogelijk om twee servers te starten die naar dezelfde client luisteren. De client wordt in dat geval eerder dan servers gestart. Ook moet een IP-adres gekozen worden waar alle nodes naar luisteren/sturen.
Als men IPv6 wil gebruiken in plaats van IPv4 kan dit met de -V optie. Eerst moet het IPv6 adres van de server verkregen worden met behulp van het commando *ifconfig* (*ipconfig* onder Windows). Dit IPv6 adres moet zowel aan de server als aan de client meegegeven worden.
Het is ook mogelijk om de iPerf server als daemon te laten draaien op Unix/Linux. Om dit te starten moet de optie -D meegegeven worden. Omdat een daemon geen directe uitvoer genereert moet deze uitvoer naar een file weggeschreven worden. Het commando dat gegeven wordt, ziet er als volgt uit:

```
iPerf -s -D > uitvoerfile
```

iPerf heeft een mechanisme waarmee de client zijn venstergrootte kan aanpassen tot de meest efficiënte venstergrootte. Dit mechanisme kan worden geactiveerd door de -W optie te gebruiken. Om het netwerk zo efficiënt mogelijk te laten werken, moet de juiste venstergrootte ingesteld worden.
Om goed begrip te krijgen van de tool zijn een aantal tests uitgevoerd over een zelfgemaakt netwerk. De resultaten van deze tests zijn te vinden in bijlage D. De volgende materialen zijn gebruikt:

- Twee Intel NC5CPYH machines met Ubuntu 15.10 en een 1 Gb/s netwerkkaart;
- Een Juniper EX2200-C switch, maximale bandbreedte 1 Gb/s;
- Een 3Com 4210 26-port switch, maximale bandbreedte 1 Gb/s;
- Een MacBook Pro met Os X 10.10.

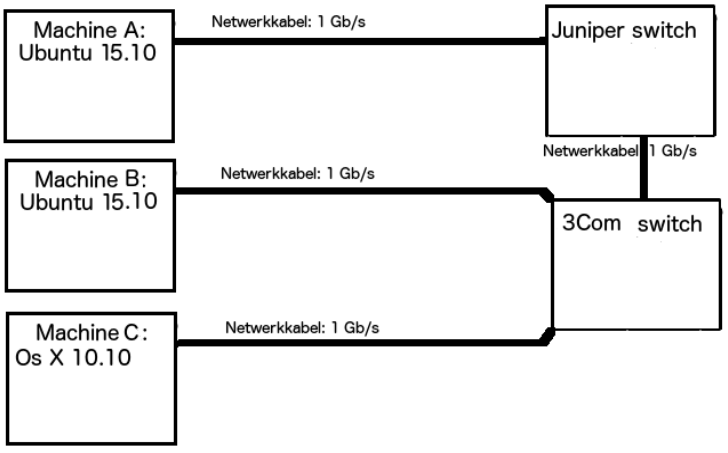
De testopstelling die gebruikt is voor het testen met beide switches is te zien in figuur 4.1. Deze manier was gekozen om te zien op welke manier iPerf omgaat met twee machines die data genereren en één machine die data ontvangt. Dit was interessant omdat de huidige situatie een soortgelijke opstelling vereist bij hogere datasnelheden. De testopstellingen met één switch zijn vergelijkbaar, het enige verschil is dat alle drie de testmachines dan op één switch aangesloten zijn en de andere switch uit het netwerk is gehaald. De commandos aan de server- en aan de cliëntzijde die tijdens de TCP tests werden gegeven waren:

```
Serverzijde: iperf -s
Cliëntzijde: iperf -c <ipadres> -o 2
```

De eerste twee seconden van de test werden genegeerd om de TCP slow start niet mee te nemen in de test. Bij de UDP tests was het commando aan de serverzijde hetzelfde. Het commando dat aan de cliëntzijde werd gegeven was:

```
iperf -c <ipadres> -u -b 1024
```

De bandbreedte die bij de UDP test ingegeven werd was 1 Gb/s omdat dit het theoretisch maximum van de testopstelling was.



Figuur 4.1: De testopstelling met beide switches

4.2 Onderzoek naar iPerf syntax

Om erachter te komen wat belangrijke functies in iPerf zijn, is een onderzoek uitgevoerd naar de syntax van iPerf. Door een aantal van deze functies terug te laten komen in de uiteindelijke testsuite, hoeven de gebruikers van de testsuite niet te wennen aan de

nieuwe tool. Op deze manier is de migratie van iPerf naar de nieuwe tool eenvoudiger. Belangrijke functies die iPerf biedt zijn de volgende:

- **TCP en UDP tests**
  Het is bij beide versies mogelijk om een netwerk te testen met het TCP of met het UDP (*iPerf developers*, 2016). Dit was ook één van de eisen aan de uiteindelijke testsuite. Deze mogelijkheid zal dus terugkomen in de uiteindelijke testsuite.

- **Eenheid van de uitvoer instellen (-f)**
  Deze optie dient om de eenheid van bijvoorbeeld Mb/s naar MB/s te veranderen (*iPerf developers*, 2016). Dit was een van de extra wensen van de opdrachtgever, als de tijd het toe laat zal ook deze functie toegevoegd worden aan de uiteindelijke testsuite.

- **TCP slow-start negeren (-O)**
  TCP heeft een slow-start fase. Tijdens deze fase wordt een klein deel van de beschikbare bandbreedte gebruikt. Om deze fase niet in de test mee te nemen, kan ervoor gekozen worden om deze fase te negeren, zodat alleen de maximale bandbreedte getest wordt (*iPerf developers*, 2016). Ook dit was een van de extra wensen van de opdrachtgever, als de tijd het toe laat zal ook deze functie toegevoegd worden aan de uiteindelijke testsuite.

- **Lengte van de test (-t)** Deze functie wordt gebruikt om de duur van een test in seconde in te stellen. Of dit op deze manier terugkomt is niet zeker, maar de duur van een test wordt op de een of andere manier verwerkt in de uiteindelijke testsuite.

Dit zijn functies die in iPerf beschikbaar waren en terug zullen komen, misschien op een andere manier vanwege de mogelijkheden van de library die gebruikt zal worden, in de uiteindelijke testsuite. Door deze functies terug te laten komen, zal het migreren van iPerf naar de nieuwe testsuite eenvoudiger zijn voor de eindgebruiker.

4.3 Samenvatting

In dit hoofdstuk werd het onderzoek naar de werking van iPerf beschreven. Om de werking te achterhalen werd gebruik gemaakt van de beschikbare literatuur over iPerf. Op deze manier werden de opties van iPerf ontdekt. Ook werd de vorm van de uitvoer van iPerf duidelijk. Dit hoofdstuk gaf antwoord op deelvraag 1: "Hoe werkt de huidige testool van iPerf?". Door deze deelvraag te beantwoorden werd duidelijk wat de bedoeling van de uiteindelijke testsuite is.

Hoofdstuk 5

Besturingssysteemrechten per library/tool

Dit hoofdstuk geeft antwoord op deelvraag 3. Naarmate het onderzoek vorderde bleek deze deelvraag steeds minder relevant, dit kwam doordat de eindgebruikers over het algemeen systeembeheerders zijn en dus zeker rootrechten hebben op de machines die voor de test worden gebruikt. Echter om de deelvraag toch te beantwoorden, wordt in dit hoofdstuk beschreven welke besturingssysteemrechten de eindgebruiker nodig heeft om de tools, of programma's gemaakt met de libraries te starten.

5.1 iPerf en Netcat

iPerf en Netcat hebben, om een test te starten, geen root rechten nodig. Dit komt doordat beide tools de datapakketten via systeemaanroepen genereren en aan de netwerkkaart doorspelen. Ook maken iPerf en Netcat geen gebruik van beschermde locaties.

5.2 DPDK

DPDK heeft voor het starten van zijn programma's altijd root rechten nodig. Dit komt doordat DPDK gebruik maakt van specifieke drivers die de netwerkkaart effciënt laat werken. Ook maakt DPDK gebruik van *hugepages*. Wanneer een proces in Linux geheugen gebruikt, wordt een deel van het RAM geheugen door de CPU gemarkeerd als bezet. Met het oog op efficiëntie, wordt ruimte in het RAM geheugen door de CPU geälloceerd in stukken van 4 KB. Die stukken heten pages. Soms heeft een proces meerdere pages nodig om te kunnen werken. Als een proces bijvoorbeeld 1 GB aan geheugen gebruikt, moeten 262144 ($\frac{1GB}{4KB}$) pages bekeken worden. Wanneer dit, of iets soortgelijks, het geval is, kan het efficiënter zijn om grotere pages te gebruiken. Deze grotere pages worden binnen Linux hugepages genoemd (*Debian*, 2015). Om deze hugepages aan te maken heeft DPDK dus rootrechten nodig.

5.3 Web10G

Bij Web10G is het standaard, dus bij het genereren van data, niet nodig om rootrechten te hebben. Echter wanneer delen van de library gebruikt worden die wel locaties aan willen passen die alleen door root gebruikers aangepast mogen worden, is het nodig dat de eindgebruiker rootrechten heeft.

5.4 Samenvatting

In dit hoofdstuk zijn de benodigde rechten van de verschillende libraries bekeken. Dit hoofdstuk gaf antwoord op deelvraag 3. Naarmate het onderzoek vorderde bleek deze deelvraag steeds minder relevant.

Hoofdstuk 6

De beste library/tool

In dit hoofdstuk worden de libraries/tools met de eisen geconfronteerd. Bij iedere library zal per eis bekeken worden in hoeverre ze aan die eis voldoen en hoeveel punten zij dus voor de eis zullen krijgen. De library/tool met de meeste punten wordt uiteindelijk gebruikt voor het onderzoek. Zoals in hoofdstuk 3 is uitgelegd zijn de libraries/tools die verder onderzocht gaan worden: iPerf, DPDK, Netcat en Web100/Web10G. Web100 is de voorganger van Web10G, omdat Web10G nog relatief nieuw is en dus nog niet veel over Web10G te vinden is wordt Web100 als referentie library gebruikt. De reden hiervoor is dat deze libraries/tools voldoen aan de knock-out eisen, beschreven in hoofdstuk 3. Deze eisen waren:

1. De library/tool moet ondersteund worden door Linux en Linux ondersteunen.
2. De broncode moet beschikbaar en open-source zijn.
3. De mogelijkheid tot het genereren van datapakketten moet ondersteund worden.

Ook is in hoofdstuk 3 beschreven hoe de verdeling van de punten bij de verschillende eisen is geregeld. Voordat de libraries/tools aan de eisen blootgesteld worden, wordt eerst een korte beschrijving van de library/tool gegeven. Onder het woord beste in de titel wordt verstaan: met de meeste kans op slagen voor dit project. Voor iedere eis kan maximaal vijf punten behaald worden. In totaal kunnen dus, na het vermenigvuldigen met de weegfactoren, eis één: 1; eis twee: 2; eis drie: 0,5, 17,5 punten behaald worden.

6.1 iPerf

iPerf is een testtool waarmee de bandbreedte van netwerken getest kunnen worden. Dit testen doet iPerf door een stroom aan datapakketten te genereren en te versturen.

6.1.1 Broncode moet duidelijk zijn

De broncode wordt door verschillende mensen ter wereld aangeleverd, omdat iPerf open-source is. De aangeleverde broncode wordt door de oorspronkelijke ontwikkelaars, en uitgevers, gescreend op de werking (*iPerf ontwikkelaars*, 2016). Dit betekent dat het zeker is dat alle aangeleverde code goed functioneert. Echter wordt de opmaak van de broncode niet gecontroleerd. Dit heeft als nadeel dat de broncode van iPerf niet is opgesteld als één geheel. Hiermee wordt bedoeld dat de opmaak van de functies en bestanden inconsistent is. Dit maakt het uitbreiden van de code lastiger, omdat het minder overzichtelijk is. Ook verschilt de duidelijkheid van de namen van functies. Bij sommige functies is het direct duidelijk wat de functie doet, terwijl andere functies eerst volledig geanalyseerd moeten worden voordat het doel van de functie duidelijk is. Dit analyseren is wel eenvoudig gemaakt, doordat alle functies van duidelijk commentaar voorzien zijn. De bestandsnamen zijn wel duidelijke weerspiegelingen van de doelen van de bestanden. iPerf is dus qua duidelijkheid gemiddeld omdat de broncode niet overzichtelijk is opgesteld, maar wel duidelijk is door het commentaar dat bij de functies is gezet. Daarom krijgt iPerf drie punten voor deze eis. Dit puntenaantal past bij iPerf omdat de duidelijkheid van de broncode gemiddeld is, niet uitermate slecht, maar ook niet uitzonderlijk goed.

6.1.2 De library/tool moet potentieel de 40-100 Gb/s kunnen halen

Over de potentiële maxima van iPerf is weinig te vinden. Echter is één artikel gevonden waarin gesproken wordt over dataverkeer testen over een netwerk van 40 Gb/s. Bij deze tests wordt gebruik gemaakt van iPerf, in dit artikel wordt niet duidelijk van hoeveel machines gebruik gemaakt werd tijdens deze tests (*Lee, Park, Jang, Moon & Han*, 2015). Aangezien de opdracht is dat de uiteindelijke testsuite vanaf één machine meer dan 40 Gb/s moet kunnen versturen is dit wel van belang om te weten. Omdat dus een aantal onderzoeken zijn gevonden betekent dit dat iPerf in de tweede categorie uit hoofdstuk 3 en dus krijg iPerf vier punten voor deze eis.

6.1.3 De library/tool moet overzichtelijke uitvoer genereren

De tool iPerf wordt wereldwijd gebruikt. De redenen hiervoor zijn de eenvoud met het installeren, het feit dat de Linux kernel geen aanpassingen/parameters nodig heeft en de overzichtelijke uitvoer. De uitvoer van een standaard iPerf test is te vinden in figuur 6.1. Zoals te zien is in de figuur, genereert iPerf een uitvoer in een tabelformaat. Ook staat boven iedere kolom duidelijk vermeldt wat in de kolom te vinden is. De gemeten bandbreedte is ook duidelijk weergegeven. Dit betekent dat iPerf voor deze eis vijf punten krijgt.

```
Connecting to host 10.60.66.172, port 5201
[  4] local 10.60.66.156 port 55263 connected to 10.60.66.172 port 5201
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-1.00   sec  93.5 MBytes   785 Mbits/sec
[  4]   1.00-2.00   sec   112 MBytes   936 Mbits/sec
[  4]   2.00-3.00   sec   112 MBytes   940 Mbits/sec
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec
[  4]   4.00-5.00   sec   112 MBytes   940 Mbits/sec
[  4]   5.00-6.00   sec   112 MBytes   940 Mbits/sec
[  4]   6.00-7.00   sec   112 MBytes   940 Mbits/sec
[  4]   7.00-8.00   sec   112 MBytes   940 Mbits/sec
[  4]   8.00-9.00   sec   112 MBytes   940 Mbits/sec
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-10.00  sec  1.08 GBytes   924 Mbits/sec                  sender
[  4]   0.00-10.00  sec  1.08 GBytes   924 Mbits/sec                  receiver
```

Figuur 6.1: Uitvoer iPerf test

6.1.4 Eindresultaat

In totaal heeft iPerf, na vermenigvuldiging met de weegfactoren, 13,5 punten gekregen. iPerf heeft vooral punten laten liggen bij de opmaak en naamgeving van de broncode. Waar iPerf beter op scoorde was de overzichtelijkheid van de uitvoer. Wat de potentie van iPerf betreft is niet zoveel gevonden, maar wel genoeg om in de tweede categorie van de eis terecht te komen.

6.2 DPDK

DPDK is een groep libraries die het mogelijk maken om verschillende soorten programma's te maken die een netwerk kunnen manipuleren. DPDK maakt het dus ook mogelijk om datastromen te kunnen versturen en ontvangen. Op deze manier kan een testsuite gemaakt worden die de bandbreedte van een netwerk test.

6.2.1 Broncode moet duidelijk zijn.

DPDK is open-source, maar voordat de nieuwe functionaliteit wordt toegevoegd, wordt de aangeleverde broncode eerst door een team van Intel gescreend. Dit gebeurt voordat de updates doorgevoerd worden en op de website komen te staan. Tijdens dit screenen wordt niet alleen gekeken naar de werking van de contributie maar ook naar de opmaak en naamgeving van de code. Daardoor wordt de code van DPDK één geheel. Verder heeft Intel veel documentatie vrijgegeven over het installeren van DPDK en over het ontwikkelen met DPDK. Ook is alle broncode voorzien van duidelijk commentaar. Vanwege al deze documentatie en vanwege het feit dat de broncode van consistente opmaak en naamgeving is voorzien, krijgt DPDK voor deze eis vijf punten.

6.2.2 De library/tool moet potentieel de 40-100 Gb/s kunnen halen

IBM Aspera Solutions heeft DPDK gebruikt om dataverkeer te versturen met snelheden tot 80 Gb/s (*Shiflett*, 2015). Zij deden dit met vijf Intel NVMe SSD's, twee Intel Xeon E5-2697 v3's en twee intel 40 Gb Ethernet kaarten (*Shiflett*, 2015). Ook heeft Chelsio Communications DPDK gebruikt om datapakketten te genereren met snelheden van 40 Gb/s (*Chelsio*, 2015). Dit zorgt ervoor dat de verwachting ontstaat dat met behulp van DPDK, het genereren van datasnelheden van meer dan 40 Gb/s ook kan lukken. Omdat DPDK dus al eens gebruikt is voor het genereren van dataverkeer met een snelheid van 40 Gb/s, valt DPDK in de derde categorie van deze eis, zoals die is beschreven in hoofdstuk 3. Dit betekent dat DPDK vijf punten krijgt voor deze eis.

6.2.3 De library/tool moet overzichtelijke uitvoer genereren

DPDK heeft geen standaard uitvoer. Dit betekent dat de uitvoer zelf ontworpen moet worden, dus de data bevat waar de ontwikkelaar zelf voor kiest. Op deze manier is de uitvoer gegarandeerd zo overzichtelijk als de ontwikkelaar zelf wilt. Wel is een tool gevonden die gebruik maakt van DPDK, genaamd Pktgen. Deze tool is gebruikt als referentie tool en zal ook uitgebreid worden als DPDK als beste uit het onderzoek komt. Omdat DPDK verschillende functies heeft gewijd aan het mogelijk maken van uitvoer, krijgt DPDK drie punten.

6.2.4 Eindresultaat

In totaal heeft DPDK, na vermenigvuldiging met de weegfactoren, 16,5 punten gekregen. DPDK laat vooral punten liggen bij het genereren van uitvoer. Dit komt doordat bij DPDK zelf een functie geschreven moet worden om uitvoer te genereren.

6.3 Netcat

Netcat is een testtool waarmee op verschillende manieren een netwerk getest kan worden. Zo kan Netcat onder andere de stabiliteit van de connectie testen en kan Netcat de bandbreedte van een netwerk testen.

6.3.1 Broncode moet duidelijk zijn

De broncode van Netcat is beschikbaar, maar de code is niet overzichtelijk. Dit komt onder andere door de namen van de functies, variabelen, enzovoorts. Deze bevatten namelijk regelmatig onzinwoorden en geven niet duidelijk weer waar de functies en variabelen voor dienen. Ook zijn de functies aan de relatief grote kant en gebeurt het regelmatig dat één functie meerdere doelen heeft. Verder is de code van Netcat allemaal bevat in slechts één bestand, dit bestand heeft daarom een enorme omvang, en verliest

daardoor aan overzichtelijkheid. De broncode is wel voorzien van commentaar en daarmee kan de broncode toch ontcijferd worden. Omdat de namen gebruikt in Netcat dus erg onduidelijk zijn, de functies erg lang zijn en alle code in één bestand staat, krijgt Netcat weinig punten voor deze eis. Dankzij het commentaar dat bevat is in het bestand is dit echter niet het minimaal aantal punten, omdat daardoor de code toch te begrijpen is. Om deze redenen krijgt Netcat twee punten voor deze eis.

6.3.2 De library/tool moet potentieel de 40-100 Gb/s kunnen halen

Over de combinatie van Netcat en 40-100 Gbs/s zijn geen specifieke artikelen of andere bronnen te vinden. Noch zijn er over dit onderwerp persoonlijke projecten op internet te vinden. Dit komt waarschijnlijk door de kleine bekendheid van Netcat. Zoals in hoofdstuk 3 is beschreven krijgt Netcat drie punten, vanwege het feit dat geen onderzoek niet betekent dat het onmogelijk is om deze snelheden te behalen. Op deze manier wordt Netcat niet direct weggeschreven vanwege de hoge weging van deze eis.

