Modelleren en analyseren van een Ethernet switch

Ruud Groen
r.groen@student.utwente.nl

Universiteit Twente
Informatica
Leerstoel DIES

Afstudeercommissie
ir P.G.Jansen
ir F.Hanssen
ir M.H.Wiggers

6 juni 2006

Samenvatting

De opdracht valt binnen het @Home-Anywhere-project. Het doel van dit project is thuisapparatuur in een samenhangende gedistribueerde architectuur te integreren. Het opzetten van een robuust *real-time* netwerk (RTnet) is onderdeel van dit project. In de huidige opzet van het RTnet wordt gebruik gemaakt van een hub. Echter met de toename van het aantal *Ethernet switches* in thuisnetwerken is de vraag naar een mogelijke toepassing van de *Ethernet switch* voor RTnet ontstaan. Het toepassen van een *Ethernet switch* heeft als voordeel dat meerdere stromen tegelijk kunnen zenden.

Een *Ethernet switch* biedt de mogelijkheid om Ethernet of andere soortgelijke netwerk segmenten met elkaar te verbinden tot een heterogeen netwerk.

Het *real-time* gedrag van de huidige *Ethernet switch* wordt bepaald door de gekozen architectuur en strategie waarmee een pakketje verzonden wordt. De uiteindelijke tijd die een *switch* nodig heeft om een pakketje te versturen is afhankelijk van de wachtrijen in de *switch*. Met de huidige architectuur van de *switch* is het echter niet mogelijk *hard real-time* garanties te geven over de *Ethernet switch*.

Om het gedrag van de *Ethernet switch* te voorspellen worden twee verschillende theorieën gebruikt: netwerk calculus en de standaard stochastische methode. Beide modellen schatten de lengte van de wachtrij in de *Ethernet switch* af aan de hand van de ingaande en de uitgaande stroom. De verschillen zitten in de complexiteit van de modellen. Uit het vergelijken van de resultaten van de modellen met de resultaten van de metingen blijkt, dat het goed mogelijk is een indicatie te geven van de lengte van de wachtrij en de verblijftijd van een pakketje. De *switch* kan dan ook goed gebruikt worden in het huidige RTnet. Echter de voordelen van de switch kunnen niet gebruikt worden, het gedrag van de *switch* is door het ontbreken van een exacte indicatie van de architectuur en door andere eigenschappen, die verschillen per fabrikant, niet voorspelbaar genoeg om garanties te kunnen geven over het gedrag van de verschillende stromen over verschillende *switches*.

Abstract

The assignment falls in the @Home-Anywhere-project. The goal of the @Home-Anywhere research project is to integrate home appliances into one coherent distributed architecture. Part of the project is the development of a robust real-time network (*RTnet*). In the present situation the network uses a hub. With the increased usage of Ethernet switches in home networks the question has come up if it is possible to use an Ethernet switch in RTnet. The advantage of using an Ethernet switch is the possibility to send multiple streams at the same time.

An Ethernet switch offers the possibility to connect similar network segments with each other.

The real-time behaviour of a current Ethernet switch is determined by the chosen architecture and the chosen switching strategy. The time needed for sending a frame depends on the queue length in the switch. With the current architectures it is not possible to give hard real-time guarantees.

To predict the behaviour of an Ethernet switch there are two possible theories: network calculus and stochastic models. Both models indicate the length of the queue inside an Ethernet switch by using the input and output flows. The difference is the complexity of the models. By comparing the results of the models with measurements, it appears it is possible to predict the length of the queue and the delay over an Ethernet switch. Therefore it is possible to use an Ethernet switch for the current RTnet. However, the advantages of using an Ethernet switch cannot be used. The behaviour of a switch makes it impossible to give a gurantee, due to the unknown precise architecture and other, manufacturer-dependent properties.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Hoofdstuk 1

Inleiding

De opdracht valt binnen het @Home-Anywhere-project. Het doel van dit project is thuisapparatuur in een samenhangende gedistribueerde architectuur te integreren. Een onderdeel van het project is het opzetten van een robuust *real-time* netwerk (RTnet). In de huidige opzet van het RTnet wordt gebruik gemaakt van een hub. Met de toename van het aantal *Ethernet switches* in thuisnetwerken is de vraag ontstaan naar een mogelijke toepassing van een *Ethernet switch* voor het RTnet.

Ethernet

Het Ethernet is verantwoordelijk voor het transport van pakketjes tussen twee computers. De Ethernet laag regelt twee dingen: data encapsulatie en het verzorgen van de verbinding. Ethernet is bedacht in 1970 door Metcalf [MB76] en wordt op dit moment als standaard beschreven in de IEEE 802.3 [IEE02].

The 802.3 [IEE02] standaard definieert een standaard data frame format dat gebruikt moet worden voor alle protocollen die van de MAC-laag gebruik willen maken. Een Ethernet frame moet uit de volgende onderdelen bestaan:

- Data veld (46-1500 Bytes). Het dataveld in het frame bevat de data die een bovenliggende datalaag wil versturen. De minimale grootte van het dataveld is 46 Byte. Als de hoeveelheid data kleiner is dan 46 Byte dan wordt het overgebleven deel van het veld aangevuld.
- *Destination address* (6 Bytes). Het aankomst adres is het adres waarnaar een pakketje verstuurd wordt. Binnen Ethernet zijn er 3 mogelijkheden voor het destination adres:
  - Het aankomst adres is een Ethernet adapter. het pakketje is direct verstuurd naar een andere computer. het pakketje wordt alleen door die computer geaccepteerd en de andere computers negeren het pakketje
  - Het aankomst adres is het broadcast adres. Het pakketje wordt naar alle aangesloten computers verstuurd en door alle computers geaccepteerd.
  - Het adres is een multicast adres. Bij multicast wordt een pakketje met een enkele handeling naar een groep van gebruikers verstuurd.
- Bron adres (6 Bytes). Het bron adres bevat het adres van de verzender.
- Type veld (2 Bytes). Dit veld bevat het type van het door een bovenliggende laag gebruikte protocol.

- CRC (4 Bytes). De Cyclic redundancy Check maakt het voor de ontvanger van een pakketje mogelijk om te controleren of tijdens het transport het pakketje niet is veranderd.
- *Preamble* (8 Bytes). Voordat het pakketje verzonden wordt bevat het frame een preamble. De eerste 7 Bytes van de preamble worden gebruikt als wake up van een Ethernet kaart en om de kaarten te synchroniseren. De laatste 2 bits worden gebruikt om aan te geven dat de data eraan komt.

[IEE02, KR01]

RTnet

Een robuust *real-time* netwerk is onderdeel van het @Home-Anywhere-project. Dit netwerk is zo opgezet dat het aan de volgende eigenschappen kan voldoen:

- Het moet gebruikt kunnen worden door apparaten met beperkte geheugen capaciteit. Ook kleinere onderdelen van het netwerk moeten deel kunnen nemen aan het RTnet. Het protocol is zodanig opgesteld, dat er weinig eisen aan de *hardware* gesteld worden.
- Het voldoet aan *Quality of Service (QoS) Guarantees*. QoS betekent dat het netwerk garanties af kan geven over de verbinding en de beschikbare ruimte op het netwerk.
- Het ondersteunt niet *real-time* verkeer. Naast het *real-time* verkeer moet het ook mogelijk zijn om niet *real-time* data te versturen zonder het *real-time* verkeer te hinderen.
- Het is fouttolerant.
- Het ondersteunt *plug and play*. De gebruiker hoeft geen extra handelingen te verrichten om zijn apparaat aan het netwerk te verbinden. Het netwerk detecteert zelf de nieuwe gebruiker en registreert deze.
- Het is gebaseerd op bestaande protocollen. Om het netwerk bruikbaar te maken en te kunnen invoeren in de bestaande infrastructuur moet het ook met de al bestaande protocollen kunnen werken.

Binnen het RTnet worden twee verschillende soorten netwerk verkeer onderscheiden: *real-time* en niet *real-time* netwerk verkeer. Het *real-time* netwerk verkeer heeft voorrang op het niet *real-time* netwerk verkeer. Dit houdt in dat het niet *real-time* netwerk verkeer alleen verstuurd wordt als er geen *real-time* verkeer aanwezig is. *Real-time* verkeer wordt aangeboden in de vorm van een stroom die wordt gekarakteriseerd door de bandbreedte die voor een periode nodig is [HJSM05].

Door de toepassing van een *Ethernet switch* in het RTnet kunnen er mogelijk meerdere stromen te gelijk zenden. De *switch*, in tegenstelling tot de tot nu toe gebruikte hub, zendt de pakketjes alleen naar het doeladres. Het is nu mogelijk om tegelijk met de *real-time* data ook niet *real-time* data te versturen.

Om dit mogelijk te maken moet de switch aan een aantal eigenschappen voldoen. De verschillende stromen mogen elkaar niet beïnvloeden en er mag geen pakket verlies optreden bij de *real-time* stroom. De eigenschappen van de *Ethernet switch* worden gemeten en beschreven. De uiteindelijke beschrijving van de switch gebeurt met behulp van een blackbox model, waarin de interne werking van de switch aan de hand van de verzamelde gegevens zo goed mogelijk benaderd zal worden.

Het verslag is als volgt opgebouwd:

- Hoofdstuk 2 beschrijft de werking en architectuur van de *Ethernet switch*

- Hoofdstuk 3 beschrijft de mogelijke modellen om het gedrag te beschrijven
- Hoofdstuk 4 beschrijft de verschillende experimenten
- Hoofdstuk 5 en 6 beschrijven de resultaten en geven een korte discussie en conclusie

Hoofdstuk 2

De *Ethernet switch*

Een *Ethernet switch* biedt de mogelijkheid om Ethernet of andere soorten netwerksegmenten met elkaar te verbinden tot een heterogeen netwerk.

Bij aankomst in de *Ethernet switch* worden het MAC-adres en de bronpoort van het binnenkomende pakketje opgeslagen in de adrestabel van de *switch*. Hierna zendt de *switch* het pakketje op basis van het aankomstadres en de adrestabel naar de juiste poort. Bij het opzoeken van het adres in de MAC-adrestabel zijn er drie mogelijkheden [KR01]:

- Het adres is een onbekend, een *broadcast*-, of een *multicast* adres. In dit geval stuurt de *switch* het pakket naar alle aangesloten poorten behalve naar de poort die het pakket verstuurd heeft.
- Het adres is bekend. Het pakketje wordt alleen naar de poort verstuurd die overeen komt met het bestemmingsadres in de MAC-tabel.
- Het adres is bekend en hetzelfde als het adres van herkomst. Het pakketje wordt uit de stroom weggefilterd en niet verder verzonden.

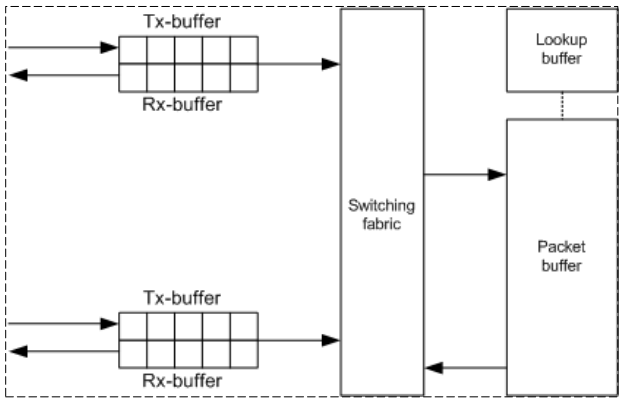
Virtuele Local Area Networks (VLAN's) kunnen gebruikt worden om de grootte van *broadcast* domeinen in *switches* te verkleinen, ze bieden tevens de mogelijkheid om enige *Quality of Service* (QoS) te bieden.

In dit hoofdstuk wordt een beschrijving gegeven van verschillende strategieën, architecturen, en diensten die nodig zijn om een pakketje door een *switch* te transporteren.

2.1 Strategieën

Voor het decoderen en routeren van een pakketje door een *Ethernet switch* zijn drie mogelijkheden, te weten *cut-through*, *store and forward* en *error free cut-through switching*.

Cut-through switching Bij *cut-through switching* bekijkt de *switch* alleen de *header* van het Ethernet pakket op zoek naar het doeladres. Nadat het adres is gedecodeerd wordt het datapad opgezet en het pakketje naar de juiste poort geleid. De correctheid van het pakketje wordt niet gecontroleerd. Omdat incorrecte pakketjes nog een keer verstuurd moeten worden, kan dit leiden tot een toename in het netwerkverkeer. Het tijdsverlies over de *Ethernet switch* wordt nu de tijd die nodig om de *header* van het pakketje met daarin het doeladres te ontvangen en de tijd die nodig is om het doeladres op te zoeken in de *hash-tabel*.



Figuur 2.1: Eenvoudig block diagram van een *Ethernet* switch

***Store and forward* switching** In een *store and forward switch* wordt eerst het hele pakketje ontvangen. De CRC van het pakketje wordt gecontroleerd en, wanneer het verzonden pakketje foutloos is, wordt het datapad goedgezet en het pakketje naar de juiste poort verzonden. Het minimale tijdverlies over de switch wordt dan de tijd die nodig is om een volledig pakketje te ontvangen. Het voordeel van de CRC controle is een afname in het netwerkverkeer. Dat komt, omdat verminkte pakketjes niet verstuurd worden naar de eindbestemming.

Error–free cut-through switching De laatste strategie die hier wordt beschreven is *error–free cut-through switching*. De *error–free cut-through switch* is een kruising tussen de *cut-through switch* en de *store and forward switch*. Het eerste pakket wordt volledig ontvangen en gecontroleerd. Wanneer de CRC van het pakket niet klopt, configureert de *switch* zichzelf als een *store and forward switch*. Het resultaat hiervan is, dat de tijd die nodig is om een pakket te versturen langer wordt, echter het netwerkverkeer wordt verminderd. Wanneer de *switch* is geconfigureerd als een *store and forward switch* en het aantal foutieve pakketjes daalt beneden een bepaalde grens, dan herconfigureert de *switch* zichzelf als een *cut-through switch*.

Wanneer het eerste pakket correct is, configureert de *switch* zich als een *cut-through switch*. Hierbij dient opgemerkt te worden dat de CRC online gecontroleerd wordt, zodat wanneer het aantal foutieve pakketten boven een bepaalde grens komt, de *switch* wordt geherconfigureerd als een *store and forward switch* [Eng].

In het geval een uitgaande poort geblokkeerd is, slaan de verschillende *switch*es allemaal de niet verzonden pakketjes op in hun buffer. Van buiten af is er dan geen onderscheid meer zichtbaar tussen de verschillende strategieën [KR01].

2.2 Architectuur

De architectuur van de *switch* bestaat uit drie delen die nodig zijn om een pakketje door de *switch* te routeren. Een eenvoudig overzicht van de architectuur is gegeven in figuur 2.1. De belangrijkste onderdelen van de *switch* zijn:

- Ingaande poort. De ingaande poort vervult verscheidene taken binnen de *switch*. Hij zorgt voor het afhandelen van de inkomende stroom en het opzoeken en het doorzenden van de pakketjes door de *switch*.
- *Switching fabric*. De *switching fabric* is het eigenlijke hart van de *switch*. De ingaande en de uitgaande poorten van de *switch* worden met behulp van de

switching fabric met elkaar verbonden. *Switching fabric* fungeert als netwerk in de *switch*.

- Uitgaande poort. De uitgaande poort slaat de pakketjes komende vanaf de *switching fabric* op en verzendt ze over het netwerk naar de bestemming.

Ingaande poort

De zoekfunctie van de ingaande poort is centraal voor de functionaliteit van de *Ethernet switch*. Tijdens het zoeken wordt besloten naar welke uitgaande poort een pakketje verzonden moet worden. Het opzoeken van een adres gebeurt door gebruik te maken van de in de *switch* aanwezige *MAC-lookup* tabel. De MAC-tabel is centraal opgeslagen in de *switch* en wordt iedere keer wanneer er een nieuw pakket bij de *switch* aankomt bijgewerkt. Van het binnenkomende pakketje wordt het MAC-adres van de zender opgeslagen [RTL03a]. Om vertragingen bij het verwerken te voorkomen heeft iedere ingaande poort een eigen kopie van de *lookup* tabel. Het doeladres wordt opgezocht in de *lookup* tabel, op een dusdanige manier dat het pakket naar het best gelijkende adres wordt verstuurd. Wanneer het adres in de *lookup* tabel aanwezig is wordt het pakket direct naar de juiste uitgaande poort verstuurd, anders wordt het pakket naar alle uitgaande poorten verstuurd behalve het bronadres [KR01].

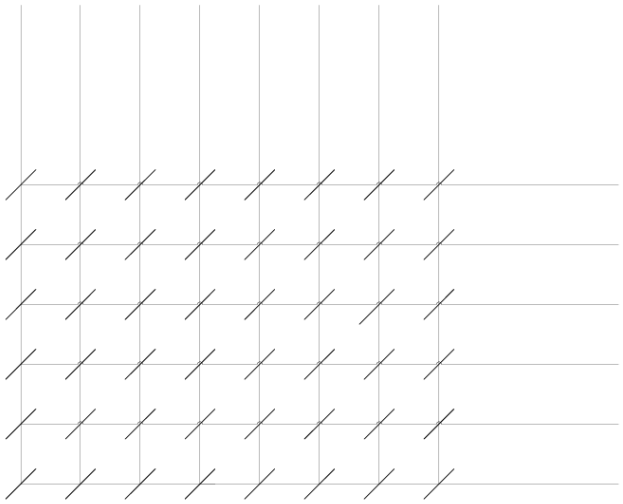
Switching fabric

Switching fabric vormt het hart van de *Ethernet switch*. De *switching fabric* verbindt de ingaande poorten met de uitgaande poorten. *Switching fabric* kan op verschillende manieren worden uitgevoerd: als een centrale bus, als centraal geheugen en als een *interconnected* netwerk.

Routeren via centraal geheugen De oudste en meest eenvoudige vorm van *switching fabric* is de *fabric* in een centraal geheugen uitvoeren. Iedere ingaande poort bevat een ontvangstbuffer voor arriverende pakketjes en een buffer met *pointers* naar vrije plaatsen in het centrale geheugen. Het binnenkomende pakketje wordt opgeslagen in de ontvangstbuffer van de ingaande poort. Nadat het pakketje volledig is ontvangen, wordt het adres opgezocht en naar de uitgaande poort verplaatst. Omdat het pakketje is opgeslagen in de centrale geheugenruimte, hoeft alleen de *pointer* die bij het pakketje hoort verplaatst te worden naar de uitgaande buffer.

Routeren via een centrale bus Wanneer een centrale bus als medium wordt gebruikt, wordt het binnenkomende pakketje direct van de ingaande naar de uitgaande poort verstuurd. Ondanks de directe verbinding tussen de ingaande en uitgaande verbinding wordt de centrale bus gedeeld door de verschillende poorten. Dit heeft als nadeel dat er slechts een enkel pakketje per keer verplaatst kan worden. Wanneer een binnenkomend pakketje de centrale bus bezet vindt, wordt het geblokkeerd en opgeslagen in de ontvangstbuffer. Aangezien ieder pakketje via de centrale bus verplaatst moet worden is de bandbreedte van de *switch* beperkt tot de bandbreedte van de centrale bus. Gezien de huidige technieken is een centrale bus geen belemmering meer om de juiste hoeveelheid bandbreedte te kunnen bieden [KR01].

Routeren via *interconnection* netwerken Om de nadelen van een enkele bus te overwinnen kan gebruik worden gemaakt van een *interconnected* netwerk. Een *interconnected* netwerk bestaat uit 2N bussen die de N ingaande poorten met de N



Figuur 2.2: *switching* via *crossbar* netwerk

uitgaande poorten verbinden. Een pakketje, dat binnenkomt op een ingaande poort, gaat via de horizontale bus aangesloten op de ingaande poort naar de verticale bus die is aangesloten op de uitgaande poort via de kruising tussen beide (fig. 2.2). Dit is alleen mogelijk als de bus naar de uitgaande poort niet bezet is. Indien de bus bezet is, wordt het pakketje opgeslagen in de buffer bij de ingaande poort [KR01].

Uitgaande poort

De uitgaande poort verstuurt de datagrammen opgeslagen in het geheugen over de uitgaande verbinding. Verder verzorgt de uitgaande poort het stopzetten van de lijn en de functionaliteit in de fysieke netwerklaag, die nodig is om te communiceren met de ingaande poort aan de andere kant van de verbinding. Het buffermanagement is nodig wanneer de pakketjes sneller bij de uitgaande poort aankomen dan de fysieke snelheid van de verbinding. De gevolgen van data opslaan bij de inkomende of de uitgaande poort worden later in dit hoofdstuk beschreven.

Wachtrijen

Wanneer naar de functionaliteit van de *switching fabric* wordt gekeken, valt op dat wachtrijen zich zowel aan de ingaande als aan de uitgaande poorten kunnen vormen. Het is belangrijk deze wachtrijen nader te bekijken, omdat wanneer de wachtrij te groot wordt, het beschikbare geheugen in de buffer te klein kan blijken te zijn waardoor pakketverlies optreedt.

De plaats waar verlies optreedt, is afhankelijk van de belasting, relatieve snelheid van het *switching* medium en de snelheid van de verbinding. Wanneer de *switching fabric* minimaal even snel is als alle ingaande poorten, dan ontstaan er geen wachtrijen bij de ingaande poorten. Zelfs in het slechtste geval is de *switch* snel genoeg om de pakketjes van de ingaande poorten naar de uitgaande poorten te verplaatsen. Nu ontstaat de mogelijkheid dat zich een wachtrij vormt bij de uitgaande poort. In het slechtste geval zenden alle ingaande poorten naar een en dezelfde uitgaande poort. In dit geval ontvangt de uitgaande poort in de tijd die nodig is om een pakketje te verzenden meerdere pakketjes. Omdat het niet mogelijk is meer dan een pakketje per tijdseenheid te verzenden, moeten de andere pakketjes worden opgeslagen in de uitgaande buffer. Uiteindelijk is het aantal pakketjes in de buffer zo groot, dat de

buffer overloopt en er verlies optreedt. Wanneer de mogelijkheid bestaat dat een wachtrij kan worden gevormd bij de uitgaande poort, wordt er gebruik gemaakt van een pakket-scheduler om te kiezen welk pakket verzonden gaat worden.

Een andere mogelijkheid is dat de *switching fabric* niet snel genoeg is. Aan de binnenkomende poort kan dan een wachtrij ontstaan. Groot nadeel hiervan is, dat een door de *switching fabric* geblokkeerd pakket aan het begin van de wachtrij, de hele wachtrij blokkeert onafhankelijk van de bestemming van de andere pakketjes [KR01]. Indien er geen *flow control* is die voorkomt dat een poort verstopt raakt, is de enige oplossing om dit probleem te voorkomen, de pakketjes die naar een verstopte poort verstuurd worden te verliezen..

2.3 Extra mogelijkheden

Naast de architectuur van de *switch* beïnvloeden scheduling en buffermanagement algoritmes ook het gedrag van de *Ethernet switch*. Op dit moment ondersteunen de meeste *switches* enige vorm van *priority queuing* en *flow control*. *Priority queuing* garandeert, dat een pakket met een hoge prioriteit eerder verstuurd wordt dan een pakket met een lagere prioriteit. *Flow control* voorkomt, dat te veel data naar een buffer verstuurd wordt en dat deze dan overloopt. De effecten hiervan zijn moeilijk te voorspellen, omdat deze geen fysieke oorzaken hebben.

Priority queuing

Om extra servicegaranties te kunnen bieden aan verschillende datastromen bieden de meeste *switches* de mogelijkheid om te schedulen aan de hand van prioriteiten. Dit wordt gedaan zoals beschreven in de IEEE 802.1p [IEE99]. Pakketjes die bij de uitgaande poort aankomen worden in twee of meer klassen verdeeld. De prioriteit die een klasse krijgt, is afhankelijk van een prioriteitsbit die gezet wordt in de *header* van een pakketje. Iedere klasse heeft zijn eigen wachtrij. De selectie welk pakketje er verstuurd wordt, hangt af van de prioriteit die een pakketje heeft. Het geselecteerde pakketje komt uit een gevulde wachtrij met de hoogste prioriteit. De volgorde in de wachtrij zelf vindt meestal op een FIFO manier plaats: het pakketje dat het eerste is aangekomen wordt het eerst geholpen. Om te voorkomen dat een enkele stroom constant aan de beurt is, worden er algoritmes gebruikt die er voor zorgen dat iedere wachtrij aan de beurt komt. Aan de hand van de prioriteit van de wachtrij wordt bepaald hoe vaak per ronde deze wachtrij geselecteerd wordt. Als een wachtrij de maximale tijd heeft verzonden, gaat de *scheduler* naar de volgende niet lege wachtrij [KR01, RTL03b]. De gebruikte methode maakt geen gebruik van *deadlines* voor de verschillende stromen en de verschillende pakketjes. De *switch* biedt op dit moment dus geen mogelijkheid om *hard real-time* garanties te kunnen geven.

Flow control mechanismen

Flow control is een kritieke factor voor de prestatie van een computernetwerk. De belangrijkste taak van *flow control* is het zo veel mogelijk voorkomen van verliezen over het netwerk. Een van de problemen van een *Ethernet switch* is de hoeveelheid geheugen die nodig is om pakketjes op te slaan zonder pakketjes te verliezen wanneer de buffers vol zitten. Het doel van *flow control* is dan ook: ervoor te zorgen dat *switches* met een beperkte hoeveelheid geheugen geen pakketjes verliezen [AOM02]. Door *Ethernet switches* worden twee protocollen ondersteund: *Collision based flow control*, dat wordt gebruikt bij half duplex verbindingen, en IEEE 802.3x *flow control* dat wordt gebruikt bij full duplex verbindingen.

Collision-Based Flow Control. *Collission-based flow control*, of *backpressure flow control* voorkomt het accepteren van pakketjes door het versturen van een stoorsignaal. Normaal wordt dit signaal gebruikt wanneer er een botsing is opgetreden, of wanneer de zender heeft geconstateerd dat de lijn niet vrij is. De zender stopt met zenden en wacht een willekeurige tijd met het opnieuw versturen van het bericht. De *Ethernet switch* verstuurt het stoorsignaal wanneer de vrije bufferruimte beneden een vooraf door de gebruiker vastgestelde waarde zakt. Het signaal wordt verstuurd naar alle poorten die vanaf dat moment beginnen met zenden om een botsing uit te lokken.

IEEE 802.3x *Flow Control*. IEEE 802.3 *flow control* vermindert het netwerkverkeer naar poorten die in *full duplex* mode staan door gebruik te maken van MAC-control-PAUSE-frames. De PAUSE-operatie geeft de MAC-laag de opdracht om de ontvangst van berichten met een multicast adres als bestemming toe te staan. Deze PAUSE-berichten zijn een subset van de zogenaamde *MAC-control-frames* en zorgen ervoor dat de ontvanger van deze pakketjes stopt met het verzenden van niet gecontroleerde berichten gedurende een bepaalde tijdsperiode. De Pauze-timer wordt geladen met de tijd uit het Pauze-bericht en wordt gestart vanaf de tijd dat het bericht is ontvangen. In het algemeen laat de IEEE standaard geen Pauze-berichten langer dan 64 Bytes toe. Langere berichten mogen vervolgens de IEEE-standaard beschouwd worden als fout [ZL504, IEE02].

Real-time gedrag

Het *real-time* gedrag van de huidige *Ethernet switch* wordt grotendeels bepaald door de architectuur en de strategie die gekozen wordt om een pakketje te verzenden. De uiteindelijke tijd die een *switch* nodig heeft om een pakketje te versturen is afhankelijk van de wachtrijen in de *switch*. Zaken als *flow control* beïnvloeden deze wachttijden en maken het moeilijk te voorspellen welk gedrag de *Ethernet switch* uiteindelijk gaat vertonen. Door de wachtrijen in de *Ethernet switch* te voorspellen moet het echter mogelijk zijn om toch enig uitsluitsel te geven over de vertraging die een *switch* toevoegt aan de totale verzendtijd. Het is dan mogelijk om te voorspellen wanneer *flow control* ingeschakeld wordt en dat moment dan te voorkomen. Door gebruik te maken van de *priority queuing* kan de wachttijd van *real-time* pakketten worden verminderd en kan de garantie dat pakketten overkomen worden gegeven. Echter, metingen zullen dit moeten uitwijzen.

Om een *Ethernet switch* zelf *soft real-time* garanties te laten geven moeten er echter wel aanpassingen aan de *switch* gedaan worden. De wachtrijen met verhoogde prioriteit dienen uitgebreid te worden en een andere *scheduling* methode kan worden gebruikt om meer garanties te kunnen geven. Om een *Ethernet switch hard real-time* garanties te laten bieden dient de *switch* de *scheduling* aan de hand van *deadlines* te doen en dient er dus een nieuwe *switch* ontworpen te worden.

Hoofdstuk 3

Model van een *Ethernet switch*

Zoals in hoofdstuk 2 is beschreven bestaat de *Ethernet switch* voornamelijk uit drie delen: de ingaande poort, de uitgaande poort en de *switching fabric*. Voor de verschillende modellen is de aanname gedaan dat de *switching fabric* is uitgevoerd als gedeeld geheugen. Het geheugen is, zoals in de *switch*, onderverdeeld in verschillende pagina's en is gelijk verdeeld over de verschillende poorten. De hoeveelheid geheugenruimte die per poort beschikbaar is bepaalt de maximale lengte van de wachtrij. Hierbij wordt aangenomen dat een ingaande en uitgaande poort voor eenzelfde stroom dezelfde geheugenruimte delen.

De beschreven modellen worden gebruikt om de voorspelbaarheid van de *Ethernet switch* te bepalen. De te voorspellen variabelen zijn de verliezen over de *Ethernet switch* en de verwachte vertraging. Het tijdverschil over de *Ethernet switch* bestaat uit drie verschillende delen [Loe03]:

- De *multiplex delay*. De *multiplex delay* (t_{mux}) is specifiek voor een *switch*. Deze waarde geeft het tijdverschil over de *Ethernet switch* zonder de invloed van de wachtrij, waarna de *switch* een ontvangen pakketje verstuurt. De vertraging wordt voornamelijk veroorzaakt door de *switching fabric* en is normaal minder dan $0,1ms$.
- Tijdverlies door wachtrijen t_{switch}. Het tijdverlies door een wachtrij is de tijd die een pakketje in een wachtrij doorbrengt plus de tijd die uiteindelijk nodig is om het pakketje te versturen. In een FIFO wachtrij is het tijdverlies afhankelijk van de lengte van de wachtrij, door de lengte van de wachtrij te beperken wordt de maximale wachttijd ook beperkt.

Tijdverlies over de *switch*. Het tijdverlies over de *switch* is de totale tijd die een pakketje in de *Ethernet switch* doorbrengt. $t_{switch} + t_{mux}$.