6.3.3 De library/tool moet overzichtelijke uitvoer genereren

De enige uitvoer die tijdens het testen gegenereerd werd was een bestand met een aantal ”y”-s onder elkaar gezet en de duur van de test in de terminal. Bij complexere tests wordt wel duidelijkere uitvoer gegenereerd. Deze uitvoer bevat dan bijvoorbeeld als informatie dan het aantal binnengekomen pakketten op bij de server, het aantal verzonden pakketten, het totaal aantal verstuurde bytes, de duur van het versturen en de gemeten bandbreedte. Daarom krijgt Netcat voor deze eis vier punten.

6.3.4 Eindresultaat

In totaal heeft Netcat, na vermenigvuldiging met de weegfactoren, 10 punten gekregen. Netcat krijgt vooral weinig punten voor de duidelijkheid van de broncode.

6.4 Web10G

Web100 en Web10G zijn twee libraries van dezelfde ontwikkelaars en uitgevers. Beide versies zijn ongeveer hetzelfde met als enige verschil dat met Web10G snelheden van 10 Gb/s behaald kunnen worden. Web100 en Web10G maken het mogelijk om datapakketten te genereren en te versturen.

6.4.1 Broncode moet duidelijk zijn

Web100/Web10G heeft vrij weinig code, daardoor is de code al snel overzichtelijk. De code is helaas niet goed gedocumenteerd, dit betekent dat het programmeren voornamelijk trial-and-error is. Daarom krijgt Web100/Web10G drie punten voor deze eis.

6.4.2 De library/tool moet potentieel de 40-100 Gb/s kunnen halen

Over de potentie van hogere snelheden met Web100 is niet veel te vinden. Dit komt doordat met Web10G al een stap is gemaakt in de richting van de hogere snelheden, daardoor wordt dus niet veel meer met Web100 bij hogere snelheden geëxperimenteerd. Omdat Web10G nog relatief nieuw is, zijn er nog geen artikelen of verslagen geschreven over Web10G die het testen van hogere datasnelheden beschrijven. Web10G valt dus in dezelfde categorie als Netcat en krijgt daarom drie punten voor deze eis.

6.4.3 De library/tool moet overzichtelijke uitvoer genereren

Web100/Web10G genereert alleen de data om te versturen en levert dus uit zichzelf geen uitvoer. Voor de uitvoer moet dus een ander programma ontwikkeld worden, of moet Web100/Web10G uitgebreid worden. Dit betekent dat, doordat de uitvoer zelf ontworpen wordt, het altijd overzichtelijk is voor de gebruiker. Wel biedt Web100/Web10G functies voor het maken van uitvoer. Daarom krijgt Web100/Web10G voor deze eis drie punten.

6.4.4 Eindresultaat

In totaal heeft Web100/Web10G, na vermenigvuldiging met de weegfactoren, 10,5 punten gekregen. Web100/Web10G laat vooral punten liggen door het gebrek aan documentatie.

6.5 Samenvatting

Uit de puntenaantallen, te zien in tabel 6.1, is gebleken dat DPDK het beste gebruikt kan worden voor het vervolg van dit onderzoek.

Tabel 6.1: Punten per eis gegeven aan de libraries/tools, na vermenigvuldiging met de weegfactoren

Library/tool	Broncode	Potentie hogere snelheden	Uitvoer	Totaal
iPerf	3 * 1	4 * 2	5 * 0,5	13,5
DPDK	5 * 1	5 * 2	3 * 0,5	16,5
Netcat	2 * 1	3 * 2	4 * 0,5	10
Web100/Web10G	3 * 1	3 * 2	3 * 0,5	10,5

Hoofdstuk 7

Ontwerp

Dit hoofdstuk beschrijft het ontwerp van de testsuite. In paragraaf 7.1 is beschreven op welke manier DPDK gebruikt gaat worden. Uit de eisen, beschreven in paragraaf 7.2, is een functioneel ontwerp ontstaan. Voor dit functioneel ontwerp is een use case diagram gemaakt, dit use case diagram is te vinden in figuur F.1. Ook is een work-flow diagram gemaakt. Dit diagram is te vinden in figuur F.2. Verder is een communicatiediagram gemaakt, dit diagram is te vinden in figuur F.3. Omdat de testsuite is geprogrammeerd in de taal C, is geen klassendiagram gemaakt.

7.1 DPDK

Uit het onderzoek naar de beste library/tool, beschreven in hoofdstuk 6, is gebleken dat voor dit project DPDK de beste kandidaat is. Daarom wordt voor de rest van het onderzoek naar de snelheden tussen 40 Gb/s en 100 Gb/s DPDK gebruikt. Om de focus op het behalen van de hoge snelheden te laten liggen, wordt als basis voor de testsuite Pktgen gebruikt. Pktgen is een programma, ontwikkeld met behulp van DPDK, waarmee datapakketten gegenereerd worden. Dit programma wordt dus gebruikt om de datapakketten te genereren die ervoor moeten zorgen dat de bandbreedte getest kan worden. Pktgen zal uitgebreid worden, zodat met Pktgen de juiste snelheden behaald kunnen worden en Pktgen de functionaliteiten biedt waar de opdrachtgever om gevraagd heeft.

7.2 Functionele eisen

De opdrachtgever heeft enkele eisen aan het eindproduct gesteld. Deze eisen zijn beschreven in tabel 2.1. De functionele eisen die de opdrachtgever aan de testsuite had zijn:

- Met de testsuite moeten minimaal snelheden van meer dan 40 Gb/s getest kunnen worden

- De testsuite moet UDP en TCP kunnen testen
- De testsuite moet werken onder bekende Linux distributies
- De testsuite moet TCP slow start kunnen negeren[1]
- De eenheid van de uitvoer aanpassen[1]

7.3 Functioneel ontwerp

Uit de eisen zijn verschillende use cases voortgekomen. Deze use cases zijn: Test starten; Test stoppen; Instellingen poort wijzigen; Testresultaten controleren; Test uitvoeren; Datapakketten genereren; Datapakketten versturen; Test uitvoeren beëindigen; Stoppen datapakketten genereren; Stoppen datapakketten versturen. In paragraaf 7.3.1 wordt kort beschreven wat de doelen van de use cases zijn. Een uitgebreide beschrijving van de use cases is te vinden in bijlage F.

7.3.1 Use case diagram

De use case *Test starten* heeft als actor Tester. Deze use case stelt de tester in staat om een test naar de bandbreedte van een netwerk te starten. Deze use case include de use case *Test uitvoeren*, omdat de testsuite een test moet uitvoeren zodra de tester een test start.
De use case *Test stoppen* heeft als actor Tester. Deze use case biedt de tester de mogelijkheid om een draaiende test te sluiten. Deze use case include de use case *Test uitvoeren* beëindigen, omdat de testsuite een test moet beëindigen wanneer de tester dat aangeeft.
De use case *Instellingen poort wijzigen* zorgt ervoor dat de tester de instellingen van netwerkpoorten kan veranderen, om bijvoorbeeld de test te finetunen. Deze use case heeft als actor Tester.
De use case *Testresultaten controleren* heeft als doel de tester in staat stellen de testresultaten te bekijken, zodat de tester de resultaten kan vergelijken met de verwachtingen die de tester had.
De use case *Test uitvoeren* heeft als doel om de testsuite een test uit te laten voeren. Wanneer een test uitgevoerd wordt, moet data gegenereerd worden. Daarom include deze use case de use case *Datapakketten genereren*.
Datapakketten genereren moet de datapakketten genereren om te versturen. Om deze datapakketten te versturen zodra ze gegenereerd zijn, include deze use case de use case *Datapakketten versturen*.
De use case *Test uitvoeren beëindigen* heeft als doel om de testsuite een test te beëindigen zodra de tester dit aangeeft. Wanneer een test beëindigd wordt, moet het genereren van de datapakketten ook stoppen. Om die reden include deze use case de use case *Stoppen datapakketten genereren*.

[1] Deze eisen zijn samengevoegd tot de eis: de testsuite moet per netwerkpoort instellingen kunnen wijzigen

De use case *Stoppen datapakketten genereren* moet ervoor zorgen dat de eindeloze stroom datapakketten gestopt wordt, door de testsuite te laten stoppen met het genereren van datapakketten. Omdat het stoppen van het genereren ervoor zorgt dat er niks meer te versturen valt, include deze use case de use case *Stoppen datapakketten versturen.*
De use case *Stoppen datapakketten versturen* zorgt ervoor dat, wanneer geen datapakketten meer gegenereerd worden, de testsuite kan stoppen met het versturen van data. Dit is nodig omdat de netwerkkaart anders blijft wachten op data die nooit zal komen.

7.4 Technisch ontwerp

7.4.1 Sequentie diagram

Het sequentie diagram begint bij de gebruiker. Deze dient de instellingen van de poorten te wijzigen naar de gewenste instellingen. Deze instellingen worden dan door de testsuite naar de versturende en ontvangende machine doorgestuurd. Vervolgens worden de nieuwe instellingen door de testsuite op het scherm weergegeven en wordt de verantwoordelijkheid teruggespeeld naar de gebruiker. Nu is het aan de gebruiker om een test te starten. Als de gebruiker dit doet, zal de testsuite een test uitvoeren en naar de verzendende machine het verzoek om data te genereren en te verzenden sturen. De testsuite stuurt dan ook naar de ontvangende machine een verzoek om de datapakketten te verwerken. Beide machines zullen dan de testsuite de snelheid van de data toesturen waarmee de testsuite de bandbreedte teruggeeft aan de gebruiker. Dit proces gaat continue door totdat de gebruiker het testen stopt. Wanneer dat gebeurt stuurt de testsuite het verzoek om de datastroom te stoppen naar het systeem, daarmee wordt het testen beëindigd. Vervolgens kan de gebruiker de testresultaten vergelijken met de verwachtingen die de gebruiker van het systeem had.

7.4.2 Communicatiediagram

In het communicatiediagram is beschreven hoe de communicatie tussen de verschillende onderdelen van het systeem werkt. De communicatie wordt gestart door de gebruiker. Deze geeft de instellingen voor de netwerkpoort in. Deze instellingen worden door de netwerkpoort doorgegeven aan de datagenerator. Vervolgens stuurt de gebruiker het start commando naar de gebruiksinterface. De testsuite geeft dan aan de datagenerator door dat deze moet beginnen met het genereren van data. De datagenerator stuurt zijn gegenereerde data vervolgens door naar de netwerkkaart (dit gaat met behulp van de juiste driver) en stuurt daarna de snelheid van het data genereren terug naar de testsuite. Vervolgens update de testsuite de huidige bandbreedte en stuurt dit, via het beeldscherm, naar de gebruiker. Ook wordt de data dan van de versturende netwerkkaart naar de ontvangende netwerkkaart verstuurd. Dit proces wordt herhaald totdat de gebruiker het stop commando stuurt. Wanneer de gebruiker het stop commando naar de

testsuite stuurt, omdat de test afgerond is, stuurt de gebruiksinterface een signaal naar de datagenerator, zodat deze stopt met het genereren van data.

7.5 Evaluatie ontwerp

Aan dit ontwerp hangen enkele voor- en nadelen, deze zijn beschreven in paragraaf 7.5.1. Ook horen bij dit ontwerp een aantal beperkingen, beschreven in paragraaf 7.5.2.

7.5.1 Voordelen en nadelen

Met dit ontwerp wordt aan alle functionele eisen voldaan, door de poortinstellingen te wijzigen kunnen namelijk keuzes gemaakt worden tussen UDP en TCP en kunnen dingen als TCP slow start negeren en de eenheid van de uitvoer ingesteld worden. Wat betreft het testen van een netwerk heeft dit ontwerp als grootste nadeel dat de tester de test zelf moet beëindigen. Dit kan opgelost worden door een instelling toe te voegen die de duur van de test bepaald. Bij meer tests achter elkaar is het echter handiger om de test te beëindigen op commando van de gebruiker, omdat de gebruiker anders een bepaalde tijd moet wachten tot de test voltooid is. De gebruiker kan dan ook kortere tests uitvoeren, maar deze tests zijn minder nauwkeurig dan een test van een onbepaalde tijd, omdat bij een test van onbepaalde tijd de gebruiker de test kan stoppen wanneer de gewenste resultaten bereikt zijn of duidelijk is dat de gewenste resultaten niet met de huidige opstelling behaald kunnen worden.

7.5.2 Beperkingen

Een beperking van dit ontwerp is dat de tests met twee machines uitgevoerd moeten worden. Omdat om de bandbreedte van een netwerk goed te kunnen testen een machine gebruikt moet worden voor het verzenden van de datapakketten en een tweede machine voor het ontvangen van de datapakketten. Wanneer slechts één machine gebruikt wordt voor het verzenden en ontvangen van de datapakketten, dan wordt de data via localhost verstuurd. De datapakketten worden dan dus niet door de netwerkkaart verwerkt. Daardoor wordt de hardware die gecontroleerd wordt niet gebruikt en heeft de test dus geen nut gehad. Deze beperking valt dus niet te omzeilen.

7.6 Drivers

Voor het vervolg van het onderzoek gaat dus DPDK gebruikt worden. Om DPDK te kunnen gebruiken zijn ander drivers nodig dan de standaard drivers. Deze drivers zijn anders dan de standaard kernel drivers, omdat met de DPDK drivers de communicatie met de netwerkkaarten efficiënter zou verlopen (*Intel*, 2015a). Om de netwerkkaarten te kunnen gebruiken met DPDK moest dus een, voor DPDK geschikte driver gebruikt worden. Voor de meeste netwerkkaarten die gebruikt zijn was dit de *uio_pci_generic* driver. Bij de netwerkkaart die voor snelheden van 100 Gb/s geschikt is, is een andere

driver vereist. De driver die voor deze netwerkkaart gebruikt dient te worden is de *MLNX_OFED* driver. Het lukte echter niet om deze driver te laten werken. Daarom moest een driver ontwikkeld worden. Dit bleek echter niet mogelijk binnen de tijd. Ook viel dit niet binnen het kader van dit onderzoek, omdat hiervoor veel nieuwe kennis van de netwerkkaart nodig was.

Hoofdstuk 8

Werking DPDK

In dit hoofdstuk worden de werking van DPDK (de library die als beste uit deelvraag vier kwam) en de opbouw van de code van DPDK beschreven. Op deze manier wordt de benodigde kennis verkregen om met DPDK een eigen testsuite te kunnen ontwikkelen. In bijlage E wordt kort gekeken naar een meegeleverd voorbeeld.

8.1 Werking

DPDK is een groep libraries waarmee netwerken beïnvloed kunnen worden. DPDK is ontworpen om te werken met alle processoren. De libraries kunnen gebruikt worden om bijvoorbeeld datapakketten te verzenden en ontvangen binnen 80 CPU cycles, datapakketten te genereren op hoge snelheden of kortste paden in een stack op te slaan (*Intel*, 2015c). Omdat de eerste twee functionaliteiten het belangrijkste zijn voor dit onderzoek, zullen deze functies verder uitgelegd worden.

8.1.1 Datapakketten verzenden en ontvangen

DPDK is ontwikkeld om zo snel mogelijk datapakketten te kunnen verzenden en ontvangen. De datapakketten worden over het algemeen binnen 80 CPU-cycles verstuurd of ontvangen (*Intel*, 2015c). Dit betekent dat binnen 80 *ticks* van de interne klok een datapakket verstuurd of ontvangen kan worden. Een CPU van 2.0 GHz, die dus 2.000.000.000 *ticks* per seconde haalt, heeft met DPDK dus slechts 40 nanoseconden nodig om één pakket te verzenden of te ontvangen.

8.1.2 Datapakketten genereren

Om de datapakketten te kunnen versturen, moeten ze eerst gegenereerd worden. Dit genereren kan met behulp van DPDK erg efficiënt gedaan worden. Met behulp van DPDK kunnen per core namelijk ongeveer 14 miljoen pakketten per seconde gegenereerd

worden. Dit is in theorie genoeg om een netwerkkaart van 10 Gb/s te verzadigen met pakketten van 64 bytes aan data (*Pktgen ontwikkelaars*, 2015).

8.2 Benodigdheden

In deze paragraaf wordt uitgelegd wat DPDK nodig heeft om te kunnen werken en waarvoor DPDK deze benodigdheden gebruikt.

CPU cores
Ieder DPDK programma verwacht een coremasker parameter. Een coremasker wordt gerepresenteerd door een hexadecimaal getal. De bits van dit masker wijzen naar een logische core. Deze logische cores worden genummerd door Linux. DPDK zal de cores aangegeven in het coremasker volledig gebruiken. Deze cores zullen dus niet meer beschikbaar zijn voor andere processen. De layout van de logische cores kan verschillen tussen platforms, daarom moet van te voren goed uitgezocht worden welke cores beschikbaar gesteld kunnen worden voor DPDK (*Intel*, 2015a).

Hugepages
In paragraaf 5.2 is verteld wat hugepages zijn. In het kort: hugepages zijn plekken in het geheugen die verdeeld zijn in grotere stukken dan standaard (*Debian*, 2015). Omdat bij dit onderzoek uit gegaan wordt van hoge snelheden, moet in het geheugen een flinke buffer opgesteld worden voor DPDK. Om dit geheugen snel te kunnen bereiken heeft DPDK dus hugepages nodig. Ruimte voor deze hugepages kan in Linux klaargezet worden met het commando:

```
echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages.
```

Door dit commando uit te voeren, wordt ruimte voor de hugepages klaargezet. De hugepages zullen in het geval van dit commando een grootte hebben van 1 GB (*Intel*, 2015a).

Netwerkpoorten
DPDK heeft toegang nodig tot de netwerkpoorten. De reden hiervoor is dat DPDK anders geen data kan versturen en ontvangen. Daarom verwacht DPDK een poortmasker bij het starten van een programma. Een poortmasker kan verschillende vormen hebben. Soms is het een hexadecimaal getal dat lijkt op het coremasker. In andere gevallen kan het een decimaal getal zijn. In deze gevallen moeten de netwerkpoorten ook een lijst met cores toegewezen krijgen. Dit gebeurt in de vorm *[core1 - coreN].poortnummer*.

8.3 De code

Op het moment dat deze scriptie geschreven werd, was DPDK 2.2.0 de laatste versie. Van deze versie wordt dus ook uitgegaan in deze scriptie.

De code van DPDK is zeer uitgebreid, daarom zullen in deze scriptie alleen de core components behandeld worden. De core components zijn: *Ring Manager*, *Memory Pool Manager*, *Network Packet Buffer Management* en *Timer Manager* (*Intel*, 2015b).

Memory Pool Manager
De taak van de Memory Pool Manager is het alloceren van zogenaamde *objectpools* in het geheugen. Een pool wordt geïdentificeerd door een naam en gebruikt een ring om objecten op te slaan. Ook zorgt de Memory Pool Manager voor een object cache per core en biedt het hulp bij het gelijk verspreiden van de objecten over de RAM kanalen (*Intel*, 2015b).

Ring Manager
De ring structuur zorgt voor een multiproducent, multiconsument FIFO API in een tabel. De voordelen van een ring manager in plaats van een gewone *queue* zijn: ze zijn eenvoudiger te implementeren, aangepast aan grote operaties en sneller. De Memory Pool Manager gebruikt een ring en kan gebruikt worden als een standaard communicatie mechanisme tussen cores (*Intel*, 2015b). De ring manager is te vinden in librte_ring.

Network Packet Buffer Management
Het Network Packet Buffer Management (NPBM) zorgt voor een methode om buffers aan te maken en te vernietigen. Deze methode kan door een DPDK applicatie gebruikt worden om zogenaamde message buffers te maken. Een buffer wordt tijdens het starten aangemaakt en opgeslagen in een *memory pool*. Het NPBM maakt gebruik van de Memory Pool Manager. Het NPBM levert een API om geheugenbuffers te alloceren/vrij te geven, om controle berichten buffers, welke gewone berichten buffers zijn, te manipuleren en om pakket buffers, welke netwerk pakketten overdragen, te manipuleren (*Intel*, 2015b).

8.4 Samenvatting

In dit hoofdstuk werd de werking van DPDK uitgelegd. In paragraaf 8.1 werden de functionaliteiten van DPDK uitgelegd. De twee functionaliteiten waar verder uitleg over is gegeven, Datapakketten verzenden en ontvangen en Datapakketten genereren, zijn de twee belangrijkste functies voor dit project. In paragraaf 8.2 werd uitgelegd wat een typisch DPDK programma nodig heeft om te kunnen werken. Als laatste werd in paragraaf 8.3 uitgelegd wat de core componenten van de DPDK code zijn.

Hoofdstuk 9

Onderzoek en resultaten

9.1 Uitvoering

Het onderzoek is uitgevoerd in twee verschillende fasen. In de eerste fase werd onderzoek gedaan naar de mogelijkheid tot het genereren van 10 Gb/s aan data. In de tweede fase werd onderzoek gedaan naar het genereren van 40 tot 100 Gb/s.

9.1.1 10 Gb/s

In de eerste fase werd een testsuite gebruikt die 10 Gb/s aan datapakketten moest genereren. Deze fase had als doel om te oriënteren op DPDK, daarom werd gekozen voor een snelheid die zeker te halen was met DPDK. Tijdens deze fase werden verschillende tests uitgevoerd om erachter te komen wat de juiste instellingen voor de testsuite waren. De tests hadden als doel om te bepalen met welke toekomstbestendige instellingen de testsuite datapakketten kon versturen met een snelheid van 10 Gb/s. De onderzoeksmethode die gebruik werd, werkte door iedere keer dat een test uitgevoerd was de behaalde bandbreedte te bekijken, wanneer dit geen 10 Gb/s was, werd uitgezocht wat de reden is voor de lagere snelheid en werd een oplossing voor het probleem gezocht. Dit onderzoek was dus een iteratief proces waaruit uiteindelijk de meest toekomstbestendige instellingen voor de testsuite naar voren zou komen.
De tests zijn uitgevoerd met behulp van het programma Pktgen, dit programma is ook gebruikt als basis voor de testsuite. Het programma Pktgen geeft de snelheid in Mb/s weer. Alle tests zijn unidirectioneel uitgevoerd. Dat wil zeggen, een machine was de zender en een tweede machine was de ontvanger. Voor de tests zijn twee machines gebruikt. Op deze machines zijn twee verschillende besturingssystemen geïnstalleerd, omdat de testsuite op meerdere verschillende Linux distributies moet kunnen werken en niet alle testmachines altijd dezelfde besturingssystemen kunnen draaien. De gekozen besturingssystemen zijn Ubuntu 14.04 en CentOS 7.2.1511. Het systeem dat Ubuntu draaide werkte op de Linux 4.2.0-27-generic kernel en het systeem dat CentOS draaide, werkte op de Linux 3.10.0-327.el7.x86_64 kernel. Deze machines zijn, zoals te zien is in

figuur 9.1 direct verbonden met een glasvezelkabel. Beide machines zijn voorzien van *Intel(R) Xeon(R) CPU E5335 @ 2.00GHz* processoren. In de beide machines zit een *Intel Corporation 82598EB 10-Gigabit AF* netwerkkaart. Dit betekent dat het theoretische maximum van het netwerk dus 10 Gb/s is.

Figuur 9.1: Testopstelling gebruikt voor het testen van 10 Gb/s

9.1.1.1 Finetunen van de testsuite

De eerste fase was het maken van een testsuite die datapakketten kan genereren met een snelheid van 10 Gb/s. Hiervoor werd als basis het programma *Pktgen* gebruikt. Dit programma is gemaakt met behulp van DPDK en door Intel op de website van DPDK geplaatst. In eerste instantie lukte het niet om de volledige theoretische bandbreedte te behalen. Dit kwam doordat de pakketten niet snel genoeg gegenereerd werden. Om dit probleem op te lossen werd de TX-burst verhoogd. TX zijn de pakketten die verstuurd worden, de tegenhanger hiervan is RX, wat de ontvangen pakketten zijn. De TX-burst kon echter niet verhoogd worden, omdat de waarde direct in de code was verwerkt. Dankzij het aanmaken van een variabele voor de TX-burst, kan de TX-burst nu eenvoudig verhoogd worden met behulp van de voorgenoemde variabele. Deze oplossing leek direct te werken. Het theoretisch maximum werd gehaald bij een TX-burst van 128. Echter door het geheugengebruik van het programma te analyseren, bleek dat een TX-burst van 128 al 20% van het geheugen nodig had. Om te controleren hoe toekomstbestendig deze oplossing was, werd de TX-burst verder verhoogd. Dit verhogen maakte alleen maar duidelijker dat deze oplossing ervoor zorgde dat het geheugengebruik evenredig is met de grootte van de TX-burst. Dit is te zien in tabel 9.1.

Tabel 9.1: Geheugengebruik bij verhoogde TX-burst

TX-burst (pakketten)	Geheugengebruik (%)
64	10
128	20
256	40
512	80

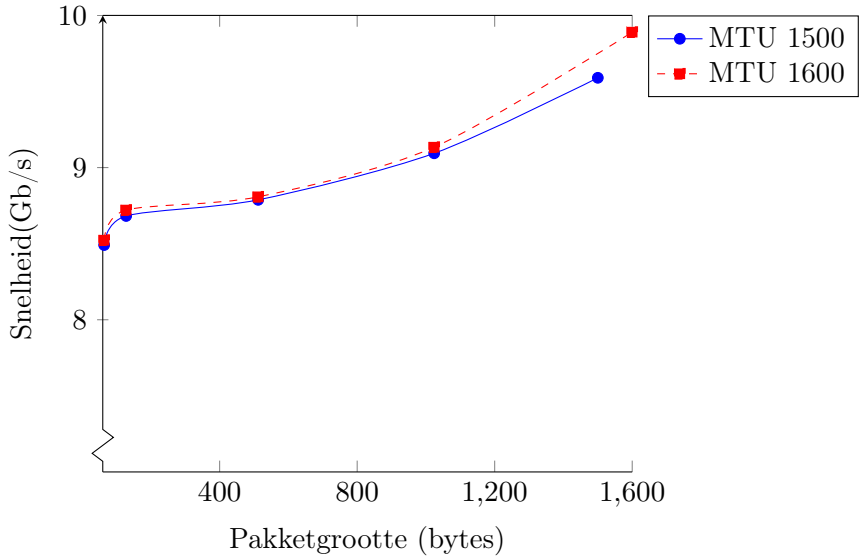
MTU vergroten

Het vergroten van de TX-burst leverde dus problemen op. Het genereren van meer

pakketten bleek niet te lukken. Daarom werd gekozen om grotere pakketten te versturen, met de filosofie: één groot pakket heeft hetzelfde effect als meerdere kleine pakketten. De standaardgrootte van een pakket is 64 bytes. Door dit stapsgewijs te vergroten, bleek dat de maximale grootte 1500 bytes was, en nog steeds het theoretisch maximum niet bereikt werd, zoals te zien in tabel 9.2a. De reden hiervan: de Maximum Transmission Unit (MTU) stond op 1500 ingesteld. Omdat DPDK niet de MTU van Linux overneemt maar de MTU direct weer terugzet naar 1500, de standaard MTU, moest de code van de pakketgenerator gemodificeerd worden. Echter om deze functie toe te voegen moest een Kernel Network-interface-card Interface (KNI) ontwikkeld worden. Een KNI is een interface die tussen de Linux kernel en de user-mode applicatie geïnstalleerd wordt en ervoor zorgt dat de user-mode applicatie, kernel-mode instructies uit mag voeren, zoals het verhogen van de MTU. DPDK levert een KNI mee, echter kan deze niet in combinatie met Pktgen worden uitgevoerd. Dit komt doordat beide programma's verwachten dat ze de enige programma's zijn die recht hebben op de driver en de netwerkkaart. Daarom moest een KNI worden geïmplementeerd in de code van Pktgen. Hiervoor werd de KNI meegeleverd door DPDK gebruikt als voorbeeld. Vanwege de, door de programma's verwachte, rechten kon de KNI van DPDK niet direct geïntegreerd worden. Daarom moest de code ook worden aangepast, zodat het wel kon samenwerken met Pktgen. De code van de KNI wordt uitgelegd in paragraaf 9.2. Toen de KNI geïnstalleerd was en de MTU verhoogd werd naar 9000, wat de maximale MTU is, bleek dat pakketten van 1500 bytes net niet groot genoeg zijn. Met pakketten van 1600 bytes groot, was de netwerksnelheid 10 Gb/s, en dus het theoretisch maximum, behaald. Een MTU van 1600 is dus voldoende om datapakketten te genereren die groot genoeg zijn om een netwerkkaart van 10 Gb/s te verzadigen. In figuur 9.2 staat de gemeten bandbreedte als functie van de gekozen grootte van de gegenereerde datapakketten. Bij deze test werd een TX-burst van 64 bytes gebruikt. In deze figuur is duidelijk zichtbaar dat, wanneer de pakketten op 1600 bytes ingesteld worden, de netwerkkaart zijn theoretisch maximum bereikte.
Het nadeel van deze oplossing is dat van de standaard MTU afgeweken moet worden.

Uiteindelijke oplossing

Voorgenoemde oplossing zal dus niet werken wanneer het niet mogelijk is van de standaard MTU af te wijken. Daarom is verder onderzoek gedaan naar een andere oplossing. De oplossing bleek *Non-uniform memory access* (NUMA) te zijn. NUMA zorgt ervoor dat de toegangstijd tot het geheugen afhangt van de locatie van het geheugen waar toegang tot gezocht wordt, relatief tot de processor (*SGI*, 2015). Dit zorgt ervoor dat het geheugen optimaal gebruikt wordt. Het voordeel van deze oplossing is dat de MTU op 1500 kan blijven en toch het theoretische maximum wordt bereikt. Wanneer NUMA niet ondersteund wordt, maar een hogere MTU wel, kan dus gebruik gemaakt worden van de KNI. Als beide niet mogelijk zijn dan kan de TX-burst verhoogd worden. Dit wordt echter sterk afgeraden vanwege het geheugengebruik.



Figuur 9.2: Gemeten snelheid dataverkeer 10 Gb netwerkkaart uiteengezet tot de pakketgrootte

Om de data duidelijker weer te geven is in tabel 9.2a en in tabel 9.2b de data van de grafiek opgenomen.

Tabel 9.2: Gemeten snelheid dataverkeer 10 Gb netwerkkaart bij verschillende MTU groottes

(a) MTU 1500

Pakketgrootte (B)	Snelheid (Gb/s)
64	8,491
128	8,682
512	8,788
1024	9,094
1500	9,590

(b) MTU 1600

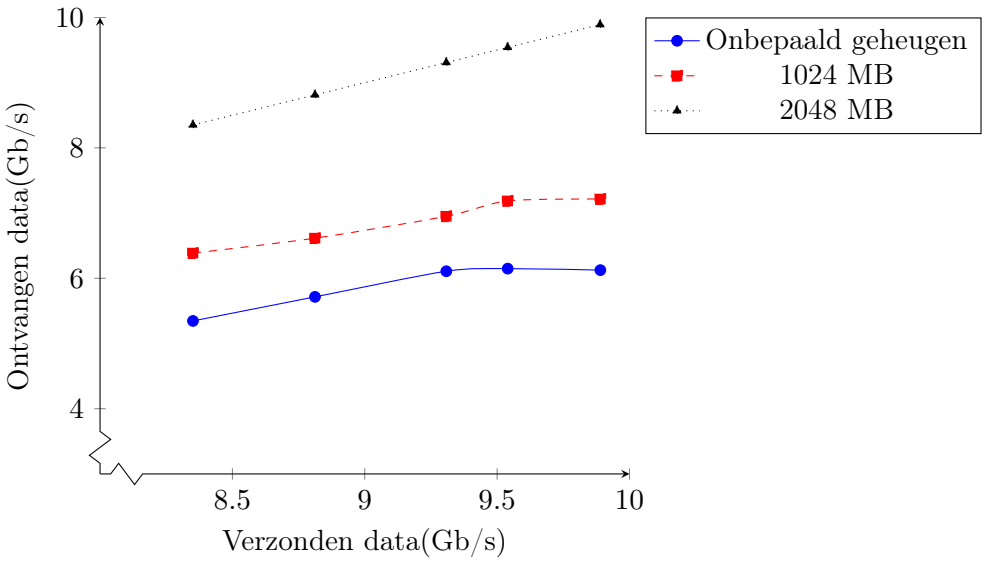
Pakketgrootte (B)	Snelheid (Gb/s)
64	8,523
128	8,721
512	8,808
1024	9,134
1600	9,890

De reden dat de snelheid niet dichter bij 10 Gb/s kon komen dan de 9,890 is het hardware limiet van de PCI-Express bus. Omdat de gebruikte PCI-E bus de nieuwste versie is, namelijk een PCI-E versie 3.0, kon de snelheid op geen enkele manier dichter bij 10 Gb/s komen. Bij de tests met de MTU en NUMA stond de TX-burst gewoon op 64 ingesteld. De parameters die bij het testen aan de testsuite werden meegegeven zijn beschreven in paragraaf 9.1.1.2.

Ontvangen van de data

Opvallend aan de tests was het ontvangen van de data. Het ontvangen van data bleek namelijk afhankelijk te zijn van het beschikbare geheugen. Wanneer geen hoeveelheid

geheugen ingesteld wordt, bepaalt het besturingssysteem hoeveel geheugen Pktgen toegewezen krijgt. Wanneer dit het geval was raakte ongeveer de helft van de data verloren. Zoals te zien is in figuur 9.3 en in kwam bij een hoeveelheid toegekend geheugen, bepaald door het besturingssysteem ongeveer de helft van de data niet aan en kwam de ontvangen data ook niet boven de 6100 Mb/s. Dit probleem ontstond waarschijnlijk door het feit dat de data sneller aankwam dan verwerkt kon worden. De data die aankwam terwijl het geheugen nog bezig was, werd daardoor genegeerd en kwam nooit aan. Bij een toegekend geheugen van 1024 MB werd al meer dan de helft van de data ontvangen. Echter kwam ook hier nog niet de ontvangen data tot de hoeveelheid verstuurde data en lag het maximum rond de 7200 Mb/s. Na het toegekende geheugen te verdubbelen naar 2048 MB, kwam alle data aan, zoals te zien is in figuur 9.3. Deze test werd uitgevoerd met NUMA. De MTU stond bij deze test op 1500 ingesteld, wat de standaardwaarde is. In tabel 9.3 is de data van figuur 9.3 duidelijker weergegeven.



Figuur 9.3: Ontvangen data uiteengezet tegen de verzonden data

Tabel 9.3: Ontvangen data bij verschillende hoeveelheden toegekend geheugen

Verzonden	Ontvangen	Verzonden	Ontvangen	Verzonden	Ontvangen
8,351	5,346	8,351	6,586	8,351	8,351
8,812	5,715	8,812	6,815	8,812	8,812
9,308	6,108	9,308	6,951	9,308	9,308
9,540	6,148	9,540	7,186	9,540	9,540
9,890	6,125	9,890	7,215	9,890	9,890

9.1.1.2 Resultaten

Het doel van deze tests was om de instellingen voor de testsuite te vinden die toekomstbestendig zijn en waarmee minimaal een netwerkkaart van 10 Gb/s verzadigd wordt.

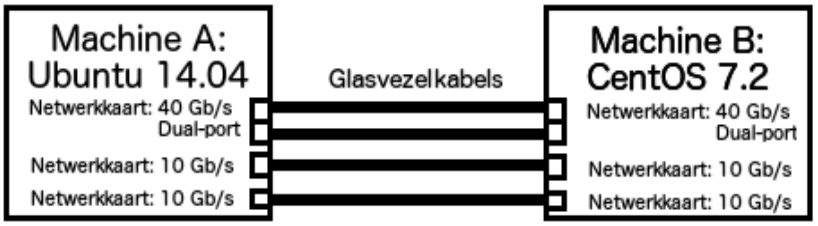
De beste instelling bleek het aanzetten van NUMA ondersteuning. Dit maakt het namelijk mogelijk om 10 Gb/s te genereren met datapakketten van 64 bytes groot. Het uiteindelijke commando dat werd gebruikt was de volgende:

```
./pktgen -c 0xff -n 4 -m 2048 -- -N -P -m "[2].0"
```

Met dit commando werd het praktische maximum van de testopstelling bereikt. De parameters voor het dubbele minteken zijn de parameters die ieder DPDK programma nodig heeft. Deze parameters zijn uitgelegd in hoofdstuk 8 in de paragraaf over de benodigdheden van DPDK. De parameters na het dubbele minteken zijn parameters voor de testsuite. Deze parameters betekenen het volgende: -N staat voor NUMA modus, deze functie zorgt ervoor dat NUMA gebruikt wordt voor het verwerken van de datapakketten; -P zet promiscuous mode aan, dit betekent dat alle binnenkomende datapakketten verwerkt worden door CPU, ook de datapakketten die niet door de CPU verwerkt hoeven te worden; -m regelt het "mappen"van de cores naar de poorten.

9.1.2 40 Gb/s

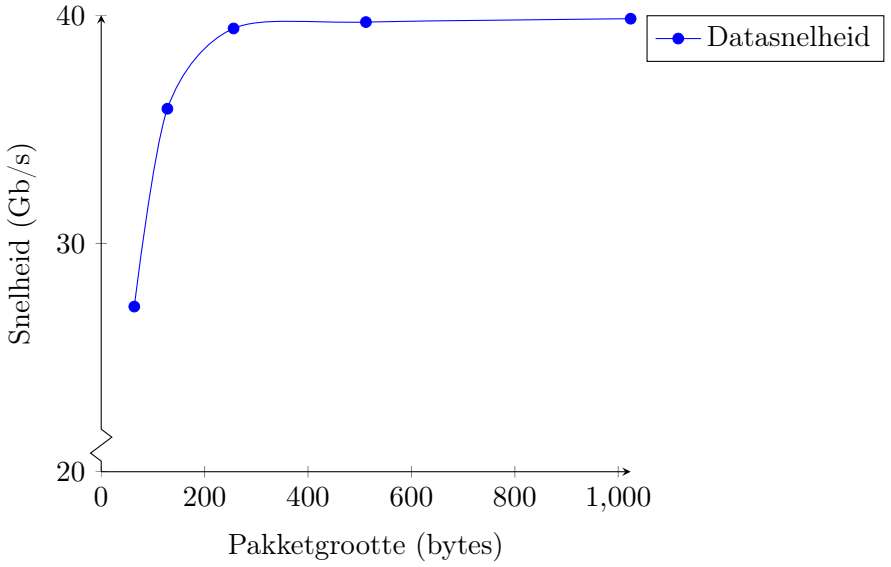
Nadat de tests met 10 Gb/s afgerond waren en toekomstbestendige instellingen gevonden waren, werd het onderzoek naar 40 Gb/s gestart. Het doel van deze tests was uitzoeken wat de testsuite nodig heeft om 40 Gb/s te kunnen bereiken. Door uiteindelijk deze testsuites te combineren zou in theorie minimaal 50 Gb/s behaald worden. Dit komt doordat de eerste testsuite 10 Gb/s kan genereren en de tweede testsuite 40 Gb/s. Ook bij dit onderzoek werd een iteratieve manier gebruikt, iedere keer dat een aanpassing gedaan werd, werd gekeken wat de nieuwe behaalde snelheid was. Voor het onderzoek zijn twee machines beschikbaar gesteld. Op één van de machines is Ubuntu geïnstalleerd. De versie van Ubuntu is 14.04.4. Op de andere machine is CentOS geïnstalleerd. De versie van CentOS is versie 7.2. De machines zijn met elkaar verbonden door middel van een glasvezelkabel op snelheid 40 Gb/s. De processoren die de machines bevatten zijn: *Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz* in de Ubuntu machine en *Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz* in de CentOS machine. Beide machines bevatten een dual-port *Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+* netwerkkaarten en twee *Intel Corporation Ethernet Controller X710 for 10GbE SFP+* netwerkkaarten. In figuur 9.4 is de testopstelling schematisch weergegeven. Het protocol dat werd gebruikt tijdens de tests was TCP, omdat om het aankomen van de datapakketten van belang was om te controleren of de bandbreedte juist gecontroleerd kon worden.



Figuur 9.4: Testopstelling gebruikt voor het testen van 40+ Gb/s

9.1.2.1 Resultaten

Als basis voor het onderzoek werd de uitgebreide versie van *Pktgen* gebruikt. Om één netwerkkaart van 40 Gb/s vol te kunnen krijgen, bleek de code niet veel aangepast te hoeven worden. Het bleek namelijk dat door de grootte van de pakketten aan te passen de netwerkkaart al snel verzadigd raakte. In figuur 9.5 en tabel 9.4 is te zien dat bij een pakketgrootte van 256 bytes een snelheid van iets minder dan 40 Gb/s gehaald wordt. Dit betekent dat de netwerkkaart verzadigd is, aangezien rekening moet worden gehouden met data-overhead.