De multiplex vertraging is een statische waarde, die in geval van een *store and forward* strategie kan worden uitgebreid met de tijd die nodig is om een pakketje volledig te ontvangen. De t_{mux} kan dan als volgt worden berekend:

$$t_{mux} = \begin{cases} C + size_{frame} \cdot F_in & \text{, wanneer mode = } \textit{store and forward} \\ C & \text{, anders} \end{cases} \tag{3.1}$$

In vergelijking 3.1 is C een constante waarde die voor de constante tijd staat die een switch nodig heeft om het adres op te zoeken en het pakketje te forwarden. F_{in} staat voor de snelheid van de ingaande stroom in Byte/s. Om de invloed

van de extra tijd die een wachtrij aan het tijdverlies toevoegt te bepalen moet een inschatting worden gegeven van de lengte van deze wachtrij. Om de lengte van een wachtrij te bepalen zijn er verschillende technieken. In dit hoofdstuk worden de meest gebruikte technieken beschreven en met elkaar vergeleken. Alle technieken berekenen de invloed van de *Ethernet switch* op een enkele *Ethernet* stroom die door de *switch* geleid wordt. Het is met alle beschreven technieken mogelijk om meerdere stromen naar een enkele poort te simuleren. Door de aanname dat de *switching fabric* gedeeld geheugen is, wordt de invloed van de verschillende stromen op elkaar klein. Er hoeft geen bandbreedte gedeeld te worden waardoor de stromen elkaar niet hinderen. Het totale tijdverlies over de *Ethernet switch* zal dan ook overeenkomen met het tijdverlies van een enkele stroom.

3.1 Netwerk calculus

Netwerk calculus [BT01] behoort tot wat men ook wel exotische algebra's noemt. Dit is een set van mathematische systemen die inzicht geven in door de mens ontwikkelde systemen. Netwerk calculus is een theorie die deterministische wachtrijen in computernetwerken beschrijft. De Ethernet switch kan beschouwd worden als een van de wachtrijen in een computer netwerk en zal om deze reden met deze theorie beschreven worden. Het is gebruikelijk de stromen binnen deze systemen te beschrijven met de cumulatieve functie $R(t)$, de definitie is gegeven als het aantal eenheden in het tijdsinterval $[0, t]$.

In de rest van dit hoofdstuk worden de volgende aannames gedaan:

Gegeven is een systeem S; S ontvangt data die beschreven kan worden met de functie $R(t)$, en verzendt de data na een variabel tijdverlies. $R*(t)$ is de uitgaande funtie.

Om de stromen te beschrijven wordt er in de netwerk calculus de T-SPEc notatie gebruikt. Deze beschrijft de stroom als volgt: T-SPEC(C, M, r, b).

- C is de maximale snelheid van de stroom. Voor fast-Ethernet is dat bijvoorbeeld 100 Mbit/s, voor Gigabit-Ethernet is dat 1000 Mbit/s.
- M is de maximale pakket grootte. Voor Ethernet is dat zonder de CRC 1514 Byte en met de CRC 1518 Byte.
- r is de gemiddelde snelheid van de stroom.
- b is een factor voor de burstiness van de stroom. Deze factor geeft een waarde voor de burst grootte.

Aankomstcurves

Om garanties te kunnen geven over de service die een netwerk aan een datastroom kan geven, is er extra ondersteuning van het netwerk nodig. De boven grens van de binnenkomende stroom op een Ethernet switch kan wiskundig worden beschreven met behulp van een aankomst curve. De definitie van de curve is hieronder gegeven

Definitie 1 (aankomstkromme). *Gegeven een stijgende functie α die is gedefinieerd over $t \geq 0$, R is beperkt door α wanneer $s \leq t$:*

$$R(t) - R(s) \leq \alpha(t - s) \tag{3.2}$$

R heeft dan α als aankomst curve

Affine aankomstkromme. Wanneer $\alpha(t) = rt$. Deze voorwaarde betekent, over iedere interval met breedte τ, dat het aantal bits dat is verzonden wordt beperkt door $r\tau$. In dit geval wordt de stroom door de piekstroom beperkt. Dit is het geval wanneer de stroom aankomt op een link, waarbij de fysieke snelheid wordt beperkt door r (bit/s): een stroom met als enige beperking de pieksnelheid van de stroom wordt een *constante bit* of een *deterministiche bit* stroom genoemd.

In het algemeen kan een affine aankomstkromme $\gamma_{r,b}$ gebruikt worden. Deze aankomstkromme kan als volgt worden beschreven:

$$\gamma_{r,b}(t) = \begin{cases} rt + b & \text{for } t > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

De parameters b en r zijn de burst tolerantie (in bit) en de snelheid (in $\frac{b}{s}$).

Trap functies Ook een aankomst in de vorm van $kv_{T,\tau}$ kan gebruikt worden als aankomst curve. Hierin is $v_{T,\tau}$ een trap functie die gedefinieerd wordt als

$$v_{T,\tau}(t) = \begin{cases} \lceil \frac{t+\tau}{T} \rceil & \text{for } t > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

De parameters T het interval en τ de tolerantie worden uitgedrukt in tijd [BT01]. De aankomstkromme die normaal gebruikt wordt voor ATM-verkeer kan ook worden toegepast op de Ethernet switch. De aankomstkromme kan met de volgende formules beschreven worden.

$$\alpha(t) = \min(Ct + M, rt + b) \tag{3.5}$$

Deze kromme beschrijft het minimum tussen de maximale snelheid van de stroom plus de maximale pakketgrootte, en de maximale gemiddelde snelheid plus de maximale burstgrootte. De verkregen kromme beschrijft dan de maximale aankomst curve voor de Ethernet switch [Loe03].

Servicekromme

Het principe van geïntegreerde servicenetwerken is aankomstkrommen te beperken. Om middelen te kunnen reserveren moeten routers en switches in netwerken bepaalde garanties aan een stroom kunnnen geven. Deze garanties kunnen worden beschreven met behulp van een servicekromme. Om bruikbare informatie over een servicekromme te kunnen krijgen is de volgende definitie nodig:

Definitie 2 (servicekromme). *Beschouw een systeem S en een stroom die door S gaat met respectievelijk input functie en output functie R en R^*. S biedt de stroom een servicekromme β dan en alleen dan wanneer $\beta \in F$ en $R^* \geq \int_0^t R(t-s)\beta(s)ds$.*

Waarbij F de verzameling van ondergrenzen voorstelt van de functie $R(t_0 + \beta(t - t_0)$. Dit houdt in dat β een stijgende functie is waarvoor geldt dat $\beta(0) = 0$ en waarbij $t \geq 0$ geldt.

$$R^*(t) \geq \inf_{s \leq t}(R(s) + \beta(t - s)) \tag{3.6}$$

Wanneer β een continue functie is, kan het infimum vermeden worden [BT01]. Het infimum is de hoogste ondergrens van een set S, gedefinieerd als de hoeveelheid m waardoor er geen waarde in de S kleiner is dan m, zodanig dat, wanneer er een positieve waarde bij m wordt opgeteld er altijd een kleinere waarde is dan m + positieve waarde.

Propositie 1. *Als β continu is, dan wordt de service curve voor alle t, $t_0 \leq t$:*

$$R^*(t) \geq R_1(t_0) + \beta(t - t_0) \tag{3.7}$$

Waarvoor geldt dat $R_1(t_0) = \sup_{\{s<t_0\}} R(s)$ de limiet is van R t_0. Wanneer R links continu is dan geldt $R_1(t_0) = R(t_0)$.

Grenzen

In de theorie worden er drie grenzen beschreven. Deze grenzen gelden allemaal voor ideale systemen met servicegaranties. [1]

De eerste grens is de maximale backlog. Deze grens geeft de maximale waarde weer die nodig is om een stroom te kunnen bufferen in een systeem, dat een servicekromme β aanbiedt aan een stroom met aankomstkromme α. De maximale backlog is dan de maximale verticale afstand tussen α en β.

$$R(t) - R^*(t) \leq \sup_{s \geq 0} \{\alpha(s) - \beta(s)\} \tag{3.8}$$

De tweede grens die kan worden bepaald is de maximale vertraging die een systeem S aan een stroom kan geven. De maximale vertraging, die een systeem met servicekromme β aan een stroom begrensd door aankomstkromme α aan kan brengen en die de stroom in een FIFO orde behandelt, wordt gegeven door de maximale horizontale afstand tussen α en β. Voordat de definitie van de maximale vertraging kan worden gegeven moet eerst de virtuele vertraging bepaald worden [BT01].

$$\delta(s) = \inf \{\tau \geq 0 : \alpha(s) \leq \beta(s + \tau)\} \tag{3.9}$$

De grens wordt nu gegeven door het supremum van de set van alle $\delta(t)$. Het supremum is de kleinste maximale waarde van de set S waarvoor geldt dat er altijd een kleinere waarde M is die, indien er een kleine positieve waarde bij opgeteld wordt , groter wordt dan de maximale waarde. De maximale vertraging die wordt veroorzaakt door de Ethernet switch kan nu bepaald worden. Eerst dient het buigpunt g in de aankomst curve bepaald te worden.

$$g = \frac{b - M}{C - r} \tag{3.10}$$

In de formule is b een factor voor de vorm van het dataverkeer en hoe constant het dataverkeer is, M de maximale pakketgrootte, r de gemiddelde snelheid over langere tijd en C de maximale bandbreedte van de stroom. De servicekromme van de switch kan worden beschreven door de functie: $\beta = \beta_{R,t_{mux}}$. Hierin is t_{mux} de statische tijd die de switch nodig heeft om een pakketje te decoderen en naar de juiste poort te verzenden. Het maximale oponthoud over de switch wordt nu gegeven door [Loe03]:

$$t_{switch} = \frac{b}{C} - g(1 - \frac{r}{C}) + t_{mux} \tag{3.11}$$

t_{mux} heeft een theoretische waarde tussen $73\mu s$ en de 100 μs en is onafhankelijk van de pakketgrootte en de bandbreedte van de ingaande of de uitgaande stroom.

[1] Voor bewijzen van de grenzen wordt verwezen naar Network calculus [BT01]

Systemen met beperkte buffer capaciteit

Tot nog toe is er vanuit gegaan, dat de grootte van de buffer oneindig is. Deze aanname komt niet overeen met de beperkte geheugenruimte van de Ethernet switch. Aan de hand van de gegevens van het totale systeem kan nu met behulp van invoer- en uitvoervariabelen het verlies over de Ethernet switch op tijdstip t bepaald worden. Hierbij wordt er vanuit gegaan dat op $t = 0$ de buffer van de switch leeg is. Het verlies op tijdstip t kan als volgt worden weergegeven:

$$L(t) = a(t) - x(t) \tag{3.12}$$

Hierbij is $x(t)$ de stroom die daadwerkelijk door het systeem verder wordt afgehandeld, in dit geval door de switch wordt verstuurd, $L(t)$ de verliesstroom en $a(t)$ de werkelijke aankomstkromme. Door gebruik te maken van de Backlog kan deze verliesstroom bepaald worden. Indien alle beschikbare bufferruimte vol is wordt de aankomstkromme gelijk aan de servicekromme aangezien er een evenwicht in de buffer optreedt. De daadwerkelijke aankomstkromme wordt dan dus gelijk aan de service curve. Alles wat meer aankomt dan de aankomstkromme is dan de verliesstroom.

3.2 Stochastisch model

Stochastische modellen worden normaal gebruikt om wachtrijen te beschrijven en kunnen gebruikt worden wanneer het gebruikerspatroon niet exact bekend is. Het patroon is afhankelijk van veel factoren en kan niet deterministisch worden vastgesteld. Het enige dat bekend is, is het gebruik van het systeem in het verleden en de verwachtingen van de toekomst. Deze onzekerheden leiden tot het gebruik van onwillekeurige variabelen in de verschillende modellen. Deze variabelen beschrijven, op een stochastische manier, de onzekerheid over het gebruik van het systeem. Om deze variabelen te kunnen beschrijven wordt in dit hoofdstuk gebruikt gemaakt van Kendall's notatie.

Kendall's notatie

Om een enkele wachtrij te beschrijven op een niet ambigue manier wordt de zogenaamde Kendall's notatie gebruikt. Het gegeven systeem wordt beschreven met behulp van 6 variabelen, gescheiden door verticale lijnen. Dit geeft het volgende resultaat:
$(Aankomsten|Diensten|Servers|Buffergrootte|Gebruikers|Scheduling)$

- Aankomsten: de eerste factor karakteriseert het aankomst proces
- Diensten: de tweede waarde karakteriseert het uitgaande proces
- Servers: het aantal dienstverleners voor een enkele wachtrij
- Buffergrootte: de maximale grootte van de buffer
- Gebruikers: het aantal gebruikers dat gebruik maakt van de wachtrij
- Scheduling: de gebruikte scheduling methode.

De buffergrootte en de gebruikers worden vaak uit de beschrijving weggelaten. In dat geval worden deze verondersteld oneindig groot te zijn. In het geval van de Ethernet switch is de buffergrootte beperkt. Wanneer de scheduling methode niet wordt ingevuld, wordt aangenomen dat de wachtrij FCFS bediend wordt. Dit houdt in, dat het eerste pakketje dat aankomt het eerste geholpen wordt. De minimale waardes die meestal gekozen worden voor wachtrijen zijn:

- M (Markov of geheugenloos). Deze wordt gebruikt wanneer de service interval negatief exponentieel verdeeld zijn
- G (General).Deze wordt gebruikt wanneer de service interval willekeurig verdeeld is
- D (Deterministisch). Deze wordt gebruikt wanneer de tijden constant zijn
- E (Erlang verdeeld). Deze wordt gebruikt wanneer de tijden Erlang verdeeld zijn
- H (Hyper exponentieel). Deze wordt gebruikt wanneer de tijden hyper exponentieel verdeeld zijn

De verschillende wachtrijen worden later verder beschreven [Hav98].

Het (M|M|1) model

De wachtrij in de Ethernet switch kan beschreven worden met een eenvoudig $(M|M|1)$ systeem. In het $(M|M|1)$ wordt er vanuit gegaan dat de ingaande en de uitgaande wachtrijen exponentieel verdeeld zijn, en dat iedere wachtrij door een enkele server bediend wordt. De eigenschap dat een exponentiele verdeling geheugenloos is kan toegepast worden voor de Ethernet switch. De tijd tussen twee pakketjes is onafhankelijk van het voorgaande. De enkele server kan verklaard worden door het feit dat iedere stroom door een eigen poort bediend wordt.

Little's law In Little's law wordt de relatie gelegd tussen het gemiddeld aantal pakketjes in de wachtrij, het aantal aankomsten per tijdseenheid en de gemiddelde tijd die een pakketje in een wachtrij doorbrengt. Neem aan dat er bij een systeem λ pakketjes aankomen; λ is de aankomstverhouding. De pakketjes worden bediend op een FIFO manier. Nu zijn er twee mogelijkheden: de taak wordt direct verwerkt, of de taak moet wachten totdat de andere pakketjes die al in het systeem staan verwerkt worden. De gemiddelde tijd die een pakketje in het systeem doorbrengt wordt gegeven als $E[R]$, en het gemiddeld aantal pakketjes dat in het systeem aanwezig is $E[N]$ op een willekeurig tijdstip.