Figuur 9.5: Gemeten snelheid dataverkeer 40 Gb netwerkkaart uiteengezet tot de pakketgrootte

Tabel 9.4: Gemeten snelheid dataverkeer 40 Gb netwerkkaart

Pakketgrootte (bytes)	Gemeten snelheid (Gb/s)
64	27,235
128	35,902
256	39,419
512	39,699
1024	39,846

Uit deze metingen is gekomen dat de minimaal benodigde pakketgrootte, om 40 Gb/s te bereiken, 256 bytes is. Het versturen van datapakketten met een snelheid van 40 Gb/s kan niet met kleinere pakketten door de hardware limitaties van de PCI express bus (*Chelsio*, 2015).

9.1.3 40+ Gb/s

Nadat het onderzoek naar 40 Gb/s was afgerond en de juiste instellingen voor 40 Gb/s waren gevonden, werd het onderzoek gestart naar de snelheden die boven de 40 Gb/s liggen. Het doel van deze tests was bewijzen dat met DPDK datasnelheden van meer dan 40 Gb/s te behalen is. Omdat de machine meerdere netwerkkaarten bevatte, kon in theorie met deze testopstelling ook meer dan 40 Gb/s behaald worden. Dit lukt in eerste instantie maar amper. Een netwerkkaart van 40 Gb/s en een netwerkkaart van 10 Gb/s genereerde samen slechts 42 Gb/s aan data. Dit terwijl het theoretische maximum 50 Gb/s was, omdat een netwerkkaart van 40 Gb/s en een netwerkkaart van 10 Gb/s werd gebruikt. De reden van de teleurstellende snelheid, was dat het programma een aantal systeemaanroepen moest uitvoeren. Systeemaanroepen zijn verzoeken van een computerprogramma aan het besturingssysteem om een taak uit te voeren. Deze verzoeken kosten relatief veel tijd. Door de KNI weer te gebruiken, konden deze systeemaanroepen omzeild worden. Daarvoor moest de KNI wel uitgebreid worden, zodat deze systeemaanroepen overgeslagen konden worden en het programma het zelf direct uit kon voeren. De systeemaanroepen werden gebruikt voor het versturen van pakketten naar de netwerkkaart. Dit moest dus vanuit de KNI gebeuren, dit wordt verder uitgelegd in paragraaf 9.2. Door de systeemaanroepen niet meer te hoeven uitvoeren, kan het programma Pktgen wel ongeveer 50 Gb/s genereren. Net zoals bij 40 Gb/s is daarvoor een pakketgrootte van 256 bytes nodig en is het iets minder dan 50 Gb/s doordat rekening gehouden moet worden met data-overhead. In tabel 9.5 is de data weergegeven. Bij het testen van een TCP verbinding moest rekening gehouden worden met slow start. Om deze slow start te negeren is gebruik gemaakt van de functie die beschreven is in paragraaf 9.2.1.1.

Tabel 9.5: Gemeten snelheid dataverkeer met één 40 Gb/s netwerkkaart en één 10 Gb/s netwerkkaart

Gebruikte netwerkkaarten	Gemeten snelheid (Gb/s)
10	9,517
40	39,397
40 +10	48,914

Nadat deze resultaten bereikt waren, werd de volledige potentie van de machine ingezet. Dit betekent dat de netwerkkaart van 40 Gb/s over beide poorten ging verzenden en beide netwerkkaarten van 10 Gb/s gebruikt werden. Door deze netwerkkaarten te combineren haalde de testsuite 70 Gb/s. Dit kwam doordat de netwerkkaart van 40 Gb/s over beide poorten kon verzenden. Wanneer een netwerkkaart van 40 Gb/s over beide poorten kan versturen, kunnen deze netwerkkaarten namelijk 50 Gb/s halen. Per poort wordt dan ongeveer 25 Gb/s verstuurt. Gecombineerd met de netwerkkaarten van 10 Gb/s, levert dat dus een snelheid van 70 Gb/s op. Deze data is weergegeven in tabel 9.6.

Tabel 9.6: Gemeten snelheid dataverkeer met beide poorten van de 40 Gb/s en twee 10 Gb/s netwerkkaarten

Gebruikte netwerkkaarten	Gemeten snelheid (Gb/s)
10	9,517
40	39,397
40 +10	48,914
40 +10 +10	58,430
dualport 40 +10 +10	68,933

De reden dat de data niet dichter bij 70 Gb/s kon komen dan 68,9 Gb/s is opnieuw de hardware limitaties van de PCI-Express bus.

9.2 Toegevoegde functionaliteit

In deze paragraaf worden de technische aspecten van de aan *Pktgen* toegevoegde functionaliteiten beschreven.

9.2.1 Overige toegevoegde functionaliteiten

In deze paragraaf wordt kort uitgelegd welke functionaliteiten ook zijn toegevoegd om de testsuite beter te maken.

9.2.1.1 Negeren van slow start

Een eis van de opdrachtgever was dat het mogelijk moest zijn om TCP slow start te negeren. Slow start is een algoritme waar TCP gebruik van maakt om de ontvanger

van de data de kans te geven om de data te verwerken. Slow start is geïmplementeerd om ervoor te zorgen dat de data rij niet volstroomt, hierdoor ervaart de gebruiker, van bijvoorbeeld een browser, dus geen verbreking van de connectie. In het kader van de maximale bandbreedte testen is het echter een hinder. Dit komt doordat bij het sturen van een constante stroom data van een constante grootte, slow start alleen aan het begin optreedt en ervoor zorgt dat de meting niet de juiste bandbreedte weergeeft. Daarom is de optie toegevoegd om de slow start van het TCP protocol te negeren en dus niet mee te nemen in de test. Wanneer deze optie gekozen is, wordt iedere keer bij de start van een datastroom, een tijdklok gestart. Zodra deze tijdklok is afgelopen, zal de meting beginnen en ziet de gebruiker de uitvoer verschijnen. De duur van deze tijdklok kan door de gebruiker worden ingesteld. Deze optie kan in de testomgeving aangezet worden met de code:

```
set <portnr> slowstart <tijd in ms>
```

9.2.1.2 Eenheid van de uitvoer aanpassen

In iPerf bestond de mogelijkheid om de eenheid van de uitvoer aan te passen. Dit is ook aan de testsuite toegevoegd, zodat de gebruiker kan bepalen of de uitvoer in Mb/s, in Gb/s, in MB/s of in GB/s weergegeven wordt. Op deze manier hoeft de gebruiker niet te rekenen om de gewenste eenheid te bepalen. De gewenste eenheid kan in de testomgeving worden ingesteld met behulp van de code:

```
set <portnr> unit [<1,2,3 of 4>]
```

9.3 Samenvatting

In dit hoofdstuk is de uitvoering van het onderzoek beschreven. Een deel van de problemen werd opgelost met behulp van een KNI. Een KNI zorgt ervoor dat een user-mode programma kernel instructies uit kan voeren. Daardoor kunnen systeemaanroepen omzeild worden en kan de code dus sneller werken. De conclusie van het onderzoek is dat DPDK bruikbaar is om datasnelheden boven de 40 Gb/s te genereren en daarmee te meten. Het is namelijk gelukt om datasnelheden van 70 Gb/s te halen, zoals beschreven in paragraaf 9.1.3. Dit was de maximaal mogelijke snelheid met de testopstelling beschreven in figuur 9.4. De rest van de conclusie is te vinden in hoofdstuk 11.

Hoofdstuk 10

Projectorganisatie

In dit hoofdstuk wordt de projectorganisatie besproken. In de eerste paragraaf wordt besproken hoe de communicatie met de betrokkenen is verlopen. In paragraaf 10.2 wordt besproken aan welke eisen is voldaan en aan welke eisen voldaan had moeten worden. Ten slotte wordt in paragraaf 10.3 besproken op welke momenten van de planning is afgeweken en wat de reden daarvan was.

10.1 Communicatie

10.1.1 Bedrijfsbegeleiders

De communicatie met de bedrijfsbegeleiders is gelopen via meerdere wegen. De bedrijfsbegeleiders waren over het algemeen op hun kantoor te vinden. Wanneer een vraag gesteld moest worden, kon dit dus vrijwel direct. Wanneer geen van beide bedrijfsbegeleiders te vinden waren op hun kantoren, vanwege verplichtingen elders, dan kon het contact altijd met behulp van e-mail tot stand komen. Voor deze manieren van communicatie was gekozen, omdat op deze manier altijd contact opgenomen kon worden. Wanneer een mijlpaal bereikt was, werd dit direct bij de bedrijfsbegeleider gemeld, zodat het onderzoek zo snel mogelijk vervolgd kon worden. Bij een mijlpaal moest namelijk een nieuwe machine voor het testen aangewezen en geconfigureerd worden. Wanneer problemen voordeden konden de begeleiders ook benaderd worden. De problemen werden dan gezamenlijk bekeken en de volgende stap werd gezamenlijk bedacht. Het initiatief van de gesprekken lag bij de afstudeerder.

10.1.2 Docentbegeleider

De communicatie met de docentbegeleider is vooral via e-mail gelopen. De docentbegeleider is ook eenmaal bij het bedrijf langsgekomen en verder is via Skype contact opgenomen. Voor deze manieren van communicatie was gekozen, omdat op deze manier

altijd contact opgenomen kon worden. Het contact met de docentbegeleider ging voornamelijk over de scriptie en het plan van aanpak. Het doel van de gesprekken was om feedback te krijgen op de verschillende documenten.

10.2 Eisen

Van de eisen die in hoofdstuk 2 opgesteld zijn, is aan alle must-haves voldaan. Met de testsuite kunnen namelijk snelheden van meer dan 40 Gb/s bereikt worden. De testsuite heeft enkele functionaliteiten, zoals de instellingen van verschillende poorten wijzigen en de eenheid van de uitvoer die aanpasbaar is, waardoor hij eenvoudig in het gebruik is. Ook werkt hij onder meerdere Linux distributies en is het mogelijk om UDP en TCP te testen. Met de bedrijfsbegeleider is overlegd over de eisen beschreven in tabel 2.1. Ook heeft de opdrachtgever een demonstratie van het eindproduct gekregen, zodat hij kon zien dat aan deze eisen is voldaan.

10.3 Planning

In deze paragraaf wordt teruggeblikt op de planning. Ook wordt in deze paragraaf verantwoord om welke redenen van de planning is afgeweken. Voordat de eerste sprint van start ging, is een vooronderzoek uitgevoerd. Dit vooronderzoek ging over de werking van iPerf, de beste library/tool en de werking van de beste library/tool. Deze planning is opgesteld in paragraaf 5 van het plan van aanpak. Dit plan van aanpak is te vinden in bijlage A. In deze paragraaf wordt besproken op welke momenten van de planning is afgeweken en wat de reden daarvan was.

Eerste sprint, 18-3-2016 tot 31-3-2016
In de eerste sprint werd DPDK klaargemaakt om mee te werken en werd een zogenaamd "Hello world"programma gebruikt om te testen of DPDK werkte. Ook werd tijdens deze sprint gewerkt aan het eerste concept van het scriptie.

Tweede sprint, 31-3-2016 tot 15-4-2016
Tijdens de tweede sprint werd gewerkt aan een testsuite die 1 Gb/s kon testen. Deze sprint werd echter niet afgerond, omdat de netwerkkaart die gebruikt werd tijdens het testen niet ondersteunt werd door DPDK. Dit werd echter pas aan het einde van de sprint ontdekt en dus werd besloten om deze (minder belangrijke) sprint over te slaan en direct naar 10 Gb/s over te stappen.

Derde sprint, 15-4-2016 tot 29-4-2016
Tijdens de derde sprint werd een testsuite ontwikkeld die 10 Gb/s kon testen. Tijdens deze sprint is verder niets bijzonders gebeurd en van de planning is dus niet afgeweken.

Vierde sprint, 29-4-2016 tot 13-5-2016
De vierde sprint was het ontwikkelen van een testsuite die 40 Gb/s kon testen. Ook tijdens deze sprint was geen reden om van de planning af te wijken.

Vijfde sprint, 13-5-2016 tot 27-5-2016
De laatste sprint omvatte het behalen van datasnelheden die hoger waren dan 40 Gb/s.

Hoofdstuk 11

Conclusie

In de scriptie is het onderzoek naar datasnelheden tussen 40 Gb/s en 100 Gb/s beschreven. Eerst is onderzoek gedaan naar een library/tool die het meest geschikt is om een testsuite te maken die deze snelheden aankan. De libraries/tools die onderzocht zijn waren: iPerf, DPDK, Netcat en Web10G. Uit dit onderzoek, beschreven in hoofdstuk 5, bleek dat DPDK de meest geschikte kandidaat was. DPDK werd dus gebruikt om de rest van het onderzoek uit te voeren. Met behulp van Pktgen, een programma dat datapakketten genereert, is vervolgens onderzocht op welke manieren datasnelheden van meer dan 40 Gb/s bereikt kunnen worden. Eerst werd een testsuite ontwikkeld die datasnelheden van 10 Gb/s moest bereiken. Voor deze snelheden bleken drie oplossingen te bestaan: het vergroten van de TX-burst, het vergroten van de MTU zodat grotere pakketten verstuurd kunnen worden of het gebruiken van NUMA. Van deze drie oplossingen was de oplossing met NUMA het beste. Het probleem met het verhogen van de TX-burst was namelijk dat dit het geheugengebruik van het programma exponentieel liet toenemen. De volgende oplossing die bedacht werd, was het vergroten van de datapakketten door de MTU te verhogen. Het probleem hiermee was, dat, door het verhogen van de MTU, van de standaard afgeweken moest worden. Dit betekent dat de kans bestaat dat dit niet altijd ondersteund wordt. In hoofdstuk 2 werd de verwachting uitgesproken dat snelheden hoger dan 40 Gb/s haalbaar zouden zijn. Deze verwachting is bevestigt door de resultaten die behaald zijn in paragraaf 9.1.2. Deze resultaten lieten namelijk zien dat, door de systeemaanroepen te omzeilen met behulp van een KNI, minimaal 70 Gb/s behaald kan worden. Dit betekent dat met DPDK een testsuite ontwikkeld kan worden die datasnelheden van meer dan 40 Gb/s kan bereiken.

Hoofdstuk 12

Aanbevelingen

Het is nog onduidelijk of de testsuite datasnelheden van 100 Gb/s kan bereiken. Dit kwam doordat het niet lukte om de driver van de netwerkkaart die beschikbaar gesteld was voor het testen van deze snelheden in combinatie met DPDK te gebruiken. Daarom wordt aanbevolen om contact op te nemen met de fabrikant van deze netwerkkaart met de vraag waarom dit niet mogelijk was, of om hulp te vragen bij het combineren van DPDK en de driver. Als de fabrikant geen oplossing heeft, is het ook mogelijk om zelf een driver te programmeren. Ook kan gekeken worden naar een andere netwerkkaart die deze snelheden kan bereiken en een DPDK driver kan gebruiken. Ook is het mogelijk om de testsuite te gebruiken met twee netwerkkaarten van 40 Gb/s en twee netwerkkaarten van 10 Gb/s. Op deze manier kan getest worden of de testsuite datasnelheden van 100 Gb/s kan bereiken.

Ook moet, om een beeld te krijgen van de bandbreedte, het testprogramma op twee computers gestart worden. De reden hiervan is dat het niet is gelukt om de ontvanger van de datapakketten feedback te laten sturen naar de zender van de datapakketten. Dit kwam door tijdgebrek en andere functionaliteiten die prioriteit over deze functionaliteit hadden. In hoofdstuk 2 is in de MoSCoW tabel de prioriteit te vinden. Als Nikhef deze functionaliteit toch wil, zal deze later toegevoegd moeten worden.

Hoofdstuk 13

Evaluatie

De meeste problemen die tijdens het onderzoek ontstonden, ontstonden door het laten werken van de drivers die nodig zijn om de netwerkkaarten te laten werken met DPDK. Het probleem met de drivers was namelijk dat ze moesten werken in combinatie met DPDK. Omdat dit niet altijd even soepel verliep, kon minder tijd in het ontwikkelen van de testsuite gestopt worden. Het is gelukkig wel gelukt om meer dan 40 Gb/s te kunnen genereren en dus te testen.

De sprints verliepen allemaal netjes op tijd. De sprints waren van gelijke lengte, maar omdat de eerste sprints minder moeilijk waren dan de latere sprints, was het beter geweest om niet de sprints van gelijke lengte te maken. Achteraf gezien waren de eerdere sprints namelijk minder belangrijk dan de latere sprints. Ook had bij sprint twee de conclusie voor het overslaan van die sprint eerder genomen moeten worden. Dit was namelijk een sprint die alleen nodig was voor kennis over de code van DPDK, iets wat ook opgedaan had kunnen worden door direct verder te gaan met sprint drie.

Tijdens het afstuderen is veel nieuwe kennis opgedaan. Deze kennis bestaat vooral uit geavanceerde Linux kennis en kennis over netwerken, waar in de toekomst nog veel mee gewerkt kan worden.

Het schrijven van de scriptie kostte veel meer tijd dan in eerste instantie gedacht werd. Dit kwam vooral doordat tijdens het schrijven van de scriptie het overzicht verloren raakte en de technische details de overhand kregen in de scriptie.

Bibliografie

Bell D., (Februari, 2004)
UML basics: The sequence diagram,
Opgehaald van: http://www.ibm.com/developerworks/rational/library/3101.html

Chelsio Communications, (2015).
Linux 40 GbE DPDK Performance,
Opgehaald van http://www.chelsio.com/wpcontent/uploads/resources/T540GbLinux-DPDK.pdf

Debian, (2015).
Hugepages,
Opgehaald van https://wiki.debian.org/Hugepages

Green C. , (Februari, 2016).
Migration to 100 Gb fibre: What you need to know to future proof your network,
Opgehaald van http://www.information-age.com/technology/mobile-and-networking/123460865/migration-100gb-fibre-what-you-need-know-future-proof-your-networks

Intel Corporation, (2015a).
Getting started guide for Linux,
Opgehaald van http://dpdk.org/doc/guides/linux_gsg/index.html

Intel Corporation, (2015b).
DPDK programmer's guide,
Opgehaald van http://dpdk.org/doc/guides/prog_guide/index.html

Intel Corporation, (2015c).
DPDK,
Opgehaald van http://dpdk.org/

iPerf ontwikkelaars, (2016).
iPerf, The TCP, UDP and SCTP network bandwith measurement tool,
Opgehaald van https://iperf.fr

Kurkose F. & Ross K., (2003).
Computer Networking A Top-Down Approach Pearson.

Lee C., Park C., Jang K. Moon S. & Han D., (2015).
Accurate latency-based congestion feedback for datacenters. In 2015 USENIX Annual Technical Conference (USENIX ATC 1).

Markus, (Augustus, 2016).
NetIO-GUI | SourceForge,
Opgehaald van https://sourceforge.net/projects/netiogui/

Moergestel L. van, (2012).
Computersystemen en embedded systemen, derde druk. Sdu Uitgevers bv, Den Haag

Netcat ontwikkelaars, (2011).
Netcat: the TCP/IP swiss army,
Opgehaald van: http://nc110.sourceforge.net

Miklos, (Februari, 2004).
NetIO-GUI | SourceForge,
Opgehaald van http://www.majorgeeks.com/files/details/aida32_personal_system_information.htm

Nikhef, (2016).
Nikhef website,
Opgehaald van https://www.nikhef.nl

Nishadha, (Februari, 2012).
The Complete Guide to UML Diagram Types with Examples,
Opgehaald van: http://creately.com/blog/diagrams/uml-diagram-types-examples/

Nuts about Nets, (2016)
NetStress, network benchmarking tool,
Opgehaald van: http://nutsaboutnets.com/netstress/

Osch, P. van, (Januari, 2014).
Business prioriteiten stellen met de MoSCoW methode
Opgehaald van http://sparkeducation.nl/prioriteiten-met-moscow/

Overheem B., (2004).
Wat is SCRUM?,
Opgehaald van http://www.scrum.nl/site/Wat-is-Scrum-agile-scrum.

Pktgen ontwikkelaars, (2015).
Pktgen Read the docs,
Opgehaald van: http://pktgen.readthedocs.io/en/latest/index.html

SGI, (2015).
NUMA (Numa support in Linux,
Opgehaald van: http://oss.sgi.com/projects/numa/

Shiflett, C., (2015).
Aspera High-speed Transfer.
Opgehaald van: San Francisco DPDK Summit 2015. Website: https://dpdksummit.com/us/en/past-events

Tirumala A., Cottrell L. & Dunigan T., (April, 2003).
*Measuring end-to-end bandwidth with Iperf using Web100**,
Passive and Active Monitoring Workshop.

Totusoft Inc, (2016)
Lan Speedtest (Lite) | Totusoft,
Opgehaald van: http://totusoft.com/lanspeed1/

Velthoven J. K. (Juni, 2013)
Use case diagram, een toelichting,
Opgehaald van: http://www.velthovenconsultancy.nl/diagram/use-case-diagram.

Web10G developers, (2013).
The Web10G Project,
Opgehaald van: https://web10g.org

Weiden N. van der, (2015).
Thesis.cls, Gebruikt voor: Voorblad scriptie.

Bijlage A

Plan van aanpak

Plan van Aanpak

Auteur:
Arthur van der Weiden, 1619815

Februari, 2016

Bedrijf:	Nikhef
Bedrijfsbegeleiders:	Tristan Suerink, Ronald Starink
Docentbegeleider:	Harry Beerlage
In opdracht van:	Hogeschool Utrecht

Inhoudsopgave

1 Inleiding

In dit document staat het plan van aanpak beschreven. Dit plan van aanpak omschrijft het afstudeerproject van Arthur van der Weiden bij Nikhef. Voor dit afstudeerproject moet een onderzoek gedaan worden naar een mogelijke testsuite, deze testsuite moet ontworpen worden, zodat datasnelheden tussen de 40 gbit/s en 100 gbit/s getest kunnen worden. Nikhef is een onderzoeksinstituut. Nikhef werkt nauw samen met andere onderzoeksinstituten en draait dus ook tests op de deeltjesversnellers en andere testsystemen van deze instituten. Vanuit CERN alleen worden al petabytes aan data verzonden die door het Nikhef verwerkt worden.

1.1 Aanleiding

Binnen de servers van het Nikhef wordt veel data verwerkt per seconde. Om de snelheid van de servers te kunnen testen wordt op dit moment een tool gebruikt dat problemen heeft met snelheden testen die groter zijn dan 40 gbit/s. Omdat bij het Nikhef datasnelheden van meer dan 100 gbit/s bereikt kunnen worden, is dit een probleem omdat deze datasnelheden niet betrouwbaar getest kunnen worden. Deze tests worden uitgevoerd ter controle van de datasnelheid. Om dit probleem op te lossen moet een onderzoek gedaan worden naar een mogelijke testsuite, deze testsuite moet ook ontworpen worden, zodat deze snelheden wel getest kunnen worden.

1.2 Leeswijzer

Eerste deel
In het eerste deel van dit plan van aanpak wordt de opdracht beschreven. Het eerste deel loopt van hoofdstuk 2 tot en met hoofdstuk 4. Hierin staan de volgende onderdelen:

- Nikhef
- De opdracht
- Theoretisch kader

Tweede deel
In het tweede deel van dit plan van aanpak staat de organisatie van het afstudeerproject. Het tweede deel bestaat uit hoofdstuk 5. In dit deel staan de onderdelen:

- Beschrijving van de organisatie
- Risico's
- Planning

Derde deel

In dit derde deel zijn de gegevens van alle betrokkenen te vinden. Het derde deel is te vinden in hoofdstuk 6. De betrokkenen zijn onder andere:

- Bedrijfsbegeleiders
- Afstudeerbegeleider
- Afstudeerder

Laatste deel

Uiteindelijk zijn, in het laatste deel van dit plan van aanpak, de samenvatting, de begrippenlijst en een lijst van de figuren te vinden.

2 Nikhef

In dit hoofdstuk wordt Nikhef beschreven. Ook wordt in dit hoofdstuk de kwestie en de reden van de kwestie uitgelegd.

2.1 De context

Nikhef is het Nationaal instituut voor subatomaire fysica in Amsterdam. Bij Nikhef wordt onderzoek gedaan naar deeltjes- en astrodeeltjesfysica. Bij dit soort onderzoeken worden petabytes aan data gegenereerd en die data moet dus ook worden verwerkt. (*Nikhef*, 2016) Om deze petabytes te kunnen verwerken zijn snelle en betrouwbare serververbindingen nodig. De opdracht, in de volgende paragrafen verder uitgelegd, is dan dus ook om te onderzoeken hoe het mogelijk is deze hoge serververbindingen te testen met slechts één machine. De opdracht zal worden uitgevoerd op de afdeling computer technologie van Nikhef. Binnen deze afdeling werken ICTers met verschillende specialiteiten. De taken die uitgevoerd dienen te worden zijn: Onderzoek doen naar de beste library/tool om het dataverkeer te testen; Een testsuite ontwikkelen en uittesten. Mocht tijdens het ontwikkelen blijken dat de gekozen oplossing niet werkt, dan wordt het onderzoek uitgebreidt en wordt onderzocht wat de mogelijke alternatieven zijn.

2.2 De aanleiding

Zoals in de inleiding is uitgelegd wordt binnen de servers van het Nikhef veel data per seconde verwerkt. Om de snelheid van de servers te kunnen testen wordt op dit moment een tool gebruikt dat problemen heeft met snelheden testen die groter zijn dan 40 gbit/s. Om deze snelheden te kunnen testen moet meer dan één machine gebruikt worden om te verzenden. Dit zorgt voor nauwkeurigheidsverlies. Omdat bij Nikhef datasnelheden van meer dan 100 gbit/s bereikt kunnen worden, is dit een probleem omdat deze datasnelheden niet goed en eenvoudig getest kunnen worden. Om dit probleem op te lossen moet een testsuite ontworpen worden waarmee deze snelheden wel getest kunnen worden.

2.3 De kwestie

2.3.1 Huidige situatie

Het testen van de transfersnelheid van een netwerk gebeurt bij Nikhef met Iperf. Iperf is een tool waarmee de maximale bandbreedte van een ip-netwerk getest kan worden. Bij elke test worden bij de client minimaal de volgende onderdelen als resultaten gegenereerd: Bandbreedte, tijdsinterval en verzonden pakketten/datagrammen. Bij de server wordt ook andere data gegenereerd, namelijk de jitter (UDP) en hoeveel datagrammen verloren zijn gegaan tijdens de test (UDP).
Iperf heeft veel verschillende features. Zo is Iperf cross-platform bruikbaar, zijn Iperf servers multithreaded en kan Iperf TCP, UDP en SCTP verbindingen testen (*Iperf ontwikkelaars*, 2016).

Verder heeft Iperf de mogelijkheid om verschillende opties aan te zetten bij een test. De belangrijkste opties zijn:

- Bepaalde tijd testen (-t)
  Om de bandbreedte te testen over een langere tijd is het mogelijk om de duur van een test in te stellen met de optie -t. Deze optie krijgt één parameter mee: de tijd in seconden.
- Server als daemon (-D)
  Het is mogelijk om de server als daemon te laten draaien. Dit kan worden ingesteld met de optie -D.
- TCP slow-start negeren (-O)
  TCP heeft een slow-start mechanisme. Dit mechanisme wordt gebruikt om op-eenhoping van pakketten in een netwerk te voorkomen. Om dit punt in de TCP verbinding te negeren wordt de optie -O aangezet.
- Target bandbreedte instellen (-b)
  Om de target bandbreedte in te stellen wordt de optie -b gebruikt. Deze optie verwacht één parameter, namelijk de target bandbreedte in Mbit/s.

2.3.2 Problemen

Het probleem met Iperf is dat bij het testen van datasnelheden die hoger liggen dan 40 gbits/s de tests met meerdere machines als zenders uitgevoerd moeten worden. Omdat deze testresultaten dus onbetrouwbaar zijn, kunnen netwerken met snelheden boven 40 gbits/s niet goed getest worden. Deze kwestie is van toepassing op de computer-technologie afdeling van Nikhef. De kwestie wordt door Nikhef opgepakt, omdat de netwerken nauwkeuriger getest moeten worden en de tests dus vanaf één machine gestart moeten kunnen worden.

2.4 De doelstellingen

Het doel van de afstudeeropdracht is om een onderzoeksrapport op te leveren, met een advies en een testsuite. Het onderzoeksrapport zal gaan over de mogelijkheden wat betreft het testen van datasnelheden van mogelijk meer dan 40 Gbit/s tot 100 Gbit/s Deze doelstellingen zijn afgeleid uit de problemen die beschreven zijn in de paragraaf"*De kwestie*". De opdracht is dus geslaagd wanneer een succesvol onderzoek is afgerond. Als uit dit onderzoek blijkt dat het mogelijk is dan dient een zo volledig mogelijke testsuite ontwikkeld te worden als proof-of-concept.

2.5 Instituut

Nikhef is het Nationaal instituut voor subatomaire fysica in Amsterdam. Bij Nikhef wordt onderzoek gedaan naar deeltjes- en astrodeeltjesfysica. Bij dit soort onderzoeken

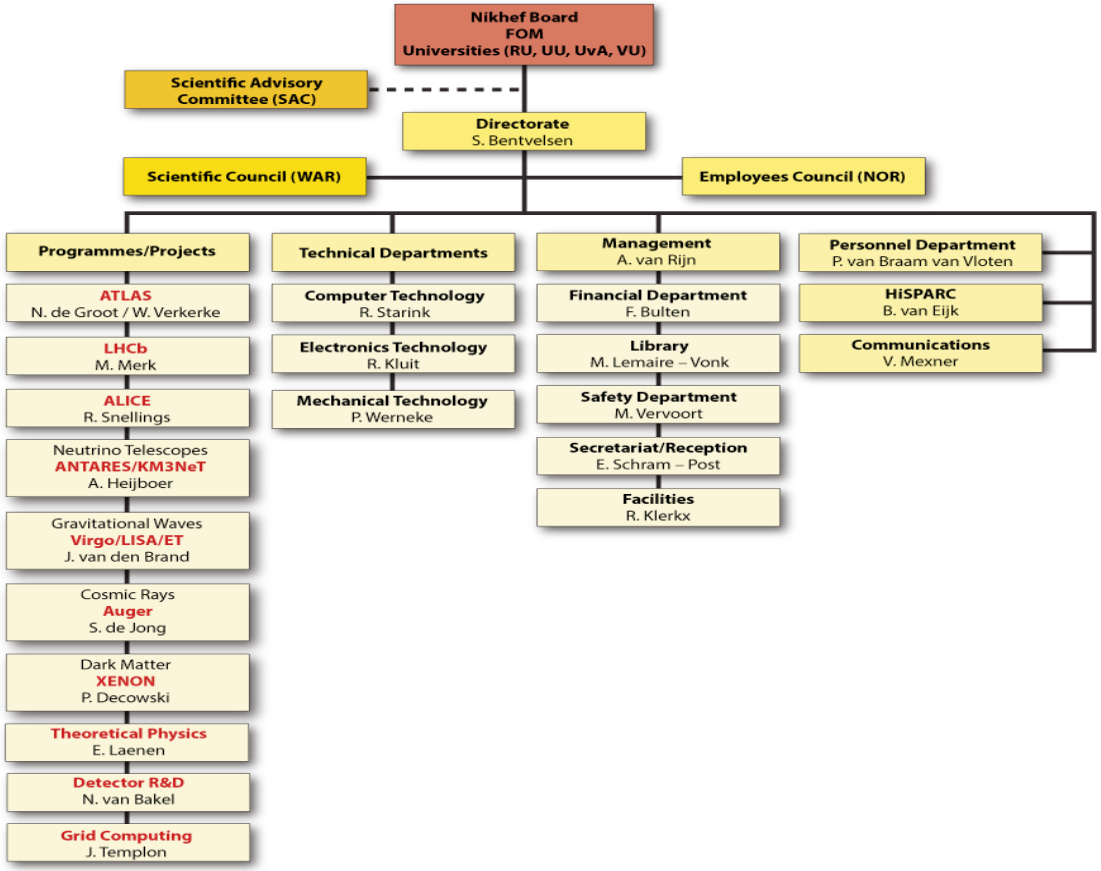
worden petabytes aan data gegenereerd en die data moet dus ook worden verwerkt. Bij Nikhef zijn ongeveer 300 mensen werkzaam (*Nikhef*, 2016).

2.5.1 Organisatie binnen Nikhef

Zoals in figuur 1 te zien is bestaat de organisatie binnen Nikhef uit vier hoofdlagen, namelijk:

- Nikhef board
- Directoraat, wetenschapsraad en personeelsraad
- Afdelingen
- Groepen

(*Nikhef*, 2016)



Figuur 1: Organogram Nikhef

2.5.1.1 Nikhef board

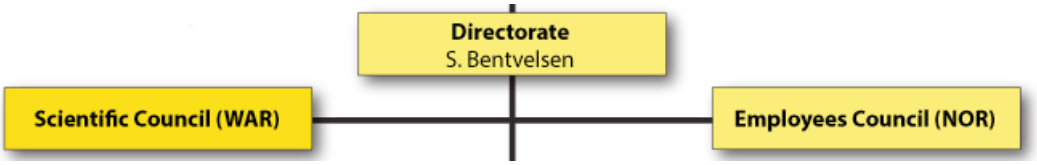
Zoals in figuur 2 te zien is, bestaat het Nikhef board, laag één, uit afgevaardigde van de universiteiten Radboud Universiteit Nijmegen, Vrije Universiteit Amsterdam, Universi-

teit Utrecht, Universiteit van Amsterdam en FOM. Het Nikhef board bestaat uit zeven afgevaardigden: één per universiteit en drie van FOM (*Nikhef*, 2016).

Figuur 2: Nikhef board

2.5.1.2 Directoraat, wetenschapsraad en personeelsraad

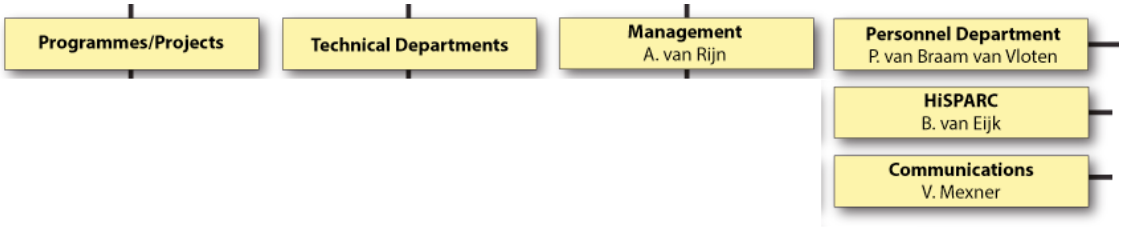
Zoals te zien is in figuur 3 bestaat de volgende laag, laag twee, uit het directoraat, Dr. Prof. Stan Bentvelsen, de WAR en de NOR. WAR staat voor Wetenschappelijke Advies Raad. NOR staat voor Nikhef Ondernemingsraad. De NOR verzorgt de inbreng van het personeel op het beleid van Nikhef. De NOR wordt samengesteld uit kandidaten gekozen door het personeel (*Nikhef*, 2016).



Figuur 3: Nikhef directoraat, wetenschapsraad en personeelsraad

2.5.1.3 Afdelingen

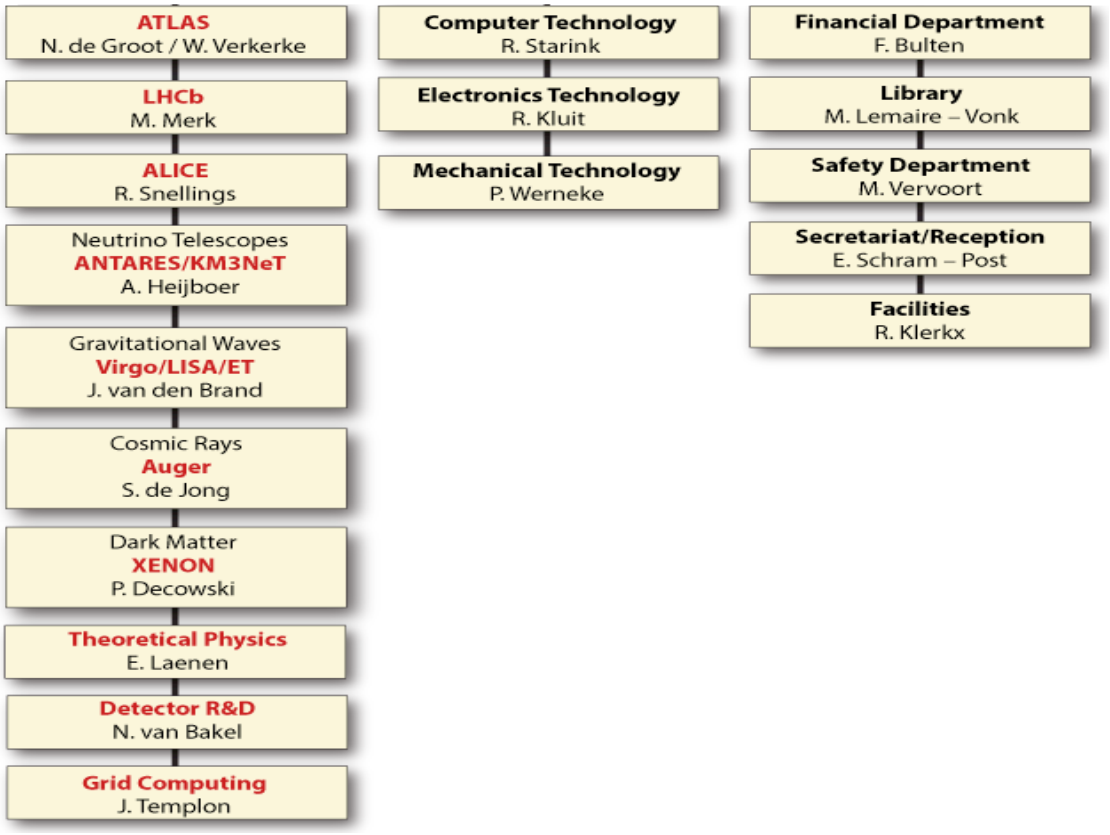
Zoals in figuur 4 is te zien, heeft Nikhef zes verschillende afdelingen. De afdelingen zijn: Projecten, Technische Afdelingen, Management, Personeelszaken, HiSPARC en Communicaties. Bij de afdeling projecten worden de verschillende onderzoeken naar elementaire deeltjes e.d. gedaan. De technische afdelingen vervullen een ondersteunende rol voor deze projecten. De technische afdelingen zijn namelijk voortdurend bezig met het ontwikkelen van nieuwe technieken. HiSPARC is een netwerk van middelbare scholen en wetenschappelijke instituten. Bij dit netwerk wordt kosmische straling met extreem hoge energie gemeten (*Nikhef*, 2016).



Figuur 4: Nikhef afdelingen

2.5.1.4 Groepen

In figuur 5 zijn de verschillende groepen binnen Nikhef te zien. Deze groepen zijn een soort subafdelingen. In dit plan van aanpak wordt de enige relevante groep uitgelegd, zodat het plan van aanpak niet te lang wordt. De enige relevante groep is: Computertechnologie, dit is tevens de plaats van uitvoering. De taak van computertechnologie is het beheren van de computer- en netwerkinfrastructuur. Verder levert computertechnologie een bijdrage aan de verschillende uitlees- en besturingssystemen van de instrumenten voor deeltjesdetectoren (*Nikhef*, 2016).



Figuur 5: Nikhef groepen

2.5.2 Plaats van de student

De plaats van de student in de organisatie is tevens het onderdeel waarvan de opdracht afkomstig is. Deze plaats is de afdeling computertechnologie. Zoals in paragraaf 2.5.1.4 al is uitgelegd zijn de taken van deze afdeling het beheren van de computer- en netwerkinfrastructuur en een bijdrage leveren aan de verschillende uitlees- en besturingssystemen van de instrumenten voor deeltjesdetectoren(*Nikhef*, 2016).

3 De opdracht

In dit hoofdstuk wordt de opdracht beschreven.

3.1 Omschrijving van de opdracht

De opdracht is om een onderzoek uit te voeren naar de mogelijkheden voor het ontwerpen van een testsuite, waarmee datasnelheden tussen de 40 gigabit/seconde en 100 gigabit/seconde verder is de opdracht het ontwerpen en ontwikkelen van de testsuite. Allereerst moet dus onderzocht worden welke libraries/tools beschikbaar zijn die dit mogelijk maken. Vervolgens wordt een onderzoek gedaan naar de beste library/tool die beschikbaar is. Uiteindelijk wordt met die tool een testsuite ontworpen of ontwikkeld. De opdracht is dus voornamelijk een onderzoeksopdracht.

3.2 Hoofd- en deelvragen

3.2.1 Hoofdvraag

De hoofdvraag van dit onderzoek is de volgende:

Wat is de beste library of tool om een testsuite te ontwerpen waarmee datasnelheden tot ongeveer 100 gigabit/seconde getest kunnen worden door slechts één machine te gebruiken en hoe kan hiermee een testsuite ontwikkeld worden?