Een pakketje wordt gemarkeerd tijdens de observatie van de wachtrij. Wanneer het pakketje het systeem binnenkomt wordt een timestamp (t_{in}) genomen. Wanneer het pakketje het systeem verlaat, wordt weer de tijd genoteerd (t_{out}). Het verschil tussen t_{in} en t_{out} is dan gelijk aan $E[R]$. In de tijd, die nodig is om een pakketje af te handelen, komen er gemiddeld $\lambda E[R]$ pakketjes het systeem binnen. Deze waarde moet gelijk zijn aan het aantal aanwezige pakketjes in de wachtrij ($E[N]$). [2].

$$E[N] = \lambda E[R] \tag{3.13}$$

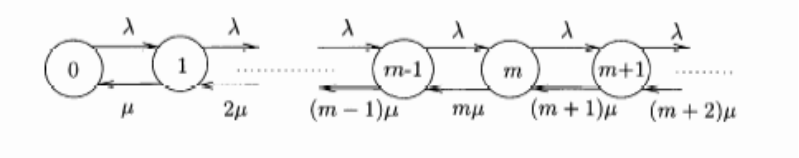
Gegeven het feit dat er verliezen kunnen optreden is λ niet altijd bekend.

Bij Little's law kunnen een aantal opmerkingen geplaatst worden:

- Er worden geen aannames gedaan met betrekking tot de distributies van de intervallen en de servicetijd
- E[N] kan eenvoudig verkregen worden

Door gebruik te maken van Little's law kunnen de karakteristieken van het simpelste wachtrijsysteem, het $(M|M|1)$ systeem worden bepaald. Gegeven is een systeem, waarbij de pakketjes negatief exponentieel verdeeld aankomen, met aankomstfrequentie λ. De service kromme van de pakketjes is ook negatief exponentieel

[2]Voor het bewijs van Little's law wordt verwezen naar [Hav98]



Figuur 3.1: Toestand transitie diagram

verdeeld met frequentie μ. Dit levert de verwachtingswaarde $E[S] = \frac{1}{\mu}$ op. Het eerste dat nu gecontroleerd kan worden is of de bezetting van het systeem kleiner is dan 1. Dit model kan gebruikt worden om de gemiddelde wachrij lengte ($E[N]$) te berekenen, de wachttijd($E[W]$) te berekenen en de reactietijd van het systeem te bepalen.

$$E[N] = \lambda E[W] \tag{3.14}$$

Hierbij wordt aangenomen dat pakketjes die volgens een Poisson proces aankomen,(een proces waarbij de intervallen tussen de pakketjes negatief exponentieel verdeeld zijn), het systeem bij aankomst als in evenwicht zien. Dit levert dan de volgende vergelijking op:

$$E[W] = E[N]E[S] \tag{3.15}$$

Door $\lambda E[S]$ door ρ te vervangen en door de termen in de vergelijking te herschikken kan de volgende vergelijking afgeleid worden:

$$E[R] = \frac{E[S]}{1-\rho} \tag{3.16}$$

Op deze vergelijking kan Little's Law toegepast worden.

$$E[R] = \lambda E[R]E[S] + E[S] = +E[R](1 - \lambda E[S]) = E[S] \tag{3.17}$$

Aangezien bekend is, dat vergelijking 3.16 geldt, kan de volgende conclusie getrokken worden:

$$E[N] = \frac{\rho^2}{1-\rho} \tag{3.18}$$

Dus wanneer de servicetijd en de aankomstfrequentie bekend zijn, kan met behulp hiervan de hoeveelheid pakketjes in de wachtrij bepaald worden.

(M|M|1) model De Ethernet switch kan worden beschouwd als een systeem waarbij de wachtrij wordt bediend door een enkele server. De tijd tussen de aankomende pakketjes is exponentieel verdeeld, hetgeen inhoudt dat $\frac{1}{\lambda_i}$ geldt als er i pakketjes in de wachtrij staan. De tijd, die nodig is om een enkel pakketje te bedienen wanneer er i pakketjes in de wachtrij staan, komt overeen met een negatieve exponentiele verdeling waarbij geldt $\frac{1}{\mu_i}$. Aangenomen dat Little's Law geldt.

Het volledige gedrag van het systeem kan nu beschreven worden met een eenvoudige Constante Tijd Markov Chain (CTMC) over de toestandsruimte $Z = (, 1, ...)$. Voor elke toestand $i \in N$ bestaat er een transitie met frequentie λ naar de staat $i+1$, die overeenkomt met de aankomst van een pakketje. Voor elke toestand $i \in N + 1$ bestaat er een transitie met frequentie μ naar de toestand $i-1$ die overeenkomt met het verzenden van een pakketje. In figuur 3.1 is een toestandsdiagram weergegeven ter verduidelijking van dit proces. De overeenkomende matrix ziet er als volgt uit:

$$Q = \begin{bmatrix} -\lambda_0 & \lambda_0 & 0 & \cdots & \cdots & \cdots & \cdots \\ \mu_1 & -(\mu_1+\lambda_1) & \lambda_1 & 0 & \cdots & \cdots & \cdots \\ 0 & \mu_2 & -(\mu_2+\lambda_2) & \lambda_2 & 0 & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{3.19}$$

Aangezien het systeem altijd een evenwicht bereikt, moet de kans dat een stroom een toestand ingaat gelijk zijn aan de kans dat een stroom de toestand verlaat, waarbij de kansstroom uit een toestand gelijk is aan de kans vermenigvuldigd met de uitgaande snelheid. Dit levert dan op:

- for state 0: $p_0\lambda_0 = p_0\mu_0$
- otherwise : $p_i(\lambda_i + \mu_i) = p_{i-1}\lambda_{i-1} + p_{i+1}\mu_{i+1}$

Waarbij geldt:

$$\sum_{i=0}^{\infty} p_i = 1 \tag{3.20}$$

De algemene oplossing van de voorgaande vergelijking wordt:

$$\sum_{i=0}^{\infty} = \sum_{i=0}^{\infty} p_0 \prod_{k=0}^{i-1} \frac{\lambda_k}{\mu_{k+1}} = 1 \tag{3.21}$$

hieruit kan dan worden afgeleid:

$$p_0 = \frac{1}{\sum_{i=0}^{\infty} \prod_{k=0}^{i-1} \frac{\lambda_k}{\mu_{k+1}}} \tag{3.22}$$

$p(0)$ is alleen positief wanneer de oneindige som een eindige limiet heeft. Als dit waar is, dan is de wachtrij in het systeem stabiel. Door gebruik te maken van Little's Law en het feit dat $X = \lambda$ kunnen de verwachte reactie tijd en de verwachte wachttijd berekend worden. Ook de kans dat het systeem meer dan k pakketjes bevat kan nu berekend worden door gebruik te maken van $B(k) = \sum_{k=0}^{\infty} p_i$.

Tot nu toe zijn er alleen systemen bekeken waarbij de wachtrij oneindig kan zijn. Echter de buffer van de Ethernet switch heeft slechts een beperkte grootte. Het $(M|M|1)$ model moet daarom worden uitgebreid, zodat het aantal pakketjes dat opgeslagen kan worden is gelimiteerd door m. Pakketjes, die aankomen op het moment dat er al m pakketjes in het systeem aanwezig zijn, worden niet meer geaccepteerd door het systeem en gaan verloren. Echter de aangeboden diensten blijven hetzelfde. De service- en aankomstsnelheden kunnen nu als volgt worden beschreven:

$$\lambda_i = \begin{cases} \lambda, & i = 0, 1, 2, 3, \ldots, m-1 \\ 0, & i = m, m+1, \ldots \end{cases} \tag{3.23}$$

$$\mu_i = \begin{cases} \mu, & i = 0, 1, 2, 3, \ldots, m \\ 0, & i = m+1, m+2, \ldots \end{cases} \tag{3.24}$$

Het toestanddiagram van het systeem wordt gegeven in figuur 3.1. Hierbij valt de stabiliteit van het systeem op; Wanneer $\lambda > \mu$ vormen de pakketjes een wachtrij totdat het aantal pakketjes groter wordt dan m. De pakketjes, die aankomen

m	ρ=0.2	ρ=0.6	ρ=0.9
0	1.0000	1 .0000	1 .0000
1	0.1667	0.3750	0.4737
2	0.0322	0.1837	0.2989
3	0.0064	0.0993	0.2120
4	0.0013	0.0562	0.1602
5	0.0003	0.0326	0.1260
6	0.0000	0.0192	0.1019
7	0.0000	0.0114	0.0840
8	0.0000	0.0068	0.0703
9	0.0000	0.0041	0.0595
10	0.0000	0.0024	0.0508

Tabel 3.1: Voorbeeld kans op verlies

wanneer de wachtrij vol is, worden genegeerd door het systeem en als verloren beschouwd. Aan de hand van deze informatie kan de volgende vergelijking worden opgesteld:

$$p_i = p_0 \prod_{k=0}^{i-1} \frac{\lambda}{\mu} = p_0 \left(\frac{\lambda}{\mu}\right)^i = p_0\rho,\ i = 1, \ldots, m \tag{3.25}$$

Waarbij $\rho = \frac{\lambda}{\mu}$. Met behulp van deze formule kunnen de kansen berekend worden op deze stabiele toestand.

$$p_i = \frac{1-\rho}{1-\rho^{m+1}}\rho^i,\ i = 0, 1, \ldots, m \tag{3.26}$$

De doorvoer van een wachtrij met verliezen is nu niet meer automatisch gelijk aan de invoer van het systeem. De doorvoer is alleen gelijk aan de invoer zolang de wachtrij niet vol is. De throughput X kan nu berekend worden met: $X = \lambda*(1-p_m)$ waarin p_m is de kans is dat een pakketje verloren gaat [Hav98].

Voorbeeld voor een $(M|M|1||5)$ **wachtrij** Dankzij het feit dat de wachtrij een beperkte lengte heeft zullen er verliezen in het systeem optreden. De kans dat een binnenkomende taak door het systeem wordt verloren, wordt door p_m uitgedrukt. In tabel 3.1 wordt de kans weergegeven als functie van de wachtrij lengte en ρ. ρ is een inputfunctie van de aankomstsnelheid en de servicesnelheid. In tabel 3.1 komt duidelijk naar voren dat de kans op verlies substantieel toeneemt voor hogere bezettingsgraden van het systeem, dit geldt zelfs wanneer de buffergrootte toeneemt.

3.3 *Real-time* gedrag

Zoals al in hoofdstuk 2 is aangegeven is het voorspellen van de wachtrij een belangrijke eigenschap om het gedrag van de *Ethernet switch* te bepalen. De in dit hoofdstuk beschreven modellen geven beide de lengte van de wachtrij als resultaat. Aan de hand van de lengte kan het tijdverlies over de *switch* bepaald worden. De modellen zijn dan ook afhankelijk van de ingaande stroom en de uitgaande stroom. De ingaande stroom is van te voren door de gebruiker bepaald. De uitgaande stroom is de minimale service stroom die de *switch* te bieden heeft.

Bij gebruik van een *switching fabric* dat door verschillende stromen gedeeld wordt is de minimale service kromme afhankelijk van de snelheid van het gedeelde medium. Is het medium langzamer dan de uitgaande verbinding dan blokkeert de

switching fabric de stroom en wordt de service kromme de snelheid van de *switching fabric*. Is de *switching fabric* sneller dan de uitgaande verbinding dan is de uitgaande verbinding het medium dat de stroom beperkt en is de service kromme afhankelijk van de uitgaande verbinding. De verschillende modellen geven inzicht in de benodigde middelen om een stroom, zonder dat er verlies optreedt, door de *Ethernet switch* te leiden.

In de netwerk calculus wordt er weinig rekening gehouden met de verschillende scheduling methodes. In het model resulteert een andere manier van schedulen echter in een andere service kromme. Het eventueel toevoegen van een scheduling strategie maakt het niet meer mogelijk twee niet gelijkwaardige stromen bij elkaar op te tellen, het resultaat zal hetzelfde blijven echter het resultaat per stroom kan veranderen. Dit is echter eenvoudig op te lossen door percentages van de service kromme aan de verschillende stromen toe te wijzen.

Beide modellen zijn geschikt om het gedrag van een *switch* te kunnen bepalen. Het netwerk calculus model geeft een makkelijke en snelle oplossing om de verliezen over de switch te bepalen en is daarom beter geschikt om te worden gebruikt voor doeleinden anders dan het modelleren van een *Ethernet switch*. Het $(M|M|1)$ model is de standaard oplossing die vaak wordt gebruikt voor het bepalen van de performance van een Ethernet netwerk. Het blijkt echter steeds vaker dat de gebruikte verdelingen niet overeen komen met de werkelijke verdeling van het Ethernet. De metingen later in het verslag moeten uitwijzen of het model geschikt is om de eigenschappen van de *Ethernet switch* te bepalen.

Hoofdstuk 4

Het experiment

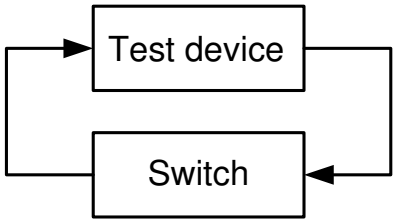
Het gedrag van een *Ethernet switch* kan worden bepaald met behulp van een aantal parameters zoals de vertraging die de *Ethernet switch* oplevert en het pakketverlies gemeten over de *Ethernet switch*. Dit zijn ook belangrijke parameters om te kunnen bepalen of de gebruikte *Ethernet switches* geschikt zijn voor *real-time* netwerken. Voor *real-time* toepassingen is het van belang dat het gedrag voorspelbaar is. Met behulp van de in hoofdstuk 3 beschreven modellen wordt een poging gedaan het gedrag van de *Ethernet switch* te voorspellen. Dit hoofdstuk beschrijft de gebruikte testopstelling en de tests die gebruikt zijn om het gedrag van de *switch* te meten. De gebruikte tests zijn standaardtests of afgeleid van de tests zoals beschreven in de RFC 1944 [BM96]. Deze is later vervangen door de RFC 2544 [BM99]. Een korte beschrijving van de gebruikte software kan worden teruggevonden in appendix A.

4.1 De testopstelling

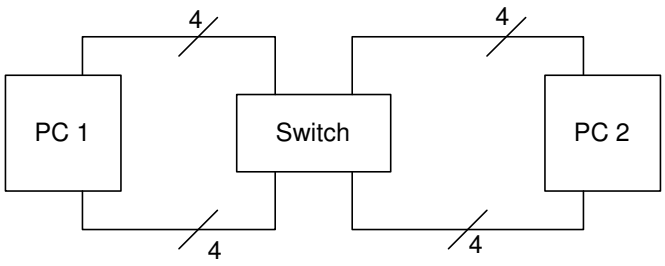
De tests kunnen het best worden uitgevoerd met een enkele testmachine die zowel ingaande als uitgaande poorten bevat. De testmachine kan dan op de *Ethernet switch* worden aangesloten, waardoor de uitgaande poorten van de testmachine via de *Ethernet switch* zijn aangesloten op de ingaande poorten van de testmachine (zie figuur 4.1).

Omdat het testapparaat zowel de data verstuurt als ontvangt, kan dit apparaat gemakkelijk de transporttijd en het pakketverlies over de *Ethernet switch* bepalen. Om het tijdverlies over de *Ethernet switch* te bepalen kunnen ook meerdere machines gebruikt worden. Deze machines gezamenlijk simuleren dan de enkelvoudige testmachine. De hoeveelheid werk om een nauwkeurige meting te kunnen doen en de complexiteit van de opstelling om de meting uit te voeren neemt echter toe, omdat de verschillende machines gesynchroniseerd moeten worden.