3.2.2 Deelvragen

Om antwoord te krijgen op de hoofdvraag zullen eerst een aantal deelvragen beantwoord moeten worden. Die deelvragen zijn:

1. **Hoe werkt de tool, iperf, die op dit moment gebruikt wordt voor het testen van server snelheden?**
   Het doel van deze deelvraag is om als vooronderzoek te dienen. Door deze deelvraag te beantwoorden wordt de benodigde voorkennis verzameld, waarmee de rest van het onderzoek makkelijker wordt. De functie van deze deelvraag is descriptief, omdat deze deelvraag de werking van de tool iperf verklaard. De onderzoeksmethoden die bij deze deelvraag gebruikt gaan worden zijn observatie en een literatuuronderzoek. Het literatuuronderzoek zal bestaan uit het gebruiken van de documentatie van iperf. Het observeren zal gebeuren door iperf daadwerkelijk te gebruiken. Als deze twee methoden toegepast zijn, kan de deelvraag goed beantwoord worden.

2. **Wat zijn de specifieke eisen aan de testsuite, naast de te bereiken testsnelheden?**
   Deze deelvraag moet beantwoord worden voordat de rest van het onderzoek kan beginnen. Dit is nodig omdat het onderzoek afhangt van de eisen. Deze eisen

bepalen namelijk welke library/tool als beste naar voren komt. Deze deelvraag is prescriptief, hij helpt namelijk bij het ontwerpen van de testsuite. Om antwoord op deze deelvraag te krijgen zal een interview met de bedrijfsbegeleider gehouden worden. Dit interview zal het antwoord bieden op deze deelvraag.

3. **Wat zijn de besturingssysteem rechten die de library/tool nodig heeft om goed te kunnen functioneren?**
   Deze deelvraag speelt door op de eisen die door de vorige deelvraag aan de testsuite gesteld worden. Als iemand zonder root/admin rechten de testtool moet gebruiken is het nodig dat de testsuite deze rechten dus ook niet nodig heeft. Deze deelvraag is prescriptief, omdat de rechten van belang zijn tijdens het ontwerpen van de testsuite. Bij deze deelvraag wordt een literatuuronderzoek uitgevoerd. Bij dit literatuuronderzoek wordt de documentatie van de verschillende libraries/tools gebruikt. Uit deze documentatie zal blijken welke rechten de library/tool nodig heeft om goed te functioneren.

4. **Wat is de beste library/tool om te gebruiken bij het ontwikkelen van de testsuite?**
   Deze deelvraag dient om het eerste gedeelte van de hoofdvraag te beantwoorden. De deelvraag is een van de belangrijkste van het onderzoek, dit omdat deze deelvraag bepaalt welke library/tool gebruikt gaat worden voor het ontwerpen/ontwikkelen van de testtool. Deze deelvraag is descriptief, omdat tijdens het beantwoorden van de deelvraag de verschillende libraries/tools met elkaar vergeleken zullen worden. Om antwoord te krijgen op deze deelvraag wordt gekeken naar de antwoorden op de vorige deelvragen. De eisen die aan de library/tool gesteld zullen worden, worden gehaald uit het antwoord van deelvraag 2.

5. **Hoe werkt de library/tool die als beste uit het voorgenoemde onderzoek komt precies?**
   Om een testtool te kunnen ontwikkelen met behulp van de library/tool moet onderzocht worden hoe deze library/tool precies werkt. Om daar achter te komen dient deze deelvraag beantwoord te worden. Deze deelvraag is descriptief, omdat deze deelvraag de werking van de beste tool verklaard. Bij deze deelvraag wordt gebruik gemaakt van de onderzoeksmethoden observatie en literatuuronderzoek. Net zoals bij deelvraag 1 wordt eerst gekeken naar de documentatie van de betreffende library/tool. Vervolgens zal de tool gebruikt worden en geobserveerd worden hoe de tool werkt.

6. **Welke aspecten heeft het proof-of-concept nodig om geslaagd te zijn?**
   Deze deelvraag zal helpen bij het ontwerpen van het proof-of-concept. In die zin dat het proof-of-concept dan aan alle eisen zal voldoen om geslaagd te zijn. Of, als dit niet binnen het kader mogelijk is, een verder onderzoek gestart kan worden naar alternatieve oplossingen.

3.3 Te onderzoeken literatuur

De belangrijkste literatuur die onderzocht dient te worden is de documentatie van de verschillende libraries/tools. Uit deze documentatie zal de benodigde informatie komen om een evenwichtige keuze tussen de verschillende kandidaten te kunnen maken. Verder zal de documentatie en de source code van de beste library/tool verder onderzocht worden.

3.4 Op te leveren producten

Bij de opdracht worden de volgende producten opgeleverd:

Plan van Aanpak:
Voordat het project officieel van start gaat wordt een plan van aanpak geschreven. Het plan van aanpak moet voldoen aan de eisen uit de afstudeerleidraad.

Onderzoeksrapport:
In de eerste fase, de onderzoeksfase, wordt een onderzoeksrapport geschreven. In dit onderzoeksrapport staat beschreven wat de beste library is om te gaan gebruiken in de ontwikkelfase.

Testsuite:
In de tweede fase, de ontwikkelfase, wordt, indien mogelijk, een testsuite ontwikkeld. Het doel van deze testsuite is om de huidige testmogelijkheden te verbeteren naar datasnelheden tussen de 40- en 100 gigabit/seconde. Ook moet de testsuite alle bekende Linux/Unix distributies ondersteunen. Zoals bijvoorbeeld: Ubuntu, CentOS, Debian en FreeBSD.

Afstudeerscriptie:
Uiteindelijk wordt ook een afstudeerscriptie opgeleverd. Deze scriptie moet voldoen aan de eisen uit de afstudeerleidraad.

3.5 Ontwikkelmethode

De opdracht zal worden uitgevoerd in twee fasen: de onderzoeksfase en de ontwikkelfase, in de subparagrafen van deze paragraaf wordt verder uitgelegd welke activiteiten beide fasen bevatten.

3.5.1 Onderzoeksfase

In de onderzoeksfase wordt voorbereidend onderzoek gedaan naar DPDK en diverse andere soortgelijke libraries. Tijdens het onderzoeken worden de verschillende libraries dus vergeleken. De libraries worden vergeleken op verschillende punten. Deze punten

zullen later nog bepaald worden, maar zullen in de richting van de volgende punten zijn: Maximale datasnelheid, eenvoud in onderhoud en ondersteuning voor Linux.

3.5.2 Ontwikkelfase

Tijdens de ontwikkelfase zal de testsuite ontwikkeld worden met behulp van de library die als beste uit het onderzoek komt. De ontwikkelmethode die hier gebruikt zal gaan worden lijkt op SCRUM met als grootste verschil dat het een eenmansproject is. De onderdelen die van SCRUM overgenomen worden zijn: de sprints en de sprint rapporten. Ook zal voor iedere sprint een nieuwe planning gemaakt worden.

3.6 Resultaten

De verschillende fasen zullen ook verschillende eindresultaten hebben. Zo zal uit de onderzoeksfase een onderzoeksrapport komen. Dit onderzoeksrapport zal duidelijkheid verschaffen over de volgende zaken: welke library/tool het beste gebruikt kan worden en de werking van de beste library/tool. Zodra het onderzoek is afgerond kan de ontwikkelfase beginnen. Door het onderzoek dat verricht is in de onderzoeksfase van het project, wordt de ontwikkelfase eenvoudiger.
De ontwikkelfase heeft als eindresultaat een testsuite voor het testen. Tijdens de ontwikkelfase bestaan echter verschillende deelresultaten. Deze deelresultaten worden opgesteld in de sprintrapporten. Per sprint wordt bekeken wat de doelen voor de volgende deadline zijn. Na iedere sprint zal met de bedrijfsbegeleider overlegt worden wat de volgende stap zal zijn, op deze manier zal voldaan worden aan de kwaliteitseisen. Deze resultaten zorgen ervoor dat de kwestie opgelost wordt. Dit gebeurt namelijk door de testsuite te ontwerpen. De kwaliteitscriteria zijn met de opdrachtgever, de bedrijfsbegeleider, besproken. De kwestie, het onderzoeken naar de mogelijkheden om de problemen met het testen van dataverkeer bij snelheden boven de 40 gbit/s vanaf één machine op te lossen, zal worden opgelost door middel van de testsuite. Om deze testsuite goed te kunnen ontwikkelen wordt eerst een onderzoek uitgevoerd. De testsuite zal dienst doen als een proof-of-concept. De testsuite moet UDP en TCP kunnen testen, snelheden van meer dan 40 gbit/s kunnen testen en moet werken onder de bekende Linux distributies. De opdracht is dus een onderzoeksopdracht.

3.6.1 MoSCoW

In tabel 1 zijn de eisen verdeelt volgens het MoSCoW criterium.

Tabel 1: MoSCoW

MSCW	**Onderdeel en eis**
Must-have	**Onderzoeksverslag:** Beste library/tool gekozen **Testsuite:** Meer dan 40 gbit/s Eenvoudig bruikbaar Werkend onder bekende Linux distributies UDP en TCP testen
Should-have	**Onderzoeksverslag:** n.v.t. **Testsuite:** Tot 100 gbit/s Eenvoudig uitbreidbaar Werkend onder andere Unix producten (Os X e.d.)
Could-have	n.v.t.
Won't-have	**Testsuite:** Windows ondersteuning

3.7 Risico's

100 gigabit/seconden is niet te bereiken
De kans dat dit risico uitkomt is vrij klein omdat de bedrijfsbegeleider al enig onderzoek heeft gedaan. Het is echter toch een risico waar enigszins rekening mee gehouden dient te worden. Als dit toch het geval blijkt te zijn, dan wordt een testsuite ontworpen/ontwikkeld die in ieder geval boven de 40 gigabit/seconden kan testen.

De voortgang gaat verloren vanwege een crash
Het is mogelijk dat de harde schijf met daarop de scriptie crasht of de harde schijf met daarop de code crasht. Wanneer dat zou gebeuren raakt de hele voortgang verloren.
Door met verschillende computers gewerkt gaat worden en alle codebestanden en documenten op de server van de afstudeerder worden gezet, bestaat de kans dat alle voortgang verloren wordt vrijwel niet.

4 Theoretisch kader

In dit hoofdstuk wordt het theoretische kader van het project uitgelegd. In de eerste paragrafen zijn de verschillende belangrijke begrippen en modellen uitgelegd. Ook is, in de laatste paragraaf, de ondersteunende literatuur beschreven.

4.1 Methoden

4.1.1 Onderzoeksmethoden

Het onderzoek bestaat, zoals altijd, uit verschillende deelvragen. In deze paragraaf wordt per deelvraag bekeken wat de beste onderzoeksmethode is.

4.1.1.1 Deelvraag 1

Hoe werkt de tool, iperf, die op dit moment gebruikt wordt voor het testen van server snelheden?
De onderzoeksmethoden die bij deze deelvraag gebruikt gaat worden zijn observatie en een literatuuronderzoek. Het literatuuronderzoek zal bestaan uit het doornemen van de documentatie van iperf. Het observeren zal gebeuren door iperf daadwerkelijk te gebruiken. Als deze twee methoden toegepast zijn, kan de deelvraag goed beantwoord worden.

4.1.1.2 Deelvraag 2

Wat zijn de specifieke eisen aan de testsuite, naast de te bereiken testsnelheden?
Om antwoord op deze deelvraag te krijgen zal een interview met de bedrijfsbegeleider gehouden worden. Dit interview zal het antwoord bieden op deze deelvraag.

4.1.1.3 Deelvraag 3

Wat zijn de besturingssysteem rechten die de library/tool nodig heeft om goed te kunnen functioneren?
Bij deze deelvraag wordt een literatuuronderzoek uitgevoerd. Bij dit literatuuronderzoek wordt de documentatie van de verschillende libraries/tools gebruikt. Uit deze documentatie zal blijken welke rechten de library/tool nodig heeft om goed te functioneren.

4.1.1.4 Deelvraag 4

Wat is de beste library/tool om te gebruiken bij het ontwikkelen van de testsuite?
Om antwoord te krijgen op deze deelvraag wordt gekeken naar de antwoorden op de vorige deelvragen. De eisen die aan de library/tool gesteld zullen worden worden gehaald uit het antwoord van deelvraag 2.

4.1.1.5 Deelvraag 5

Hoe werkt de library/tool die als beste uit het voorgenoemde onderzoek komt precies?
Bij deze deelvraag wordt gebruikt gemaakt van de onderzoeksmethoden observatie en literatuuronderzoek. Net zoals bij deelvraag 1 wordt eerst gekeken naar de documentatie van de betreffende library/tool. Vervolgens zal de tool gebruikt worden en geobserveerd worden hoe de tool werkt.

4.1.1.6 Deelvraag 6

Welke aspecten heeft het proof-of-concept nodig om geslaagd te zijn? Tijdens het interview van deelvraag 2 zullen ook enkele vragen gesteld worden over het proof-of-concept. Op deze manier zal deze deelvraag beantwoord kunnen worden.

4.2.2 Modellen

Om de eisen van het project te verdelen voor de ontwikkelfase zal MoSCoW worden gebruikt. MoSCoW is een model waarbij de eisen verdeeld worden in *Must-haves*, *Should-have*, *Could-haves*, *Won't-haves*. Deze verdeling verwijst naar de prioriteiten van de verschillende eisen. De verdeling is gesorteerd op prioriteit met de *Must-haves* als hoogste prioriteit en de *Won't-haves* als laagste prioriteit

4.2.3 Ontwikkelmethode

De ontwikkelmethode die tijdens het ontwikkelen van de testsuite gebruikt zal gaan worden lijkt op SCRUM met als grootste verschil dat het een eenmansproject is. De onderdelen die van SCRUM overgenomen worden zijn: de sprints en de sprintrapporten. Ook zal voor iedere sprint een nieuwe planning gemaakt worden.

4.2.4 Ontwerpmethode

De testsuite zal ontworpen worden aan de hand van een interview met de opdrachtgever. Omdat de layout en syntax van de testsuite niet uitmaakt, zal aan het ontwerpen van de testsuite weinig tijd besteed worden. De tijd die hiermee gewonnen wordt zal worden gebruikt worden voor het ontwikkelen van de testsuite.

4.3 Ondersteunende literatuur

In deze paragraaf is de ondersteunende literatuur te vinden die nodig zou kunnen zijn. Ook wordt per stuk literatuur uitgelegd waarvoor het handig/nodig kan zijn.

4.3.1 Unix and Linux system administration handbook

Dit handboek staat vol handige tips voor Unix- en Linux systemen. Tijdens de afstudeeropdracht wordt veel op Linux gewerkt. De voorkennis om met Linux te kunnen werken is al aanwezig. Toch is het handig om enige boeken over Linux bij het onderzoek te houden. Dit komt door de kleine verschillen die bestaan tussen, het in de verte op Unix gebaseerde, Os X (het operating system waar de afstudeerder mee werkt) en de verschillende Linux versies. Ook staan in dit handboek veel tips over het werken met netwerken en Linux. Aangezien de afstudeeropdracht voornamelijk over netwerken en Linux gaat is dit dus ook handig om bij het onderzoek te hebben.

4.3.2 DPDK en iperf documentatie

Volgens de bedrijfsbegeleider is DPDK een zeer geschikte kandidaat. Om deze reden is het plan ontstaan om zeker naar DPDK te kijken. Tijdens het onderzoek is het dus zeer belangrijk om naar de documentatie van DPDK te kijken.
In de huidige situatie bij het Nikhef wordt getest met behulp van de tool iperf. Om een goed vooronderzoek naar iperf te kunnen volbrengen is de documentatie van iperf belangrijk. Natuurlijk moet ook gekeken worden naar de documentatie van de overige libraries/tools. Echter omdat deze nog niet op dit moment bekend zijn, worden deze in dit plan van aanpak niet bij naam genoemd.

4.3.3 Computer networking, a top-down approach

Het boek *Computer networking, a top-down approach* is een boek met alle belangrijke informatie betreffend netwerken en netwerkprotocollen. Omdat er veel met netwerken gewerkt gaat worden en dus ook met netwerkprotocollen is het handig om snel informatie op te kunnen zoeken over dergelijke zaken.

4.3.4 Leren Communiceren

De verslagen en scripties moeten allemaal worden geschreven volgens de regels van het boek *Leren Communiceren*. Om deze reden is het dus handig om dit boek bij het schrijven van de verslagen en scripties te houden.

5 De projectorganisatie

5.1 Beschrijving van de organisatie

Het project zal uit 2 fasen bestaan. Hieronder staat de manier beschreven waarop de activiteiten in de verschillende fasen verdeeld zijn.

Fase 1 - Onderzoek
Tijdens de eerste fase wordt het onderzoek gedaan. Bij dit onderzoek wordt gekeken naar verschillende libraries/tools waarmee de bandbreedte van een netwerk getest kan worden. Tijdens dit onderzoek wordt bepaalt welke van deze libraries/tools het beste is om de wensen betreffende de testsuite van de opdrachtgever te vervullen. Als blijkt dat geen van de libraries/tools de 100 gigabits/seconde kan behalen dan wordt de testsuite ontworpen voor datasnelheden met minimaal 40 gigabits/seconde. Wanneer het onderzoek is beeindigd is het onderzoeksrapport af.

Fase 2 - Ontwikkelfase
In de tweede fase wordt, indien mogelijk, de testsuite ontworpen/ontwikkeld. Bij het ontwerpen/ontwikkelen van de testsuite wordt gebruik gemaakt van de library/tool die als beste uit het onderzoek komt. Aan het einde van de ontwikkelfase is de testsuite bruikbaar.

5.2 Ontwikkelmethode

De ontwikkelmethode die tijdens het ontwikkelen van de testsuite gebruikt zal gaan worden lijkt op SCRUM met als grootste verschil dat het een eenmansproject is. De onderdelen die van SCRUM overgenomen worden zijn: de sprints en de sprintrapporten. Ook zal voor iedere sprint een nieuwe planning gemaakt worden. Deze manier van ontwikkelen past bij dit project doordat iedere twee weken een werkend stuk software opgeleverd kan worden. Verder garandeert deze ontwikkelmethode een duidelijk inzicht van de projectvoortgang. Dit komt door de sprintrapporten die iedere twee weken opgeleverd worden. Ook bewaakt deze methode of de randvoorwaarden behaalt worden, omdat aan het einde van iedere sprint met de opdrachtgever wordt gekeken wat gebeurt is en wat nog moet gebeuren (*Overheem*, 2004).

5.3 Risico's

100 gigabit/seconden is niet te bereiken
De kans dat dit risico uitkomt is vrij klein omdat de bedrijfsbegeleider al enig onderzoek heeft gedaan. Het is echter toch een risico waar enigszins rekening mee gehouden dient te worden. Als dit toch het geval blijkt te zijn, dan wordt een testsuite ontworpen/ontwikkeld die in ieder geval boven de 40 gigabit/seconden kan testen.

De voortgang gaat verloren vanwege een crash
Omdat met verschillende computers gewerkt gaat worden en alle codebestanden en documenten op de server van de afstudeerder worden gezet, bestaat de kans dat alle voortgang verloren wordt vrijwel niet.

Uitdagingen
De grootste uitdaging van deze opdracht is het feit dat ik misschien diep in de linux kernel moet duiken. Aangezien ik dit al een hele tijd niet meer gedaan heb, denk ik dat de kennis enigszins is weggezakt. Verder zal het een uitdaging worden om de beste library/tool te vinden. Dit komt doordat er waarschijnlijk veel libraries met verschillende kwaliteiten zijn, hiervoor zullen hele goede en specifieke eisen gesteld moeten worden aan het begin van het onderzoek. Over deze eisen zal ik met mijn afstudeerbegeleider overleggen, op deze manier weet iedereen zeker dat de library/tool aan de juiste eisen voldoet.

5.4 Tijdlijn

5.4.1 Februari

In de maand februari wordt onderzoek gedaan naar de tool iperf. Verder worden alle libraries/tools verzamelt die verder onderzocht zullen worden. Ook wordt in deze maand het plan van aanpak geschreven. Het schrijven van het plan van aanpak is een iteratief proces en loopt door tot in maart. Het onderzoek start ook in februari en zal ook doorlopen tot in maart.

5.4.2 Maart

In maart worden het plan van aanpak en het onderzoek afgerond. Dit betekent dat in maart dus ook de ontwikkelfase starten zal. De eerste sprint zal volledig plaatsvinden in maart en aan het einde van deze maand zal dus de library/tool die gebruikt gaat worden begrepen en goed ingesteld zijn. Ook wordt op 18 maart 2016 een eerste concept voor de afstudeerscriptie gemaakt.

5.4.3 April

In april vinden de tweede en derde sprint plaats. Wat precies in deze sprints plaats zal vinden moet nog vastgesteld worden. Verder zal de afstudeerscriptie geupdate worden en wordt op 29 april een tweede conceptversie hiervan opgeleverd. De afstudeerscriptie is, evenals het plan van aanpak, een iteratief proces.

5.4.4 Mei

In mei vinden de vierde en vijfde sprint plaats. Ook voor deze sprints geld dat de precieze inhoud nog vastgesteld moet worden. Op 31 mei 2016 wordt de eindversie van de afstudeerscriptie opgeleverd.

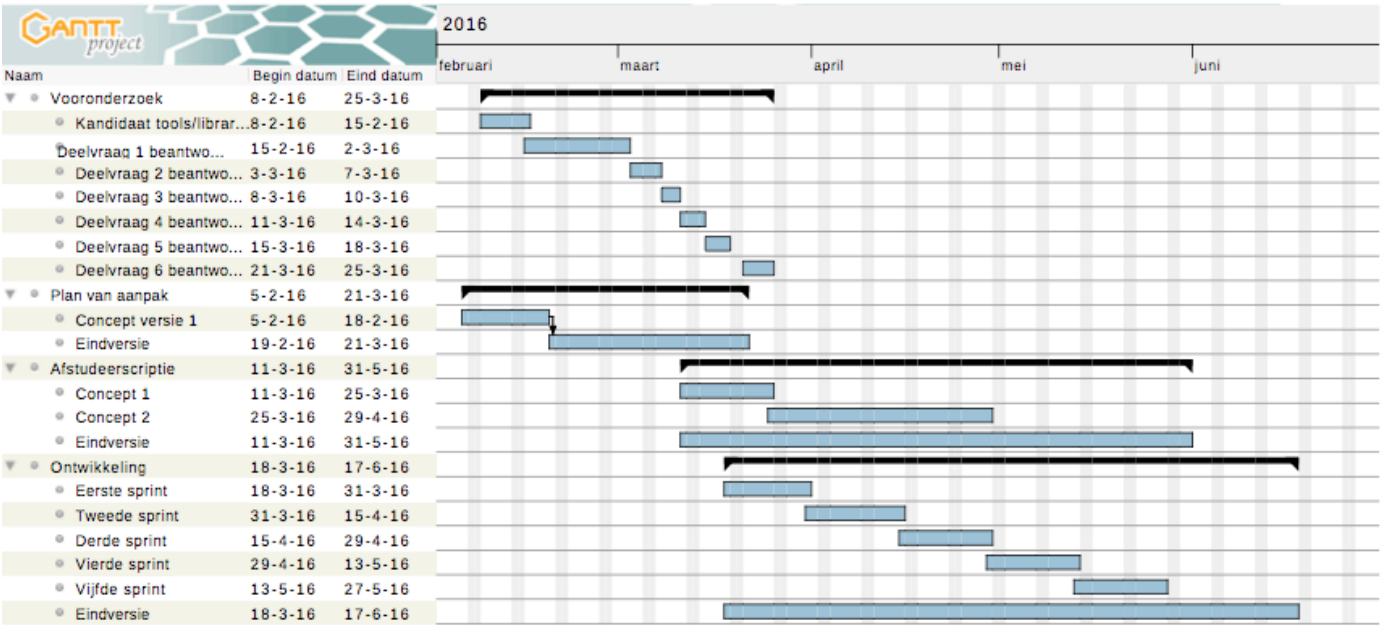
5.4.5 Juni

In juni is de afstudeerzitting gepland, dit betekent dat in juni de eindversie van de testsuite opgeleverd zal worden. Verder wordt in juni de presentatie gemaakt en voorbereidt.

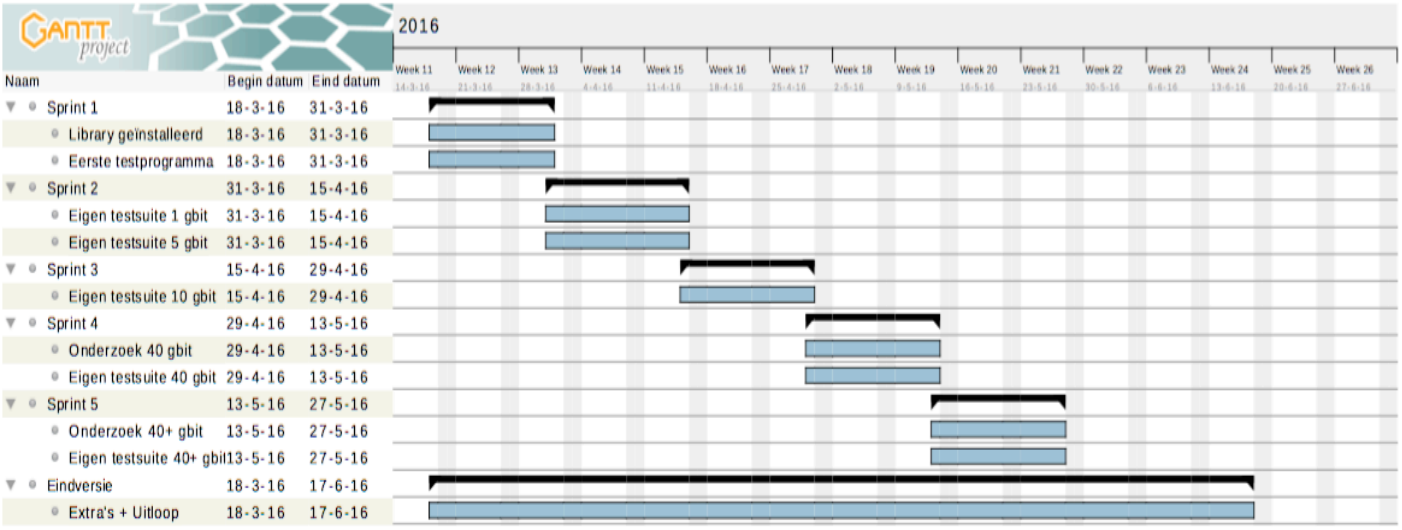
5.5 Planning

In deze paragraaf is een globale planning van het project te vinden. In het eerste Gantt-chart is de totale planning te vinden. Ook staat hierin de planning van de onderzoeksfase. In het tweede Gantt-chart is de detailplanning van de ontwikkelfase te vinden. In de detailplanning van de ontwikkelfase staan de doelen van de verschillende sprints aangegeven.

In de eerste kolom zijn de uit te voeren activiteiten te vinden. Deze activiteiten zorgen ervoor dat het eindresultaat behaald kan worden. De onderzoeksfase zal tot 25 maart 2016 lopen. Aan het einde van deze periode zal dus een beste library of tool gekozen zijn. Daarna start de ontwikkelfase. Tijdens deze fase zal de testsuite ontwikkeld worden. Deze fase vindt plaats in de periode van 18 maart 2016 tot en met 10 juni 2016.



Figuur 6: Totale planning



Figuur 7: Ontwikkeling

6 Betrokkenen

In dit hoofdstuk staan de gegevens van alle betrokkenen. De betrokkenen zijn: Bedrijfsbegeleiders, Tristan Suerink en Ronald Starink; Afstudeerbegeleider: Harry Beerlage; Eerste examinator: Marten Wensink; Afstudeerder: Arthur van der Weiden.

6.1 Bedrijfsbegeleiders

Eerste begeleider: Tristan Suerink

E-mailadres:	t.suerink@nikhef.nl
Telefoonnummer:	+31205925009
Kamer:	H156
Afdeling:	Computer Technologie
Functie:	IT Architect

Tweede begeleider: Ronald Starink

E-mailadres:	r.starink@nikhef.nl
Telefoonnummer:	+31205922271
Kamer:	H133
Afdeling:	Computer Technologie
Functie:	Hoofd Computer technologie

6.2 Afstudeerbegeleider en eerste examinator

Harry Beerlage

E-mailadres:	harry.beerlage@hu.nl
Telefoonnummer:	0541-534191
Mobiel telefoonnummer:	06-22168662

Marten Wensink

E-mailadres:	marten.wensink@hu.nl
Telefoonnummer:	+31884818591

6.3 Afstudeerder

Arthur van der Weiden

E-mailadres:	arthur.vanderweiden@student.hu.nl
Telefoonnummer:	035-6855637
Mobiel telefoonnummer:	06-17791980

6.4 Communicatie

De communicatie tussen de opdrachtgever en de afstudeerder zal direct verlopen. Dit is mogelijk omdat de opdrachtgever tevens de bedrijfsbegeleider is. De communicatie tussen deze partijen zal dus eenvoudig en vlot verlopen.
De communicatie met de opleiding zal verlopen via de afstudeerbegeleider: Harry Beerlage. Deze communicatie zal verlopen via Skype en e-mail. Wanneer de afstudeerbegeleider niet online is op Skype kan de afstudeerder dus een e-mail versturen. Op deze manier zal ook deze communicatie vlot verlopen.
De afstudeerder zal na iedere mijlpaal een update geven aan de afstudeerbegeleider, zodat deze op de hoogte blijft van de voortgang. De afstudeerbegeleider zal ook (minimaal) één keer naar het bedrijf komen om het plan van aanpak te ondertekenen.

7 Samenvatting

In dit document is het plan van aanpak van de afstudeeropdracht bij het Nikhef besproken.

Eerst werden het bedrijf en de opdracht beschreven. De aanleiding van de opdracht is dat Iperf (de tool die in de huidige situatie wordt gebruikt) niet vanaf één machine een test kan draaien met datasnelheden boven de 40 gigabit/seconde. Het doel van de opdracht stond ook in hetzelfde hoofdstuk beschreven. Het doel is om te onderzoeken of het mogelijk is om een testsuite te ontwerpen die de hiervoor genoemde snelheden kan bereiken. Ook is het een doel om, als het mogelijk is, deze testsuite te ontwerpen/ontwikkelen. De hoofdvraag van de opdracht is: Wat is de beste library of tool om een testsuite te ontwerpen waarmee datasnelheden tot ongeveer 100 gigabit/seconde getest kunnen worden en hoe kan hiermee een testsuite ontwikkeld worden? Bij deze hoofdvraag horen vijf deelvragen.

In het tweede deel stond de projectorganisatie beschreven. Hierin werd beschreven hoe het project in twee fasen verdeeld is. De twee fasen die beschreven zijn, zijn het onderzoek (fase 1) en de ontwikkelfase (fase 2). In de onderzoeksfase wordt gekeken naar de verschillende kandidaat libraries/tools en wordt onderzocht welke van deze libraries/tools het beste is om te gebruiken tijdens de ontwikkelfase. Tijdens de ontwikkelfase wordt, indien mogelijk, met deze library/tool een testsuite ontwikkeld/ontworpen. Verder zijn in hoofdstuk 3 de risico's en uitdagingen te vinden.

Als laatste stond in het derde deel beschreven wie de betrokken zijn bij de afstudeeropdracht. Ook staan in dit hoofdstuk de contactgegevens van de betrokkenen en de manier waarop de communicatie zal verlopen.

8 Begrippenlijst

CERN: CERN is een onderzoeksinstituut in Genève. CERN bezit verschillende testsystemen waaronder deeltjesversnellers.

Cross-platform: Een systeem dat cross-platform beschikbaar is werkt op verschillende besturingssystemen. Dit begrip kan ook betekenen dat het op verschillende soorten machines werkt.

Daemon: Een daemon is een Linuxproces dat op de achtergrond draait.

FOM: Stichting voor Fundamenteel Onderzoek der Materie (FOM) is een stichting bestaande uit vier FOM-instituten (FOM AMOLF, FOM DIFFER, FOM Nikhef en Advanced Research Center). Bij deze stichting vindt natuurkundig toponderzoek plaats.

Gbit/s: Gbit/s staat voor Gigabit per seconde. Dit (of Megabit per seconde ”Mbit/s”) is de eenheid die gebruikt wordt voor metingen van snelheid bij dataverkeer.

Jitter: Bij UDP bestaat de term *jitter*. De jitter die tijdens het verzenden optreed is de tijd die tussen het ontvangen van de verschillende datagrammen zit.

Library: Een library is een collectie van programmeercodes waar gebruik van gemaakt kan worden om een nieuw stuk code te schrijven.

Multithreaded: Multithreaded betekent dat een programma of systeem meerdere threads (kleine processen) tegelijk kan afhandelen.

Netwerkprotocol: Een netwerkprotocol is een verzameling afspraken op een netwerk. Communicerende nodes zullen zich aan deze afspraken moeten houden, zodat de communicatie goed kan verlopen.

SCRUM: SCRUM is een iteratieve ontwikkelmethode. Bij SCRUM worden sprints opgesteld, deze sprints zijn de mijlpalen van het project.

Server: Een server is een computer die op afstand benadert kan worden.

SCTP: SCTP, of Stream Control Transmission Protocol, is een netwerkprotocol met de garantie dat de verstuurde data aankomt. SCTP werkt met datapakketten.

TCP: TCP, of Transmission Control Protocol, is een netwerkprotocol met de garantie dat de verstuurde data aankomt. TCP werkt met datastromen.

Testsuite: Een testsuite is een programma dat een aantal tests uitvoert en hiervan

de resultaten teruggeeft.

Tool: Een tool is een hulpmiddel waarmee geprogrammeerd of getest kan worden.

UDP: UDP, of User Datagram Protocol, is de directe tegenhanger van TCP en biedt geen aankomst garantie voor de verstuurde datagrammen. UDP werkt met datagrammen

Bibliografie

Hogeschool Utrecht, (2015). *Afstudeerleidraad Instituut voor ICT*. Faculteit Natuur en Techniek Utrecht. Opgehaald van http://www.voorbedrijven.hu.nl/ /media/hu- bedrijven/docs/praktijkbureau%20cluster%20ict/afstudeerleidraad%20instituut%20voor%20 ict%20cursus%202015-2016.pdf?la=nl.

Iperf developers, (2016). *Iperf, The TCP, UDP and SCTP network bandwith measurement tool*, Opgehaald van https://iperf.fr.

Kurkose, F & Ross, K, (2003). *Computer Networking A Top-Down Approach* Pearson.

Nikhef, (2016). *Nikhef website*, Opgehaald van https://www.nikhef.nl.

Overheem B, (2004). *Wat is SCRUM?*, Opgehaald van http://www.scrum.nl/site/Wat-is-Scrum-agile-scrum.

Lijst van figuren

Lijst van tabellen

Bijlage B

Interview eisen testsuite

1. Mag de testsuite root rechten vereisen?
   Ja dat mag, iedereen die het zal gaan gebruiken heeft tenminste één computer waarop zij root rechten hebben dus dat is geen probleem.

2. Op welke besturingssystemen moet de testsuite kunnen werken?
   In principe op alle Linux distributies. Maar begin eerst maar met de bekende Linux distributies, zoals FreeBSD, RedHat, Ubuntu, Debian etc.

3. Welke protocollen moeten getest kunnen worden? (TCP/UDP/SCTP)
   TCP en UDP moeten zeker allebei getest kunnen worden. Verder willen we ook dat verschillende poortnummers open gezet kunnen worden waar dan over getest kan worden.

4. Welke iPerf opties moeten terugkomen in de testsuite?
   We moeten de maximale snelheid kunnen testen. Ook willen we een snelheid in kunnen stellen waar de testsuite niet overheen zal komen. Verder moet de testsuite een optie hebben waarmee je de slow-start van TCP niet meeneemt in de test en een UDP test waar geen server voor gestart hoeft te worden.

5. Wanneer is het proof-of-concept geslaagd?
   Als minimaal snelheden tussen de 40 en 100 gbit/s gehaald kunnen worden.

6. Zijn er nog meer eisen die niet door deze vragen benoemd zijn en aan het eind eventueel extra toegevoegd kunnen worden?
   Het zou fijn zijn als je vanaf één machine een test op een aantal machines tegelijk kan starten. En als je daarna nog tijd over hebt, kun je van de output ASCII grafiekjes maken.

7. Maakt de sysntax waarmee een test gestart wordt uit?
   De syntax van het starten van een test doet er niet toe.

8. Wanneer is het proof-of-concept geslaagd?
   Het proof-of-concept is geslaagd als datasnelheden tussen de 40 gbit/s en 100 gbit/s getest kunnen worden.

Bijlage C

KNI code

C.1 KNI

Om de pakketgenerator de juiste snelheden te laten bereiken is een KNI toegevoegd. In deze paragraaf wordt duidelijk uitgelegd wat een KNI is en hoe de code van de toegevoegde KNI werkt.

C.1.1 Principe van de KNI

Om duidelijk uit te kunnen leggen wat een KNI doet, zal eerst worden uitgelegd wat het verschil is tussen de user-mode en de kernel-mode. De CPU kan in verschillende toestanden verkeren. Wanneer de CPU in de werktoestand user-mode verkeerd, zijn niet alle instructies uitvoerbaar. Als de CPU in de werktoestand kernel-mode verkeerd, dan zijn wel alle instructies beschikbaar. De reden hiervoor is een goede bescherming tegen gebruikers en hun processen (*Moergestel*, 2012). De KNI zorgt ervoor dat user-mode applicaties toegang kunnen krijgen tot kernel-mode instructies.

C.1.2 Code van de KNI

In deze paragraaf wordt uitgelegd hoe de code van de KNI werkt. Eerst wordt uitgelegd hoe de MTU aangepast wordt, vervolgens wordt uitgelegd hoe de systeemaanroepen omzeild worden. De volledige code van de KNI is te vinden in bijlage C.

Aanpassen van de MTU

Als eerste moest de KNI het mogelijk maken om de MTU grootte aan te passen. Hiervoor was, naast functies voor het alloceren en vrijgeven van ruimte, de functie *static int kni_change_mtu(uint8_t port_id, unsigned new_mtu)* ontwikkeld. Deze functie zorgt ervoor dat, zoals de naam al doet vermoeden, de KNI de MTU kan wijzigen. Om de MTU van een netwerkpoort te kunnen wijzigen moet de poort eerst uitgezet worden. Wanneer een netwerkpoort aanstaat, is de poortconfiguratie namelijk als *read-only* geconfigureerd

en kan de poortconfiguratie dus niet gewijzigd worden. Voor het uitzetten van een netwerkpoort heeft de DPDK library een functie genaamd: *rte_eth_dev_stop(uint8_t port_id)*. Deze functie stopt de netwerkpoort en zorgt ervoor dat de poortconfiguratie als *read/write* geconfigureerd wordt. Zodra de netwerkpoort goed is afgesloten, wordt de poortconfiguratie opgeslagen, zodat de kopie aangepast kan worden, zodat als er iets mis gaat, de originele poortconfiguratie nog intact is. Het aanpassen van de poortconfiguratie hangt af van de aangevraagde grootte van de MTU. Als de nieuwe grootte van de MTU namelijk boven de maximale lengte van een ethernetframe komt (1518 bytes) dan worden in plaats van ethernetframes zogenaamde jumboframes verstuurd. De maximale grootte van een jumboframe is 9000 bytes. In het geval van dit onderzoek gebeurt dit dus bij het versturen van de 1600 bytes aan data, zoals beschreven in paragraaf 9.1.1.1. Als de keuze tussen wel of geen jumboframes is gemaakt, dan worden de nieuwe maximale groottes van de pakketten berekent met de som: $nieuweMTU + ethernetHeaderSize + ethernetFrameCheckSequenceSize$. Dit bij elkaar is dus de grootte van één verstuurd pakket. Als deze nieuwe groottes zijn ingesteld, wordt de nieuwe configuratie naar de poortconfiguratie gekopieerd. Mocht dit niet mogelijk zijn dan genereert de functie een foutmelding en springt de code uit de functie. Als het wel lukt heeft de poort dus een aangepaste configuratie en een grotere MTU. Vervolgens moet de netwerkpoort weer gestart worden. Dit gebeurt met de functie uit de DPDK library: *rte_eth_dev_start(uint8_t port_id)*. Dit kan echter ook mislukken, mocht dat gebeuren dan genereert de functie een andere foutmelding en springt de code uit de functie.