De gebruikte testopstellingen zijn dan ook zo gekozen opdat er met zo min mo-



Figuur 4.1: Ideale test-opstelling



Figuur 4.2: Testopstelling met 2 PC's

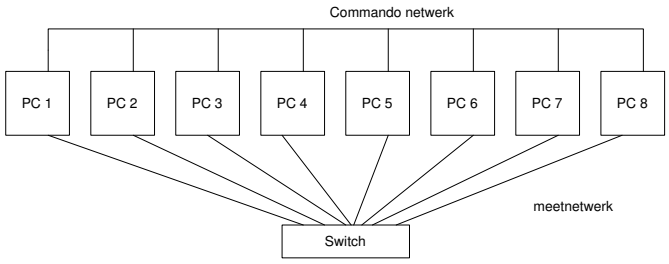
gelijk verschillende apparaten gewerkt wordt. Voor de verschillende proeven die verderop in dit hoofdstuk worden beschreven is gebruik gemaakt van verschillende testopstellingen en stonden verschillende *switches* en andere *hardware* ter beschikking. In deze paragraaf wordt een uitgebreide beschrijving van de opstelling gegeven met de benodigde informatie over de*hardware*. Voor de uitgebreide informatie over de *hardware* wordt naar de bijlage verwezen.

Delay-opstelling

In de inleiding is al vermeld, dat het gebruik van een enkele testmachine de beste manier is om het tijdverlies over de *Ethernet switch* te bepalen. Om dit zo veel mogelijk te benaderen is gekozen voor het gebruik van twee PC's met vier gelijkwaardige netwerkkaarten (figuur 4.2).

De voor deze test gekozen netwerkkaarten zijn 6 Intel 8255x Ether-Express Pro100 kaarten, een embedded netwerkkaart waarvan de chipset vergelijkbaar is met de Intel 8255x EtherExpress Pro100 kaarten en een 3COM 59x. De Intelkaarten zijn gekozen vanwege de *real-time* drivers die beschikbaar zijn.

De PC's zijn zo op de *Ethernet switch* aangesloten dat de data, die verstuurd wordt van de ene netwerkkaart naar de andere netwerkkaart, van dezelfde PC verzonden wordt. Het voordeel van het gebruik van een beperkt aantal PC's is dat er geen extra netwerk of netwerkverkeer nodig is om de PC's met elkaar synchroon te laten lopen. De gebruikte *hardware* heeft voldoende kracht om de vier netwerkkaarten apart van elkaar te kunnen aansturen. Nadeel is echter, dat onder andere de gebruikte PCI bus ook andere apparaten aanstuurt die de meting kunnen verstoren. Verder is er interactie met het geheugen nodig hetgeen enige tijd kost. Al deze factoren maken dat, alvorens er gemeten kan worden, de apparatuur eerst geijkt dient te worden om vertragingen veroorzaakt door de gebruikte opstelling uit te kunnen sluiten. Om verdere verstoringen van het meetproces uit te sluiten is er ook gebruik gemaakt van een uitgeklede Kernel. De Linux kernel is speciaal voor dit doel gecompileerd en biedt alleen de minimale ondersteuning die nodig is om de computer te kunnen gebruiken. De gebruikte Linux Kernel is de 2.4.29 Kernel. Om de verschillende *Kernel stacks* te omzeilen is er gebruik gemaakt van een zelf geschreven Kernel module die de netwerkkaarten direct aanstuurt. Hierdoor ontstaat de mogelijkheid om het moment van vertrek een nauwkeuriger *timestamp* te maken en de interactie met de netwerkkaart te vereenvoudigen. Kort voor het verzenden van het pakketje wordt door de driver van de netwerkkaart een *timestamp* gezet vlak voordat het pakketje naar de kaart verstuurd wordt. Bij aankomst wordt gebruik gemaakt van de Linux kernel *timestamp* die wordt gegenereerd op het moment dat een pakketje ontvangen is. Door een statisch *array* aan te maken kan de tijd, nodig voor het opslaan van een pakketje, worden beperkt. Van de aangekomen pakketjes worden alleen de eerste 30 *Bytes* in de Linux Kernel opgeslagen (de *header*



Figuur 4.3: Throughput opstelling met 8 PC's

en de *timestamps*). Na iedere test wordt de *array* met meetdata naar *user space* gekopieerd, waarna de data *off-line* in een bestand kan worden opgeslagen.

De opstelling is echter niet geschikt om het aantal pakketjes te tellen. Gezien de maximale belasting van de PCI-bus kan niet met zekerheid worden bepaald of pakketjes zijn verloren omdat de computer het niet aan kan of omdat de *Ethernet switch* de pakketjes verliest.

Throughtput-opstelling

De throughput is het maximaal aantal pakketjes dat zonder enig verlies over de *Ethernet switch* verstuurd kan worden. Om dit te kunnen bepalen moet de gebruikte meetopstelling zo min mogelijk andere taken uitvoeren. Omdat dit niet mogelijk is met de opstelling waarmee het tijdverlies wordt bepaald, is ervoor gekozen het zogenaamde RTnet te gebruiken.

Hardwarematig is het RTnet te beschrijven als een netwerk van 8 computers (figuur 4.3). Door hier de *Ethernet switch* in te hangen en een *socket* programmaatje te gebruiken dat alleen maar pakketjes zendt en ontvangt, kan het aantal verzonden en ontvangen pakketjes nauwkeurig worden bepaald. Groot voordeel van het gebruik van de RTnet-opstelling is, dat het netwerk is voorzien van een extra coax-netwerk (figuur 4.3) waarover de commando's aan de verschillende computers gegeven kunnen worden. De meetdata worden niet beïnvloed door eventuele extra pakketjes die nodig zijn voor de commando's om gelijk te starten en gelijk te stoppen. De netwerkkaarten in de computer worden voor dit doel in *niet-promiscous* mode gezet, zodat alleen de pakketjes bestemd voor deze computer ontvangen en geteld worden. Eventuele andere pakketjes worden dan door de Linux Kernel verwijderd.

Gebruikte *switches*

Voor de verschillende proeven zijn 3 verschillende *switches* gebruikt waarin twee verschillende chipsets zitten. Er zijn twee E-tech 8-poort *switches* gebruikt met een Realtek 8309 chipset en een Belkin 5-poort *switch* met een Micrel KS8995X chipset. Er zijn twee E-tech *switches* gebruikt om de resultaten goed met elkaar te kunnen vergelijken. Deze paragraaf geeft een korte beschrijving van de verschillende *switches* en de specifieke eigenschappen van de *switches*. Voor de exacte verschillen wordt naar de data-sheets van de fabrikanten van chipsets verwezen ([RTL03b, KS804]).

E-tech 8-poort *switch* De E-tech 8-poort *Ethernet switch* is een standaard *switch*, die ook gebruikt wordt in de meeste thuisnetwerken. De *switch* heeft 8 utp poorten en maakt gebruik van *autonegotiation* om de snelheid vast te stellen. De *switch* heeft een Realtek 8309SB [RTL03b] chipset met 256 kByte geheugen en

een MAC-adres tabel van 4 kByte. De *switch* maakt gebruik van een *store and forward strategy* zoals beschreven in hoofdstuk 2 en de *routing* is mogelijk *memory swiched*. Zoals de meeste *switches* op dit moment ondersteunt de *Ethernet switch* *Full-duplex flow control* en in geval van *half duplex backpressure flow control*. Het geheugen in de *switch* maakt gebruik van een *paging* strategie. Het maximaal aantal pakketjes (als een pakketje kleiner is dan de pagina grootte) wordt bepaald door het maximaal aantal aanwezige pagina's en de paginagrootte van de *Ethernet switch*.

De gebruikte chipset biedt de mogelijkheid voor *priority queuing*. De indeling van het geheugen is statisch verdeeld tussen de poorten: de uit de datasheet verkregen informatie meldt dat de rest van de bus structuren in de *switch* voldoende zouden moeten zijn om 8 poorten op volle snelheid te kunnen ondersteunen.

Belkin 5-poort switch De Belkin *Ethernet switch* is een ouder model dan de E-tech *Ethernet switch*. Hierdoor bestaat de mogelijkheid dat deze *switch* andere eigenschappen bezit dan de E-tech. De Belkin 5-poort *Ethernet switch* is uitgerust met een Micrel chipset. De *switch* is uitgerust met 5 poorten of 4 poorten en een enkele uplink poort. De Micrel chipset is uitgerust met een *switching fabric* gebaseerd op een gedeelde geheugenoplossing. De datasheet geeft aan dat de *switching fabric* niet blokkerend is. Het geheugen heeft een bandbreedte van 1,4 GHz, een 1 kB MAC-adres *lookup table* en is 64 kB groot. De *switch* ondersteunt het *IEEE priority protocol*. Ook bij deze *switch* is die uitgevoerd met behulp van twee verschillende *priority stacks*. De *switching fabric* behandelt de verschillende inkomende stromen volgens een FIFO principe. De *switch* ondersteund als extra service ook het IEEE *flow control* protocol in *full duplex* mode en in geval van een *half duplex* verbinding wordt *flow control* geleverd in de vorm van *backpressure* [KS804].

4.2 De uitgevoerde experimenten

Om het *real- time* gedrag van de *switch* te bepalen is een tweetal experimenten uitgevoerd: een delay meting en een throughput meting. Voor de metingen is gebruik gemaakt van standaard pakketgroottes met een standaard *Ethernet header*. Het te meten gedrag van een *Ethernet switch* bestaat uit het bepalen van het tijdverlies en het pakketverlies gemeten over de *Ethernet switch*. Voor *real-time* applicaties is het belangrijk te weten hoe lang een proces duurt. En voor een *real-time* stroom is het van belang te weten of alle data over is gekomen binnen de voor het proces gereserveerde tijdsduur. De karakteristieke eigenschappen van de *Ethernet switch* beschrijven deze waarden en kunnen dus als *real-time* gedrag beschouwd worden.

Voor de verschillende tests wordt uitgegaan van een constante belasting van de *Ethernet switch* gedurende de te meten tijd. In werkelijk netwerkverkeer is de belasting over het algemeen lager en zeker niet constant. De gekozen scenario's geven echter wel de situatie in het slechtste geval en geven inzicht in het feit of de *switch* het onder deze omstandigheden volhoudt. De korte tijden, waarop normaal netwerkverkeer verloopt, worden gesimuleerd door de tijd die een test gaat duren te verkorten.

Om dit gedrag te bepalen is er voor alle opstellingen en uitgevoerde experimenten van uit gegaan dat de bestemming van het bericht bij de *Ethernet switch* bekend is, aangezien dit gegeven duidelijk de prestaties van de *Ethernet switch* beïnvloedt.

Pakketgroottes

Voor de verschillende tests worden er pakketgroottes gebruikt, die in normaal *Ethernet* verkeer vaak voorkomen. Bij deze pakketgroottes hoort in ieder geval het klein-

grootte (bytes)	*Ethernet* (pps)
64	14880
128	8445
256	4528
512	2349
768	1586
1024	1197
1280	961
1518	812

Tabel 4.1: Aantal pakketten per pakket grootte voor 10 Mbit

ste pakketformaat 64 Byte en het grootste 1518 Byte [BM99]. De gebruikte pakketgroottes staan in tabel 4.1.

Het in tabel 4.1 genoemde aantal pakketjes is gegeven voor een 10 Mbit stroom. Voor de 100 Mbit stroom is aangenomen dat het aantal tien keer zo hoog ligt. De meest voorkomende pakketgrootte bij dataverkeer is 1518 Byte, dit omdat deze pakketgrootte de hoogste lijnefficiency biedt.

De gebruikte opstellingen negeren alle pakketjes die tijdens de test binnenkomen en niet door de apparatuur zelf zijn gegenereerd. Dit kunnen bijvoorbeeld netwerkcontrole pakketjes zijn zoals *keep-alive* pakketjes en *routing-update* pakketjes. Zoals al eerder is beschreven, wordt een voor dit doel gecompileerde Linux kernel gebruikt zonder de normale netwerkondersteuning. Dit heeft in dit geval als voordeel dat alle software-matig afgehandelde controle berichten niet verstuurd worden. De gebruikte module genereert verder alleen pakketjes met een verder niet gebruikt netwerk protocol. Andere pakketjes worden niet door de module geaccepteerd.

Bij de throughput-test worden gewoon alle pakketjes geaccepteerd, omdat te veel handelingen de test verstoren. Door de netwerkkaarten niet in niet-*promiscous mode* te zetten worden in ieder geval alleen die pakketten geaccepteerd die voor de specifieke computer nodig zijn. Tevens wordt er bij deze test een specifiek protocol meegegeven dat alleen door de testmachines wordt gebruikt.

Bepaling invloed meetapparatuur

Eerder is al aangegeven dat er een niet ideale meetopstelling gebruikt. Om dit te compenseren moet de afwijking die de meetopstelling zelf toevoegt worden bepaald. De invloed kan per scenario verschillen. De gebruikte scenario's bij de delay-meting zijn zo opgesteld, dat deze ook gebruikt kunnen worden voor het ijken van de meetapparatuur. Voor het ijken worden twee *cross-link* kabels tussen de netwerkkaarten van een van de twee testmachines geplaatst. Over deze netwerkkabels worden tussen de twee netwerkkaarten de verschillende stromen verstuurd. Aangezien dit *full-duplex* kan, bestaat de mogelijkheid om dezelfde scenario's uit te voeren als gekozen voor de delay-meting. Door het tijdverlies over deze kabels te meten kan de invloed van de Linux Kernel op een enkele stroom worden bepaald. Door meerdere stromen te versturen wordt de invloed van het gebruik van meerdere netwerkkaarten in een enkele computer bepaald. Omdat er altijd van de ene netwerkkaart naar een andere gestuurd wordt, zijn er altijd minimaal twee netwerkkaarten in gebruik die elkaar beïnvloeden.

De throughput-meting

De throughput-meting wordt gebruikt om, zonder pakketverlies, het maximaal aantal pakketten over de *switch* te versturen. Iedere test wordt gedaan door gedurende een vooraf vastgestelde tijd met de maximum snelheid een stroom tussen twee verschillende computers te versturen. Het aantal pakketten dat wordt verzonden, wordt nauwkeurig geteld door de zender, tevens wordt het aantal pakketten dat is ontvangen, geteld door de ontvangende computer. Wanneer het aantal ontvangen pakketten kleiner is dan het aantal verzonden pakketten wordt van een van de zendende computers de tijd, die de betreffende computer aan het zenden is, verkleind. De grootste hoeveelheid pakketten die verzonden kan worden, is de maximale throughput van de betreffende *Ethernet switch*. De resultaten zijn later verwerkt in een grafiek waarbij op de X-as de grootte van het pakket in Bytes wordt weergegeven en langs de Y-as het maximum aantal pakketten dat verzonden kan worden. Aan de hand van deze waarden kunnen de theoretische throughput van de *switch* en de maximale belasting van de *Ethernet switch* worden bepaald.

Tijdverlies

Het tijdverlies over een *Ethernet switch* is opgedeeld in drie verschillende delen, de tijd die nodig is voor de *switch* om het adres op te zoeken, de tijd dat een pakketje in een wachtrij staat en de tijd die nodig is om een pakket te verzenden. Om dit tijdverlies goed te kunnen meten wordt voor het verzenden een *timestamp* aan het pakket toegevoegd. Bij aankomst wordt de Kernel-*timestamp* gebruikt. Deze wordt gegenereerd na de eerste *interrupt* dat een pakketje is aangekomen. Door de waardes van elkaar af te trekken kan de tijd worden bepaald die nodig is om een pakketje via de *switch* te versturen. Omdat de beide gebruikte *switches* gebruik maken van een *store en forward* strategie, een pakketje wordt eerst volledig ontvangen en daarna pas doorgestuurd, kan ook de tijd om een bericht te ontvangen van de totale tijd worden afgetrokken. Als laatste wordt de meetafwijking van de waarde afgetrokken. De overgebleven waarde is nu de tijd die het pakketje in de *switch* heeft doorgebracht.