Omzeilen systeemaanroepen

Het omzeilen van de systeemaanroepen gebeurt door middel van een aantal verschillende functies. De twee belangrijkste zijn voor het versturen en ontvangen van de pakketten. Deze functies zijn: *static void kni_rx_handler(struct kni_port_parameters *p)* en *static void kni_tx_handler(struct kni_port_parameters *p)*. De werking van deze twee functies is bijna identiek (het enige verschil is dat de rx functie ontvangt en de tx functie verstuurt), daarom zal alleen de tx_handler kort uitgelegd worden. De kni_tx_handler regelt met behulp van de DPDK library functie rte_eth_tx_burst, de overgang van KNI naar netwerkkaart. De KNI houdt namelijk een lijst van TX pakketten bij in een geheugenbuffer. Deze TX pakketten zijn dan al gegenereerd door Pktgen en worden, zodra de netwerkkaart er klaar voor is, direct naar de netwerkkaart geschreven. Deze manier van pakketten verplaatsen is efficiënter dan met systeemaanroepen, omdat de pakketten nu in één keer vanuit een geheugenbuffer naar de netwerkkaart geschreven kunnen worden, zonder dat hiervoor het besturingssysteem voor nodig is.

```
static int pktgen_kni_change_mtu(uint8_t port_id, unsigned new_mtu){
        int ret;
        struct rte_eth_conf configuration;

        //return DPDK error if port-id is invalid
        if (port_id >= rte_eth_dev_count()) {
                RTE_LOG(ERR, APP, "Port id %d is invalid\n", port_id);
```

```
                return -EINVAL;
        }

        rte_eth_dev_stop(port_id);

        //store the port configuration
        memcpy(&configuration, &port_conf, sizeof(configuration));

        if (new_mtu > ETHER_MAX_LEN)
                configuration.rxmode.jumbo_frame = 1;
                //if the new MTU exceeds the max ethernet frame length
                // the ethernet frames become jumbo frames
        else
                configuration.rxmode.jumbo_frame = 0;


        configuration.rxmode.max_rx_pkt_len = new_mtu + KNI_ENET_HEADER_SIZE +
                                                        KNI_ENET_FCS_SIZE;
        configuration.txmode.max_tx_pkt_len = new_mtu + KNI_ENET_HEADER_SIZE +
                                                        KNI_ENET_FCS_SIZE;

        ret = rte_eth_dev_configure(port_id, 1, 1, &configuration);

        if (ret < 0) {
                //try to reconfigure the port using the DPDK configure function
                RTE_LOG(ERR, APP, "Port %d failed to reconfigure \n", port_id);
                return ret;
        }

        ret = rte_eth_dev_start(port_id);
        //try to restart the port using the DPDK restart function
        if (ret < 0) {
                RTE_LOG(ERR, APP, "Fail to restart port %d\n", port_id);
                return ret;
        }

        // if everything succeeds return 0
        return 0;

}

static void pktgen_kni_rx_handler(struct kni_port_parameters *port){
        uint8_t i, port_id;
        uint32_t no_kni;
        unsigned rx_amount, num;
        struct rte_mbuf *packets_bursts[PKT_BURST_SZ];

        //return if no parameters available
        if (port == NULL)
                return;

        no_kni = port->no_kni;
        port_id = port->port_id;

        for(i = 0; i < no_kni; i++){
                rx_amount = rte_eth_rx_burst(port_id, 0, packets_bursts, PKT_BURST_SZ);
                if(rx_amount > PKT_BURST_SZ) {
                        RTE_LOG(ERR, APP, "Error receiving data\n");
                        return;
```

```
                }

                num = rte_kni_tx_burst(port->kni[i], packets_bursts, rx_amount);
                kni_stats[port_id].rx_packets += num;

                rte_kni_handle_request(port->kni[i]);
                if(num < rx_amount){
                        //return the free mbufs to the kni
                        pktgen_kni_burst_free_all_mbufs(&packets_bursts[num], rx_amount - num);
                        kni_stats[port_id].rx_dropped += rx_amount - num;
                }
        }
}

static void pktgen_kni_tx_handler(struct kni_port_parameters *port){
        uint8_t i, port_id;
        uint32_t no_kni;
        unsigned no_tx, num;
        struct rte_mbuf *packets_bursts[PKT_BURST_SZ];

        if(port == NULL)
                return;

        no_kni  = port->no_kni;
        port_id = port->port_id;

        for(i = 0; i < no_kni; i++){
                num = rte_kni_rx_burst(port->kni[i], packets_bursts, PKT_BURST_SZ);
                if(num > PKT_BURST_SZ){
                        RTE_LOG(ERR, APP, "Error receiving from KNI\n");
                        return;
                }

                no_tx = rte_eth_tx_burst(port_id, 0, packets_bursts, (uint16_t)num);
                kni_stats[port_id].tx_packets += no_tx;

                if(no_tx < num) {
                        pktgen_kni_burst_free_all_mbufs(&packets_bursts[no_tx], num - no_tx);
                        kni_stats[port_id].tx_dropped += num - no_tx;
                }
        }
}
```

Bijlage D

iPerf praktijkonderzoek

TCP testresultaten

Met het *Transmission Control Protocol* zijn verschillende tests uitgevoerd.

De eerste test die uitgevoerd werd maakte gebruik van de 3Com switch. Bij deze test werd slechts één client gebruikt, namelijk Nuc-1. De server bij deze test was de andere Nuc, Nuc-2.

Bij de tweede test werd de Juniper switch gebruikt. Ook bij deze test was Nuc-1 de client en Nuc-2 de server.

Bij de derde test werden beide switches gebruikt, de data werd dus door beide switches vervoert. Zoals verwacht was de bandbreedte van deze test ongeveer gelijk aan de bandbreedte van de eerste test, omdat de 3Com switch de tragere switch was. Bij deze test was Nuc-1 weer de client en Nuc-2 de server.

Bij de vierde en vijfde test werden twee clients gebruikt om te versturen. De eerste client was Nuc-1 en de tweede was een MacBook Pro. Nuc-2 was in beide gevallen weer de server. Bij de vierde test werd de 3Com switch gebruikt en bij de vijfde test werd de Juniper switch gebruikt. Deze test is niet nog een keer uitgevoerd met beide switches, omdat de bandbreedte dan toch overeen zou komen met de bandbreedte van de vierde test. De testresultaten zijn te vinden in tabel D.1.

Tabel D.1: Resultaten test

Testnr.	Switch	Positie	Overgedragen data (Mbits)	Bandbreedte (Mbit/s)
1	3Com	Client: Nuc-1 Server: Nuc-2	904 113	94,2 94,2
2	Juniper	Client: Nuc-1 Server: Nuc-2	5856 5856	613 613
3	Beide	Client: Nuc-1 Server: Nuc-2	904 904	94,1 94,1
4	3Com	Client: Nuc-1 Client: MacBook Pro Server: Nuc-2	3136 5920 **Nuc-1:** 3136 **MacBook Pro:** 5920	32,8 62 **Nuc-1:** 32,7 **MacBook Pro:** 61,9
5	Juniper	Client: Nuc-1 Client: MacBook Pro Server: Nuc-2	2952 6056 **Nuc-1:** 2952 **MacBook Pro:** 6056	309 635 **Nuc-1: 308** **MacBook Pro: 634**

Het doel van deze tests was om de werking van iPerf te leren.

UDP testresultaten

Ook met het *User Datagram Protocol* zijn verschillende tests uitgevoerd.
De duur van de tests was 10 seconden per test. De opzet en opstelling van de verschillende tests waren hetzelfde als bij de TCP tests. In tabel D.2 is de verkregen data van de clients te vinden. De data die aan de serverkant werd gegenereerd staat in tabel D.3. In de tabel met de serverdata zijn de kolommen *Overgedragen data* en *Bandbreedte* weggelaten, omdat deze overeenkwamen met dezelfde kolommen van de tabel met de clientdata.

Tabel D.2: Resultaten test: client

Testnr.	Switch	Client	Overgedragen data (Mbits)	Bandbreedte (Mbit/s)	Datagram
1	3Com	Nuc-1	896	94,1	79999
2	Juniper	Nuc-1	7240	759	645696
3	Beide	Nuc-1	960	101	85471
4	3Com	Nuc-1	912	95,8	77551
		MacBook Pro	920	96,3	78231
5	Juniper	Nuc-1	960	101	85470
		MacBook Pro	960	101	85471

Tabel D.3: Resultaten test: server

Testnr.	Switch	Client	Jitter (ms)	Verloren	Out-of-order
1	3Com	Nuc-1	0,090	0	1
2	Juniper	Nuc-1	0,071	0	1
3	Beide	Nuc-1	0,014	4,4	1
4	3Com	Nuc-1	0,064	53	1
		MacBook Pro	14,887	48	1
5	Juniper	Nuc-1	0,107	0	1
		MacBook Pro	0,083	0,0035	1

Bijlage E

Hello World code

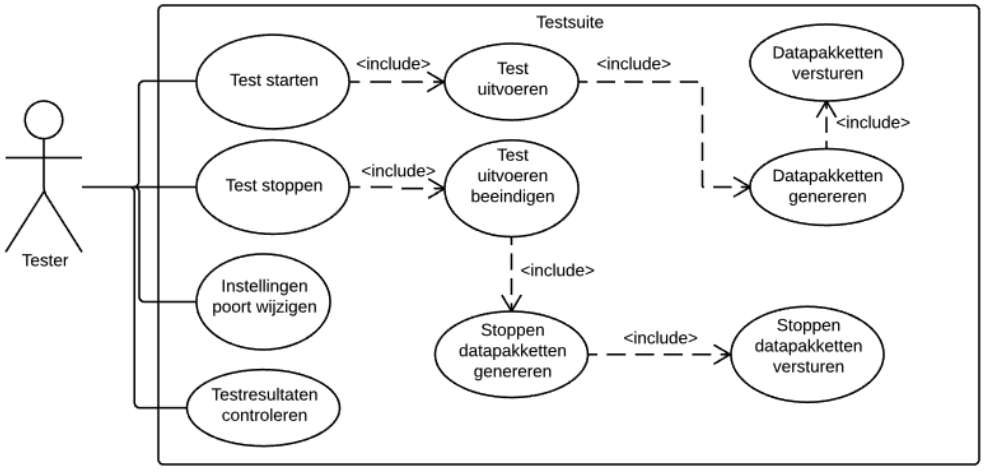
De code van het programma *Hello World* bestaat uit twee functies, namelijk: *static int lcore_hello(core_type)* en *int main*. De functie *static int lcore_hello* zoekt het identificatienummer van de aangesproken core op. Met dit nummer print hij vervolgens het bericht: ”hello from core *nummer*”. Het identificatienummer van de core wordt gevonden door de functie *rte_lcore_id()* aan te roepen. In de functie *main* wordt eerst alles geïnitialiseerd door de functie *rte_eal_init()* aan te roepen. Vervolgens wordt (met behulp van RTE_LCORE_FOREACH_SLAVE(lcore_id)) lcore_hello aangeroepen op alle slave cores. Ook wordt lcore_hello aangeroepen op de master core. Uiteindelijk wordt op de reactie van de cores gewacht door de functie *rte_eal_mp_wait_lcore()* (*Intel*, 2015b).

```
#include <stdio.h>
#include <string.h>
#include <stdint.h>
#include <errno.h>
#include <sys/queue.h>
#include <rte_memory.h>
#include <rte_memzone.h>
#include <rte_launch.h>
#include <rte_eal.h>
#include <rte_per_lcore.h>
#include <rte_lcore.h>
#include <rte_debug.h>

static int
lcore_hello(__attribute__((unused)) void *arg)
{
        unsigned lcore_id;
        lcore_id = rte_lcore_id();
        printf("hello from core %u\n", lcore_id);
        return 0;
}
int
main(int argc, char **argv)
{
        int ret;
        unsigned lcore_id;
        ret = rte_eal_init(argc, argv);
```

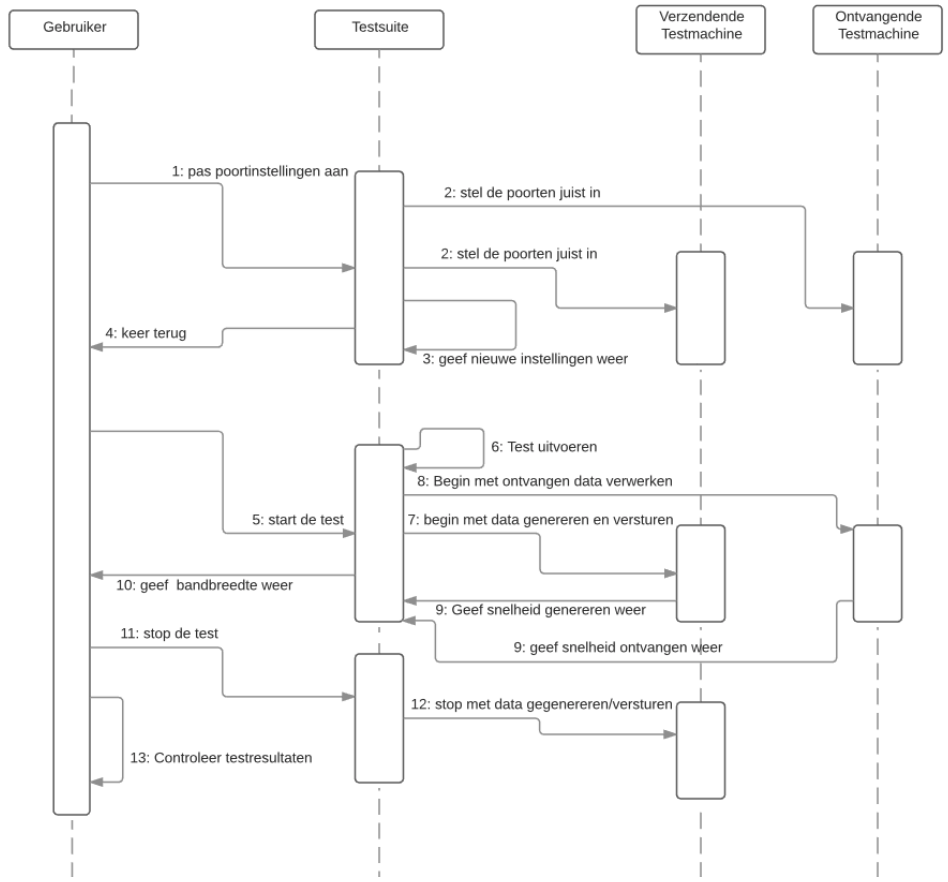
```
        if (ret < 0)
                rte_panic("Cannot init EAL\n");

        /* call lcore_hello() on every slave lcore */
        RTE_LCORE_FOREACH_SLAVE(lcore_id) {
                rte_eal_remote_launch(lcore_hello, NULL, lcore_id);
        }
        /* call it on master lcore too */
        lcore_hello(NULL);

        rte_eal_mp_wait_lcore();
        return 0;
}
```

Bijlage F

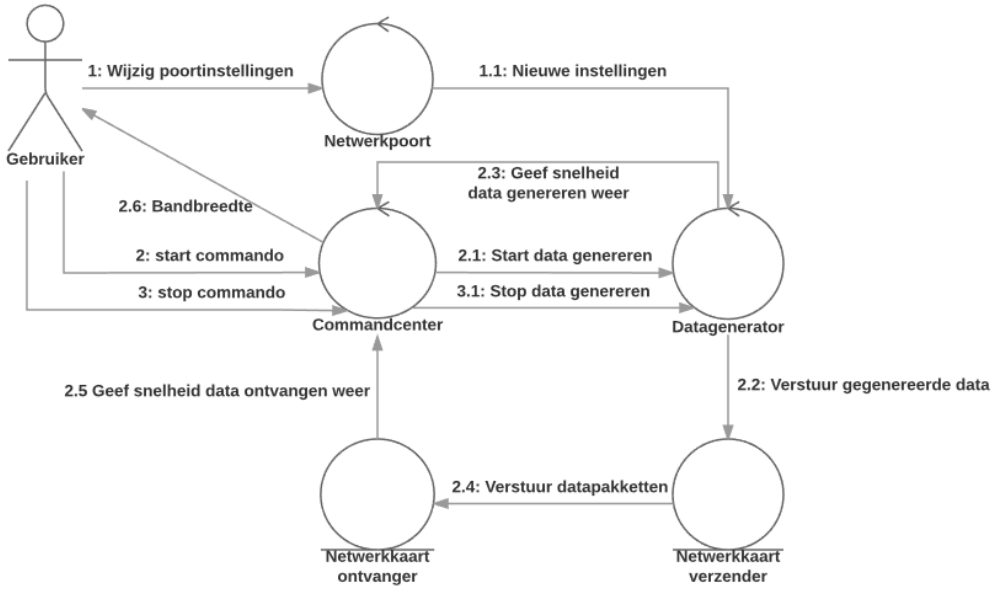
Diagrammen ontwerp



Figuur F.1: Use case diagram testsuite



Figuur F.2: Sequentie diagram testsuite



Figuur F.3: Communicatie diagram testsuite

Tabel F.1: Beschrijving use case test starten

Naam	Test starten
Doel	Deze use case zorgt ervoor dat een test gestart kan worden
Actor	Tester
Precondities	Er zijn nog geen tests gestart
Postcondities	Een draaiende test
Beschrijving	Met behulp van deze use case kan de persoon die de test uitvoert, de tester, een test starten.

Tabel F.2: Beschrijving use case test uitvoeren

Naam	Test uitvoeren
Doel	Een gestarte test uitvoeren
Actor	Geen
Precondities	Een gestarte test
Postcondities	Een draaiende test
Beschrijving	Deze use case zorgt ervoor dat de testsuite een test uitvoert. Het uitvoeren van een test geeft als uitvoer naar de gebruiker de testresultaten.

Tabel F.3: Beschrijving use case datapakketten genereren

Naam	Datapakketten genereren
Doel	Het doel van deze use case is het genereren van de datapakketten
Actor	Geen
Precondities	Een draaiende test
Postcondities	Gegenereerde datapakketten
Beschrijving	Deze use case zorgt ervoor dat de testsuite datapakketten genereert. Deze pakketten worden gebruikt om de bandbreedte te testen, en daarmee de hardware te controleren.

Tabel F.4: Datapakketten versturen

Naam	Datapakketten versturen
Doel	Het doel van deze use case is het versturen van de datapakketten
Actor	Geen
Precondities	Een draaiende test en gegenereerde datapakketten
Postcondities	Datapakketten zijn verstuurd
Beschrijving	Deze use case zorgt ervoor dat de testsuite datapakketten kan versturen. Deze pakketten worden gebruikt om de bandbreedte te testen, en daarmee de hardware te controleren.

Tabel F.5: Beschrijving use case test stoppen

Naam	Test starten
Doel	Deze use case zorgt ervoor dat een test gestopt kan worden
Actor	Tester
Precondities	Een draaiende test
Postcondities	Een gestopte test en resultaten
Beschrijving	Met behulp van deze use case kan de persoon die de test uitvoert, de tester, een test stoppen.

Tabel F.6: Beschrijving use case test uitvoeren beëindigen

Naam	Test uitvoeren beëindigen
Doel	Een draaiende test beëindigen
Actor	Geen
Precondities	Een draaiende test
Postcondities	Een gestopte test
Beschrijving	Deze use case zorgt ervoor dat de testsuite een test die uitgevoerd wordt afloopt.

Tabel F.7: Beschrijving use case datapakketten genereren stoppen

Naam	Datapakketten genereren stoppen
Doel	Het doel van deze use case is het stoppen van genereren van de datapakketten
Actor	Geen
Precondities	Een stroom datapakketten
Postcondities	Een gestopte stroom datapakketten
Beschrijving	Deze use case zorgt ervoor dat de testsuite datapakketten genereert. Deze pakketten worden gebruikt om de bandbreedte te testen, en daarmee de hardware te controleren.

Tabel F.8: Beschrijving use case datapakketten versturen stoppen

Naam	Datapakketten versturen stoppen
Doel	Het doel van deze use case is ervoor zorgen dat de netwerkkaart stopt met wachten op data
Actor	Geen
Precondities	Gestopte stroom datapakketten
Postcondities	Netwerkkaart stopt met wachten op data
Beschrijving	Deze use case zorgt ervoor dat de testsuite de netwerkkaart laat weten dat geen nieuwe datapakketten verstuurd hoeven worden. Daardoor stopt de netwerkkaart met wachten op data.

Tabel F.9: Beschrijving use case instellingen poort wijzigen

Naam	Instellingen poort wijzigen
Doel	Het doel van deze use case is om de tester in staat te stellen om de instellingen van een netwerkpoort te wijzigen
Actor	Geen
Precondities	Geen draaiende tests
Postcondities	Gewijzigde poortinstellingen
Beschrijving	Deze use case zorgt ervoor dat de gebruiker de instellingen per poort kan wijzigen

Tabel F.10: Beschrijving use case testresultaten controleren

Naam	Testresultaten controleren
Doel	Het doel van deze use case is om de testresultaten op het scherm weer te geven. Daardoor kan de tester de resultaten vergelijken met zijn verwachtingen.
Actor	Geen
Precondities	Een afgelopen test
Postcondities	Testresultaten
Beschrijving	Deze use case zorgt ervoor dat de gebruiker de testresultaten kan zien en kan vergelijken met zijn verwachtingen