Om ook de maximale afwijking te kunnen bepalen is er gebruik gemaakt van een scenario waarbij twee verschillende stromen naar een ontvanger gestuurd worden. Omdat de ontvanger deze stromen nooit volledig kan afhandelen treedt er een *buffer-overflow* op. De maximale tijd die een bericht dan nodig heeft om via de *switch* bij de ontvanger te komen, wordt bij de daaropvolgende experimenten beschouwd als de maximale vertraging. Deze scenario's worden bij de throughput-meting herhaald om te bekijken of daarbij verlies optreedt en of de *flow control* in werking treedt.

De resultaten van dit experiment worden grafisch weergegeven met langs de x-as het moment dat het bericht is aangekomen en langs de y-as het tijdverschil. Ook wordt er een plot gemaakt met het gewogen gemiddelde ten opzichte van de pakketgrootte. Om eventuele onnauwkeurigheden uit te kunnen sluiten wordt er tevens een histogram van de meetdata weergegeven, opdat de onwaarschijnlijke waardes uit de gegevens gefilterd worden en dus de laatste invloeden van het meetsysteem geëlimineerd worden.

4.3 *Real-time* gedrag

Zoals al eerder is aangegeven, is het vrij moeilijk te bepalen wat *real-time* gedrag van een *Ethernet switch* is. De exacte architectuur van de *switch* is onbekend en daar kunnen geen uitspraken over gedaan worden. Ook de *flow control* die de meeste *switches* op dit moment hebben beïnvloedt de bruikbaarheid van de *Ethernet switches* voor hard *real-time* verkeer. *Flow control* beïnvloedt de stroom, zodat de

vooraf bepaalde maximale waardes niet meer gehaald kunnen worden. De pakketjes worden mogelijk door de zendende computer in een wachtrij geplaatst waardoor het tijdverlies onafhankelijk van de *switch* toeneemt. Deze invloed van de *flow control* op de stroom maakt het onmogelijk het gedrag van de *switch* te voorspellen.

Er is voor gekozen om de throughput van de *Ethernet switch* en het tijdverlies over de *Ethernet switch* te bepalen, omdat het gemakkelijk te meten is en omdat het toch belangrijk is te kijken of deze waardes voorspelbaar zijn. De mogelijkheid van prioriteit wachtrijen op de *Ethernet switch* is een belangrijk pluspunt dat zeker aandacht verdient.

Hoofdstuk 5

Resultaten

Dit hoofdstuk beschrijft de resultaten van de verschillende modellen en de uitgevoerde experimenten. Als eerste worden de resultaten van de delay-meting per *Ethernet switch* gegeven. Daarna worden de resultaten van de throughput-meting per *Ethernet switch* besproken. Het laatste deel geeft de resultaten van de verschillende modellen en als conclusie een vergelijking tussen de uitkomsten van de modellen en die van de metingen.

5.1 Metingen

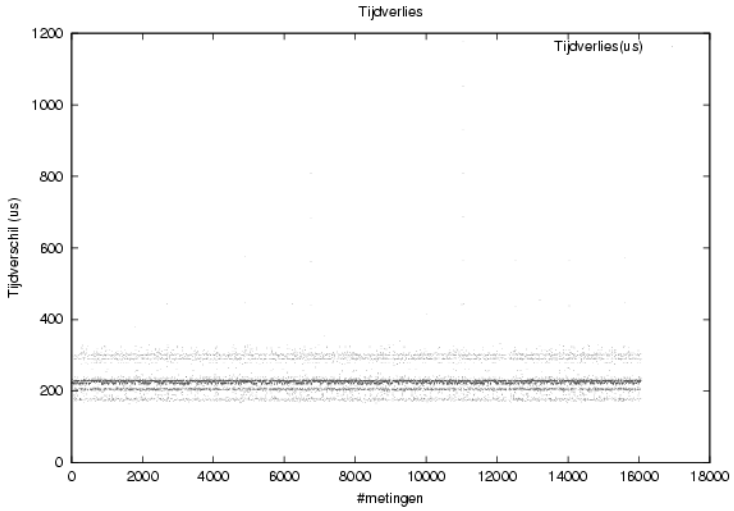
De uitgevoerde experimenten zijn gedaan zoals beschreven in hoofdstuk 4. Voordat de experimenten zijn uitgevoerd is eerst de testapparatuur geijkt. De delay-metingen worden opgedeeld in drie aparte delen, eerst worden de resultaten van de verschillende testmachines over de *cross-link* kabels beschreven, daarna wordt per *Ethernet switch* hetzelfde gedaan.

IJken van de meetapparatuur

Om exact te kunnen bepalen wat de afwijking van de *Ethernet switch* is, moet eerst de invloed van de testmachines op de meting bekend zijn. Om er zeker van te zijn dat deze afwijking niet verandert, zijn de test scenario's zo opgesteld dat ze zowel over een *Ethernet switch* als over een *cross-link* kabel uitgevoerd kunnen worden. De vertragingen in de testmachines worden veroorzaakt door de Kernel-*stack*, *interrupts* en andere niet beïnvloedbare zaken.

Als eerste wordt de vertraging voor een enkele Ethernet stroom bepaald. In figuur 5.1 is de situatie weergegeven voor een enkele stroom met pakketjes van 1514 Byte.

Wanneer het gedrag ideaal zou zijn, zou het verzenden en ontvangen van een pakketje $121\mu s$ duren. Zoals in figuur 5.1 te zien is, ligt het gemiddelde tijdsverschil rond de $200\mu s$ met uitschieters naar $1000\mu s$. Dit betekent dat de testmachine ongeveer $60\mu s$ aan de totale delay toevoegt. Deze vertraging kan worden veroorzaakt doordat de beide netwerk kaarten dezelfde PCI-bus gebruiken, buffers in de Kernel, of *interrupts* van andere systeem processen. Nu de invloed van het systeem op een meting met een enkele stroom bekend is, wordt de invloed van meerdere netwerkkaarten op de meting bekeken. De PCI bus zou in theorie snel genoeg moeten zijn om netwerkverkeer van vier verschillende netwerkkaarten af te kunnen handelen. Echter hierbij is enige *overhead* aanwezig, maar er is geen rekening gehouden met eventuele andere processen die dezelfde bus gebruiken. Om met het vorige figuur

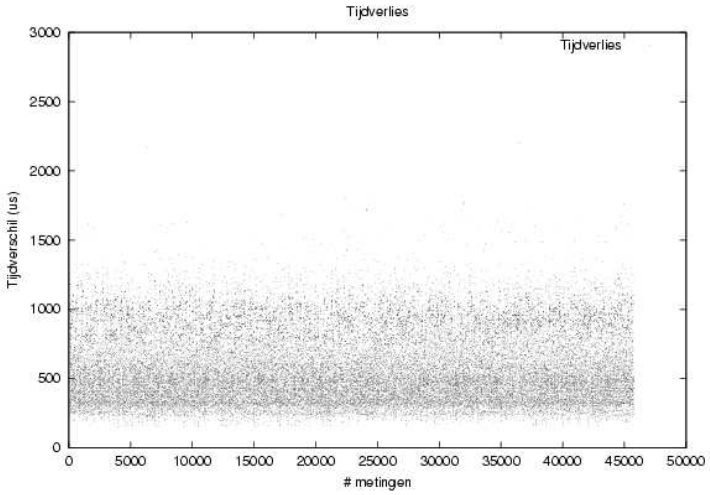


Figuur 5.1: Enkele stroom van pakketten met grootte 1514 Byte verzonden over een *cross-link* kabel

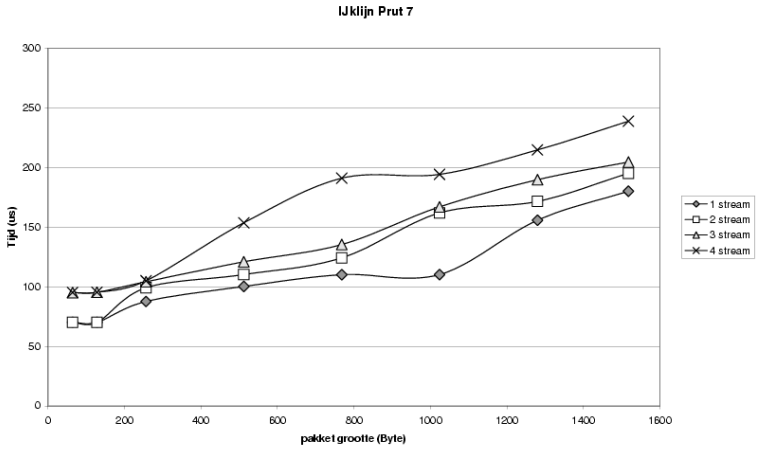
te kunnen vergelijken is gekozen voor een test uit dezelfde serie met vier stromen met pakketjes van 1514 Byte.

In figuur 5.2 is duidelijk de invloed van de extra stromen op de meting te zien. Het gewogen gemiddelde van de meting wordt nu 239μs. Het aantal pakketjes met een maximale overhead is nu groter geworden. Verder valt de grotere spreiding van meetpunten op. Dit gedrag treedt op bij alle metingen en alle pakketgroottes. Aan de hand van deze gegevens kan een ijklijn voor de verschillende pakketgroottes en het aantal stromen over de apparatuur worden opgesteld (figuur 5.3). Door de grote verschillen in de meetresultaten kan nu niet meer de garantie worden gegeven, dat de testcomputer alle pakketjes die verstuurd worden ontvangt. Dat maakt deze testopstelling verder ongeschikt om throughput-metingen mee te doen.

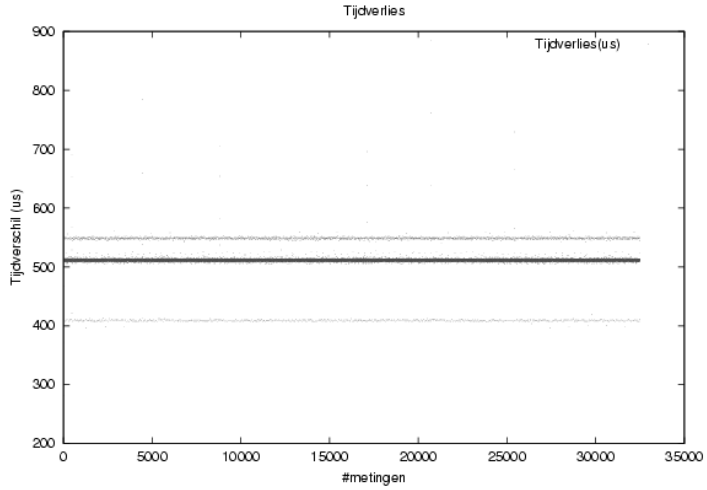
Aan de ijklijn is duidelijk de invloed van het aantal stromen op de meetapparatuur waar te nemen. Alle vooraf gepresenteerde resultaten zijn afkomstig uit een meting op PC7 een van de testcomputers. De resultaten van de andere testcomputer zijn vergelijkbaar. Bij het maken van de figuren is een filter gebruikt om eventuele negatieve resultaten niet in de grafiek weer te geven. De negatieve resultaten worden veroorzaakt door verstoringen bij het opslaan van de meetgegevens en worden als zodanig ook niet opgenomen in het gewogen gemiddelde. De negatieve metingen treden alleen op bij een extreme netwerkbelasting zoals het versturen van alleen maar 64 Byte pakketjes. In figuur 5.3 valt op dat voor de kleinere pakketgroottes de duur niet verschilt ten opzichte van het verschillend aantal stromen. Tevens valt op dat de tijd die een pakketje nodig heeft voor 64 Byte gelijk is aan de tijd die een pakketje van 128 Byte nodig heeft om verzonden te worden. Dit kan worden verklaard vanwege het feit dat de computer waarschijnlijk in deze opstelling geen kleiner tijdsverschil kan waarnemen. Een andere waarnemening die aan de hand van figuur 5.3 gedaan kan worden is dat het tijdsverschil ten opzichte van de pakketgrootte vrijwel lineair is. In de situatie dat er geen extra tijd door de Kernel wordt toegevoegd, klopt dit vanwege het feit dat de tijd die nodig is, afhankelijk is van de pakketgrootte. Blijkbaar is de extra tijd die de Kernel toevoegt constant en deels afhankelijk van de pakketgrootte. De toename in tijd door meerdere stromen te gaan versturen kan verklaard worden vanwege het feit dat er slechts een enkele PCI-bus in de meetsystemen zit, waardoor met het aantal apparaten dat pakketjes



Figuur 5.2: 4 stromen met een pakket grootte van 1514 Byte verzonden over enkele *cross-link* kabels



Figuur 5.3: gewogen gemiddelde per pakketgrootte per aantal stromen



Figuur 5.4: enkele stroom met pakket grootte van 1514 Byte verzonden via een Belkin *Ethernet switch*

verstuurt de delay over deze enkele PCI-bus ook toeneemt.

5-poort *Ethernet switch*

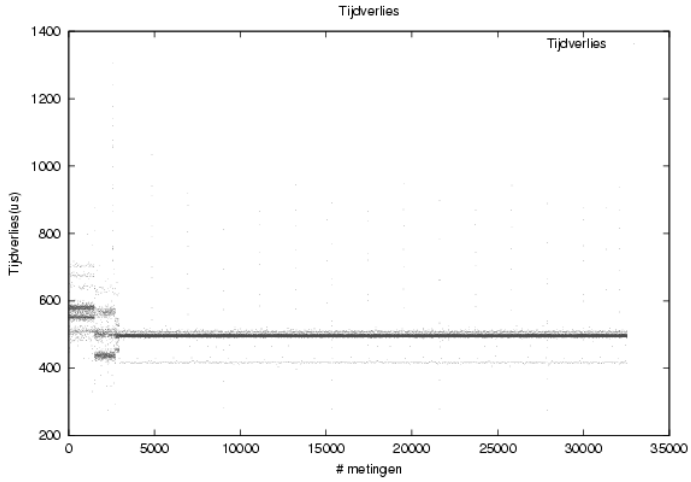
Nu het gedrag van de testcomputers bekend is, kan worden begonnen met het bepalen van het gedrag van de *switches*. Door de gekozen opstelling kunnen er echter maar vier van de vijf poorten van de 5-poort *switch* belast worden. Dit zal echter weinig invloed hebben op het gedrag van de *Ethernet switch*. Zoals beschreven wordt dezelfde serie tests uitgevoerd op de *Ethernet switch*. Het resultaat wordt gegeven in figuur 5.4.

In figuur 5.4 is duidelijk te zien, dat het pakketje er in deze test langer over doet dan over de crosslink kabel. Het gemiddelde ligt nu ongeveer bij de $510\mu s$. Opvallend is, dat wanneer figuur 5.5 met 4 stromen bekeken wordt deze een vergelijkbare vertraging heeft gemeten over de *Ethernet switch*. Hierbij ligt de gemiddelde vertraging rond de $520\mu s$. Dit geeft aan dat de invloed van de stromen op elkaar in de *switch* minimaal is.

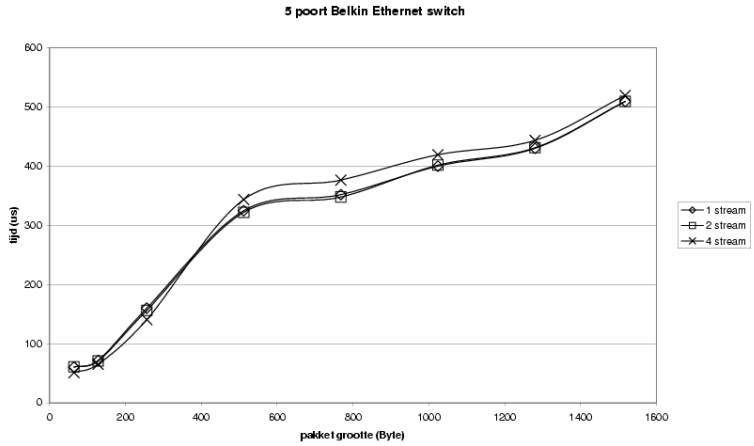
Tevens valt op dat, wanneer de grafiek met gemiddeldes (figuur 5.6) vergeleken wordt met de ijklijn dat de gemiddeldes een stuk minder lineair zijn. Bij pakketjes groter dan 512 Byte neemt de richtingcoëfficiënt duidelijk af ten opzichte van het lijndeel daarvoor. Voor de kleinere pakketjes (64B en 128B) blijft gelden, dat de gemeten tijden meer afhangen van de gebruikte meetopstelling dan van de tijd die een pakketje nodig heeft om verstuurd te worden. In figuur 5.6 zijn de resultaten met 3 stromen uit de grafiek weggelaten vanwege vreemde verstoringen in de meting. Deze verstoring is ook al opgetreden bij het ijken van de apparatuur en is dus waarschijnlijk niet veroorzaakt door de *Ethernet switch* en om die reden dan ook niet in het figuur opgenomen.

Tot nu toe zijn alleen resultaten van metingen waarbij alle geadresseerden bekend zijn weergegeven. Wanneer dit niet het geval is gaat de *Ethernet switch broadcasten* naar alle adressen. Een *broadcast* wordt weliswaar met minder prioriteit behandeld, maar verstoort toch het evenwicht over de *Ethernet switch*.

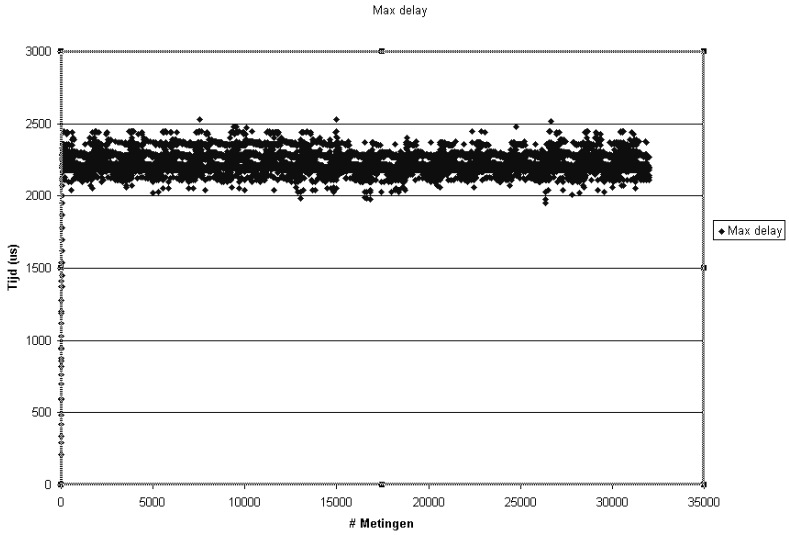
De vertraging bereikt een maximum, waarna het *back pressure* protocol in werking treedt. De wachttijd over de *Ethernet switch* neemt dan tijdelijk af, waarna



Figuur 5.5: Test 4 stromen met pakket grootte van 1514 Byte verzonden via een Belkin *Ethernet switch*



Figuur 5.6: gewogen gemiddelde per pakketgrootte per aantal stromen



Figuur 5.7: 4 stromen met pakketgrootte 1024 B over een *Ethernet switch*

deze naar hetzelfde maximum toegroeit. De maximale *delay* over de *Ethernet switch* is afhankelijk van de buffergrootte en de paginagrootte. In figuur 5.7 is de *delay* over de *Ethernet switch* gegeven met een pakketgrootte van 1024 B en gemeten door PC1. De throughput-meting, die later in dit hoofdstuk beschreven wordt, zal bepalen of in het geval van maximale *delay* pakketverlies optreedt of dat er minder pakketten verzonden worden. Hetzelfde verschijnsel treedt op wanneer er met twee ingaande stromen naar een uitgaande stroom gezonden wordt. Dit zal in de volgende paragraaf beschreven worden.

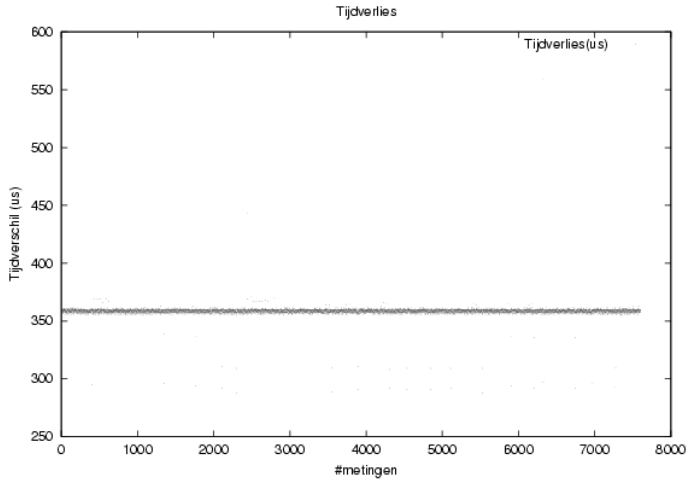
8-poort *Ethernet switch*

De metingen die verricht zijn over de 8-poort *Ethernet switch* kunnen onderverdeeld worden in twee verschillende delen; de metingen die zijn gedaan met een enkele testcomputer en de metingen die zijn uitgevoerd met de beide testmachines tegelijkertijd. De enkele tests bevatten dezelfde tests zoals uitgevoerd met de 5-poort Belkin *Ethernet switch*. De met beide computers uitgevoerde tests overlappen voor een deel de enkelvoudige test, maar zijn gedaan om een volledige belasting van de *Ethernet switch* te kunnen generen.

8-poort *Ethernet switch* met een enkele testmachine

Eerst worden metingen uitgevoerd over de 8-poort *Ethernet switch* met een enkele machine tot maximaal vier verschillende stromen over de *Ethernet switch*. Dit levert vergelijkbare resultaten op met die van de 5-poort *Ethernet switch*. Beide *switches* werken zoals eerder is beschreven in hoofdstuk 4 volgens hetzelfde principe. De eerste grafiek (figuur 5.8 geeft de situatie weer voor een enkele stroom met een pakketgrootte van 1518 B.

Bij deze eerste meting ligt het gemiddelde zo rond de $370\mu s$. De metingen van de half belaste 8-poort *Ethernet switch* volgen meer de structuur van de ijklijn. De algemene grafiek met gemiddeldes wordt gegeven bij de resultaten van de volledig belaste *switch*.



Figuur 5.8: Enkele stroom over 8-poort *Ethernet switch*

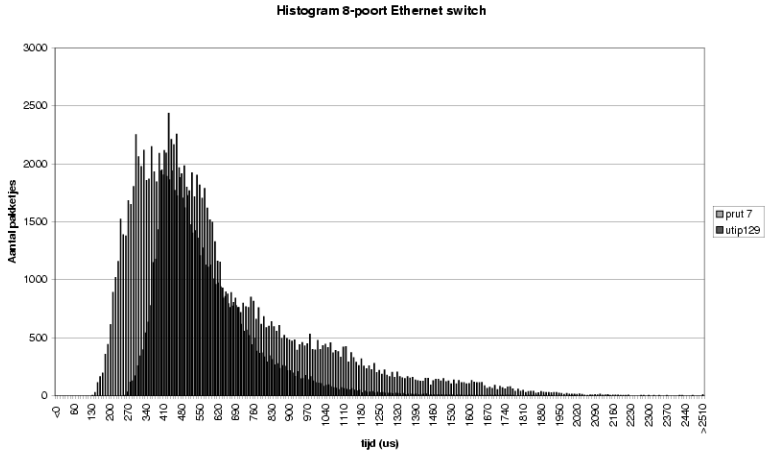
8-poort *Ethernet switch* volledig belast

Om de 8-poort *Ethernet switch* volledig te belasten is gebruik gemaakt van een opstelling van twee computers (figuur 4.2). De computers worden tegelijk gestart waarna ze gedurende een bepaalde tijd hetzelfde experiment gaan uitvoeren. De computers zijn niet met elkaar gesynchroniseerd. Het kan dus voorkomen dat aan het einde van de testrun de computers niet tegelijk stoppen. Dit maakt de laatste metingen minder betrouwbaar, maar gezien het aantal meetpunten zal de invloed hiervan niet erg groot zijn. Het blijkt echter wel noodzakelijk handmatig de verschillende tests te starten: aan het einde van een volledige testrun lopen de computers net te ver uit elkaar om nog een goed en betrouwbaar resultaat te krijgen. Om een goed beeld te geven zijn in figuur 5.9 de histogrammen van de beide machines apart gegeven. Duidelijk is dat het onderlinge verschil niet erg groot is.

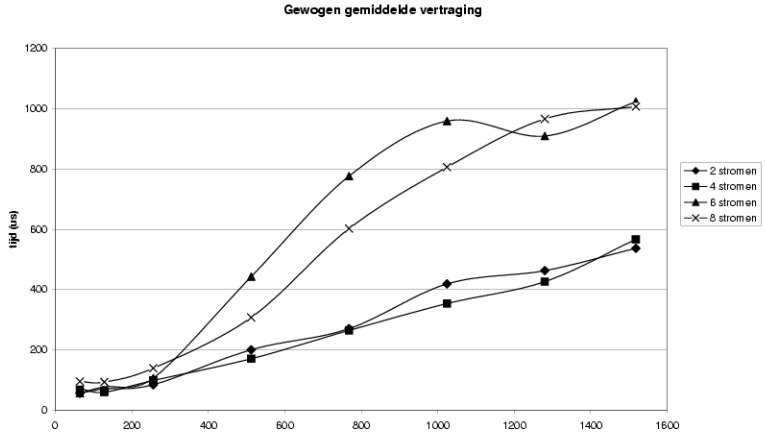
Het tijdverschil bij 1514 Byte wordt over de 8-poort *Ethernet switch* $495\mu s$. Als laatste wordt het figuur met de gewogen gemiddeldes van de meting gegeven. Duidelijk is in figuur 5.10 dat het tijdverlies ten opzichte van de pakket grootte anders verloopt dan bij de Belkin *Ethernet switch* (figuur 5.6). Het gedrag is dus voor een deel afhankelijk van de gebruikte chipset.

Throughput-meting

Bij de throughput-meting wordt het maximum aantal pakketjes bepaald, dat over de *Ethernet switch* verzonden kan worden zonder dat er pakket verlies optreedt. In deze paragraaf worden een aantal van de verschillen tussen metingen beschreven en een voorbeeld gegeven van een ideale situatie. Bij alle tests is er slechts een scenario uitgevoerd, aangezien er bij het gekozen scenario geen pakket verlies optreedt. Het scenario gaat uit van een enkele *burst* die gedurende een tijd verzonden wordt. De *burst*-lengte wordt dus bepaald door de tijd die de betreffende computer aan het zenden is. Zowel de 8-poort als de 5-poort *Ethernet switch* hebben een maximale tijd, waarin een poort niet gebruikt kan worden zonder dat de *switch* het adres vergeet. Om te voorkomen dat die situatie optreedt, is er voor een maximale tijd van 2 minuten gekozen. In tabel 5.1 is te zien dat de burst-lengte per tijdseenheid van PC1 in stappen wordt vermindert. Er wordt begonnen met de maximale tijd



Figuur 5.9: Histogram van 8-poort meting van PC1 en PC7



Figuur 5.10: Gewogen gemiddelde 8-poort meting van PC1 en PC7

PC 1	PC 2	PC 3	PC 4	PC 5	PC 6	PC 7	PC 8
7729	8124	8125	7233	8117	8127	8127	7278
7317	8124	8125	7234	8117	8127	8127	7278
6652	8124	8125	7232	8119	8127	8127	7278
5987	8124	8125	7231	8119	8127	8127	7278
5322	8124	8125	7232	8117	8127	8126	7278
4597	8124	8125	7233	8117	8127	8127	7278
3912	8124	8125	7233	8117	8127	8127	7278
3278	8124	8125	7233	8117	8127	8127	7278
2638	8124	8125	7231	8118	8127	8127	7278
1970	8124	8125	7230	8119	8127	8127	7278
1081	8125	8125	7231	8119	8127	8127	7278
540	8124	8125	7232	8119	8127	8127	7278
0	8125	8125	7231	8118	8127	8127	7278

Tabel 5.1: Aantal verzonden pakketjes/seconde

van twee minuten en deze tijd wordt verkort. In tabel 5.1is dit te zien aan het feit dat het aantal pakketjes per tijdseenheid kleiner wordt. Ook zijn niet alle computers snel genoeg om het maximale aantal pakketjes per tijdseenheid te versturen. Door een aantal computers te vervangen is dit verbeterd. Door het gebruik van verkeerde instellingen in het gebruikte meetprogramma is ook de situatie dat er bij de *switch* een onbekend adres aankomt getest. In tabel 5.1 is het aantal verzonden pakketjes per seconde voor de verschillende machines aangegeven. In deze meting halen twee machines het theoretisch maximum aantal pakketjes per seconde niet. Door de machines van de poort op de *switch* te laten wisselen is bewezen, dat het verminderde aantal pakketjes niet veroorzaakt wordt door de *Ethernet switch*

Bij dit gegeven scenario treedt in geen van de gevallen pakketverlies op blijkt uit de metingen. In alle gevallen komen de pakketjes over. Wanneer er echter een fout optreedt of een van de computers niet bereikbaar is worden de pakketjes door de *switch* gebroadcast. Indien er geen stroom met een verhoogde prioriteit is, treedt er wel pakketverlies op, slechts de helft van de door de zendende computer verstuurde pakketten wordt ontvangen. *Flow control* zou dit moeten voorkomen. Maar omdat de zendende computer een gelijk aantal pakketjes verstuurt als normaal, lijkt *flow control* geen enkele invloed te hebben op het aantal pakketjes dat verloren gaat.

RT-gedrag

De metingen geven aan dat er bij verschillende *Ethernet switches* verschillend gedrag optreedt, dat op hoofdlijnen toch vergelijkbaar is. Het gedrag, dat blijkt uit de metingen van het tijdsverschil over de *Ethernet switch*, komt overeen met het gedrag van de throughput-metingen. Hetgeen betekent, dat er alleen pakketverlies over de *Ethernet switch* optreedt, wanneer meerdere ingaande poorten naar een enkele uitgaande poort aan het zenden zijn. Dit betekent voor de *Ethernet switch* zelf, dat de gekozen vorm van de *switching fabric* niet van invloed is op de vertraging die de *switch* oplevert of op het pakketverlies. Pakketverlies wordt grotendeels veroorzaakt door het, met meerdere stromen, naar een enkele poort te zenden. De vertraging die de *switch* oplevert is afhankelijk van de strategie en de *scheduling* van de *Ethernet switch*. Tijdens het meten blijken de maximale duur van het bewaren van de MAC-tabel en het bekend zijn van de verschillende adressen een duidelijke invloed te hebben op het gedrag van de *Ethernet switch*. Op het moment dat het doeladres van een stroom niet bekend is worden de pakketjes van de stroom gebroadcast naar

alle poorten. Bij de verschillende poorten waar het pakketje niet voor bedoeld is komen dan tegelijk meerdere stromen aan waardoor pakketverlies optreedt. Om *real-time* garanties te kunnen geven is het van belang te voorkomen dat, om wat voor reden dan ook, de *Ethernet switch* gaat broadcasten.

5.2 Resultaten model

Zoals is aangegeven in hoofdstuk 3, wordt door de gebruikte modellen het gedrag van de *Ethernet switch* bepaald door de lengte van de wachtrij te berekenen. Om de resultaten van de metingen te benaderen wordt aan de hand van de metingen de Constante vertraging van de *Ethernet switch* bepaald.

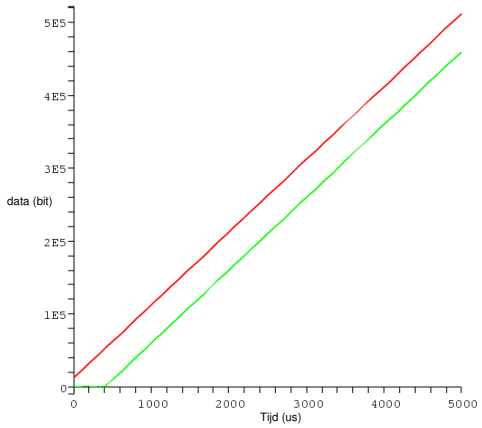
Voor het vergelijken van de resultaten van de modellen met de resultaten van de metingen is gekozen voor dezelfde scenario's als bij de metingen. Bij deze scenario's wordt er van uitgegaan dat de uitkomst het slechtst voorkomende geval is, namelijk dat alle stromen op hetzelfde moment beginnen te zenden. En de grootste belasting op de *Ethernet switch* ontstaat op het moment dat alle stromen op hetzelfde moment beginnen met zenden. Het tegelijk beginnen met zenden van de stromen moet de grootste belasting op de *switch* geven. Als de *switch* deze belasting aankan dan moet het ook minder zware lasten aankunnen.

In deze paragraaf worden de resultaten in dezelfde volgorde beschreven als in hoofdstuk 3. Voor alle modellen wordt de aanname gedaan dat er geen *flow control* aanwezig is. *Flow control* maakt het onmogelijk de stroom te voorspellen. Daardoor wordt het gedrag ten gevolge van de stroom onvoorspelbaar. Bij gevallen waarin flow control mogelijk kan optreden wordt volstaan met de maximale wachttijd. Daarbij wordt gebruik gemaakt van het feit dat de modellen de situatie in het slechtste geval geven.

Netwerk calculus

Met behulp van netwerk calculus wordt de maximale *backlog* over de *Ethernet switch* berekend. Bij dit model wordt er van uitgegaan dat de eigenschappen van de stroom bekend zijn. Deze methode wordt onder andere gebruikt door het RSVP protocol [RSV] bij het reserveren van bandbreedte in routers. In ons geval worden er aan de hand van de meetgegevens de maximale *backlog* en de maximale vertraging over de *Ethernet switch* voor een enkele stroom berekend. Het uitgebreid beschreven voorbeeld behandelt de situatie zoals die van de 5-poort *Ethernet switch*, voor een enkele of meerdere stromen met een pakketgrootte van 1518 Byte. Uit de metingen is afgeschat dat t_{mux} ongeveer $400\mu s$ zal zijn voor een pakketje met een grootte van 1518 Byte.

In figuur 5.11 is te zien dat de aankomstkromme α de buffer begint te vullen. Nadat de t_{mux} is afgelopen begint de *Ethernet switch* de pakketjes met servicekromme β verder te sturen. Duidelijk is te zien dat bij het gekozen scenario, een volledig bezette lijn, een evenwicht ontstaat op het moment dat de servicekromme begint. De langs de y-as staande waardes is de hoeveelheid data verzonden uitgedrukt in bit. De maximale *backlog* van dit systeem blijft ruimschoots onder de maximale buffergrootte van 256 kB. De maximale delay veroorzaakt in dit systeem wordt berekend met behulp van formule 3.11. De uitkomst van $\frac{13028}{25} = 521\mu s$ komt overeen met de waarde van de 5-poort meting zoals eerder in dit hoofdstuk beschreven. Aan figuur 5.11 is duidelijk te zien dat op het moment dat de *Ethernet switch* begint met zenden de wachtrij niet groeit en er dus een evenwicht ontstaat. Dit komt ook overeen met de meetresultaten. In tabel 5.2 wordt per pakketgrootte het maximaal berekende tijdverlies over de *switch* weergegeven. Dit is gedaan met de theoretisch maximale waarde voor de verschillende pakketgroottes. In het geval een pakketje



Figuur 5.11: Gegeven aankomstkromme (bovenste lijn) en bijbehorende service-kromme (onderste lijn)

pakketgrootte	maximale backlog (bit)	$t_{switch}(\mu s)$
64	5512	55
128	6024	60
256	7048	70
512	44096	440
768	46144	461
1024	48192	481
1276	50240	502
1518	52100	521

Tabel 5.2: Resultaten netwerk calculus met maximale burst groottes

kleiner is dan de paginagrootte wordt er een correctie uitgevoerd op de maximale waarde van de wachtrij. Het aantal pakketjes in de wachtrij wordt in dat geval namelijk beperkt door het aantal beschikbare pagina's voor de beschreven wachtrij.

De resultaten uit de tabel kunnen vergeleken worden met de resultaten met de 5-poort *Ethernet switch*. Voor de grotere pakketten benadert het resultaat de gemiddeldes van de *Ethernet switch*, echter het tijdverlies van de kleinere pakketten komt niet overeen. Hieruit blijkt dat voor grotere pakketten het tijdverlies afhangt van de lengte van de wachtrij en de verwerkingstijd van de *Ethernet switch*. Het tijdverlies van kleine pakketten wordt blijkbaar nog beïnvloed door andere grootheden. Een van de oorzaken hiervan kan zijn dat de interne MAC-tabel van de *Ethernet switch* de grote belasting niet aankan, of dat een pakketje eerder geblokkeerd wordt bij bepaalde pakketgroottes. Door uit te gaan van het maximale tijdverschil over de *Ethernet switch* van het grootste pakket waarvoor dit model eigenlijk bedoeld is kan toch geschat worden hoeveel tijd een pakketje nodig heeft om via de *switch* verzonden te worden zonder dat er verlies optreedt. Alle berekende *backlogs* blijven ruim binnen de beschikbare geheugenruimte van de *Ethernet switch*. Bij dit model is de maximale *delay* die kan optreden over de *Ethernet switch* een functie van de maximale hoeveelheid beschikbare geheugen ruimte en de *multiplexing delay* van de *Ethernet switch*. De maximale tijdsduur die een pakketje in de *Ethernet switch* door kan brengen wordt gegeven in tabel 5.3.

Dit geeft een maximale vertraging van $2.6ms$ welke iets hoger ligt dan het gemeten maximum. Wanneer je de *multiplex* variabele bij deze waardes optelt komt de

pakketgrootte	$t_{switch}(\mu s)$
64	676
128	1353
256	2621
512	2621
768	2621
1024	2621
1276	2621
1518	2618

Tabel 5.3: Resultaten netwerk calculus maximale delay over de Ethernet switch

maximale delay zelfs overeen met de gemeten maxima. Dit komt echter niet zo heel vaak voor. Naarmate het tijdverlies over de *switch* groter wordt neemt de invloed van het constante tijdverlies over de *switch* af. Theoretisch gezien kunnen er geen pakketjes zijn die over dit maximum heengaan, omdat die niet door de switch in het systeem worden toegelaten. De lagere maxima voor de kleinere pakketjes worden, zoals eerder beschreven, veroorzaakt door de geheugenindeling. Er wordt minder efficiënt gebruik gemaakt van het maximaal beschikbare geheugen.

Stochastisch model

Bij het stochastische model wordt aangenomen dat de Ethernet stroom een Markov keten is en negatief exponentieel is verdeeld. Dit laatste houdt in dat er een groot aantal pakketjes verstuurd wordt met daartussen een klein tijdsverschil. Aangezien er geen natuurlijk Ethernet wordt gesimuleerd is dit een verdedigbare aanname. Gekozen wordt voor een $(M|M|1||x)$ model voor pakketjes met een pakketgrootte kleiner dan de paginagrootte. De paginagrootte is afhankelijk van de gekozen *Ethernet switch*. De paginagrootte is echter van belang voor de kleinere pakketten en beinvloedt de wachttijd van de *Ethernet switch*. Tabel 5.4 geeft de kans op pakketverlies voor een enkele stroom voor verschillende bezettingsgraden van de *Ethernet switch*. Het totaal aantal pagina's wordt geschat op 1024 pagina's van 256 B, dit is overeenkomstig met de data sheet van de RTL 8309 [RTL03b]. Voor pakketgroottes groter dan 256 B is het aantal vrije plaatsen in de buffer $\frac{pakketgrootte}{256}$ Dit levert het aantal pagina's op dat bezet wordt door de verschillende pakketten. Door deze benadering is er de mogelijkheid een optimum in de bezetting van de buffer van de *switch* te vinden overeenkomstig met de metingen.

Voor de uitgaande stroom wordt aangenomen dat deze altijd maximaal is en afhankelijk van de pakketgrootte. De ingaande stroom loop van minimaal dus 0 tot maximaal dat is de lijnbezetting die voor een gekozen pakketgrootte maximaal is. Om een goede schatting van het aantal pakketjes in de wachtrij te kunnen geven wordt er een vaste service waarde bijgeteld en een variabele berekend, zoals beschreven in vergelijking 3.1 in hoofdstuk 3 op pagina 15. In de resultaten wordt de kans op het verloren gaan van een pakketje, de verwachte lengte van de wachtrij en de verwachte vertraging over deze wachtrij.

In tabel 5.4 wordt net als in het voorbeeld van hoofdstuk drie de kans berekend dat bij een bepaalde bezettingsgraad van de lijn een pakket verloren gaat. In tabel 5.4 is te zien dat de kans dat bij een normale netwerk bezetting de kans dat er een pakket verloren gaat zeer klein is.

Bij een volledig bezette verbinding is de kans dat een pakketje verloren gaat nog maar 0.7%. Deze is berekend met behulp van Maple door de limiet van ρ naar 1 te nemen. Voor een Ethernet gebaseerd netwerk is de maximale bezetting van een

ρ	kans
0.5	$1.46 \cdot 10^{-39}$
0.6	$1.60 \cdot 10^{-29}$
0.7	$4.46 \cdot 10^{-21}$
0.8	$7.88 \cdot 10^{-14}$
0.9	$1.39 \cdot 10^{-7}$
1.0	$7.75 \cdot 10^{-3}$

Tabel 5.4: De kans op verlies bij een normale netwerk bezetting

pakketgrootte	$E[N]$	vertraging (μs)
64	11.90	87
128	6.75	92
256	3.62	102
512	9.39	443
768	6.34	463
1024	4.79	483
1276	3.84	504
1518	3.24	523

Tabel 5.5: Lengte wachtrij en de vertraging over de wachtrij

stroom echter altijd kleiner dan 1 zo lang er met maar een zender naar een enkele ontvanger gestuurd wordt. Dit komt overeen met de resultaten van de netwerk calculus. Ook daar blijven de maximale hoeveelheden pakketten ruim beneden de buffer grootte. De verwachte aankomst- en de service frequenties van de pakketjes zijn afhankelijk van de pakketgrootte. De verwerkingstijd ($E[R]$) wordt dan: t_{mux}. De verwachte lengte van de wachtrij wordt dan $E[N] = \lambda E[R] + E[S]$. Nu kan aan de hand van deze formule de lengte van de wachtrij worden berekend en aan de hand daarvan de tijd die nodig is om deze wachtrij te verwerken.

Voor tabel 5.5 zijn de gegevens van de belkin 5-poort *Ethernet switch* gebruikt. Gezien het knikpunt in deze grafiek bij de kleinere waardes is aangenomen dat daar de t_{mux} ook kleiner is. Dit levert een overeenkomstig resultaat op (zie figuur 5.6). Het is niet mogelijk om dit soort afwijkingen van het ideaal gedrag op voorhand zonder metingen te voorspellen. Echter in het geval er data verstuurd wordt bestaat de stroom voornamelijk uit de grootste pakketjes, en wanneer er controle berichten verstuurd worden zullen er alleen pakketjes met de minimale pakketgrootte worden verstuurd.

5.3 Real-time gedrag

Het *real-time* gedrag van de *Ethernet switch* is afhankelijk van een beperkt aantal factoren, het tijdverlies en het pakket verlies gemeten over de *Ethernet switch*. Zowel uit de metingen als uit de modellen blijkt dat het pakket verlies over de *Ethernet switch* pas een rol gaat spelen wanneer er met meerdere stromen naar een enkel doel adres gestuurd wordt. Zolang er met een enkele stroom naar een enkel doel adres gestuurd wordt treedt er geen pakketverlies op. De andere grootheid het tijdverlies over de *switch* wordt verondersteld lineair te zijn en afhankelijk van de pakketgrootte. Uit de metingen blijkt dat dit niet voor iedere *Ethernet switch* zo is en afhankelijk van het model en merk van de *Ethernet switch*. Door de gegevens van de metingen in de modellen in te voeren blijkt de verandering van het gedrag

alleen te kunnen zijn veroorzaakt door de constant veronderstelde t_{mux}. Een van de mogelijkheden voor dit gedrag zou kunnen zijn dat de interne bus in de *switch* toch niet snel genoeg is. Er is echter te weinig data te verkrijgen over de werkelijke architectuur van de *Ethernet switch* om de eigenschappen van een eventuele bus in meer detail te kunnen bepalen. Door de t_{mux} goed te bepalen is het mogelijk een goede inschatting te geven van de tijd die een pakketje er over gaat doen om verstuurd te worden aan de hand van de gegeven modellen bij normale omstandigheden.

In het geval van meerdere stromen kan de maximale tijdsduur dat een Ethernet pakketje onderweg is ook worden afgeschat, alleen door extra services die de huidige *Ethernet switch*es bieden, kunnen er geen garanties aan ontleend worden. Indien er een vorm van *flow control* aanwezig is zal deze het gedrag van de *Ethernet switch* beïnvloeden en daardoor het gedrag onvoorspelbaar maken. Met de modellen is het mogelijk het moment dat de *flow control* wordt ingeschakeld te voorspellen en te voorkomen dat de *flow control* wordt ingeschakeld.

De afzonderlijke stromen door de *Ethernet switch* hebben wel invloed op elkaar, maar niet in die mate dat ze elkaars gedrag beïnvloeden. Bij het ijken van de testmachines beïnvloeden de stromen elkaar duidelijker dan bij de metingen over de *Ethernet switch*, het is dus de vraag of de invloed van de stromen op elkaar niet worden beïnvloed door de gebruikte meetapparatuur.

Hoofdstuk 6

Discussie en conclusie

Het gedrag van de *Ethernet switch* wordt bepaald door de interne architectuur en de gekozen *switching strategy*. Beide hebben invloed op de wachtrijen en de wachttijden gemeten over de switch. Door de lengte van de wachtrij te voorspellen is het mogelijk de verblijftijd van een pakketje in de *Ethernet switch* te bepalen. Behoudens de lengte van de wachtrij is de verblijftijd van een pakketje in de *switch* afhankelijk van de tijd die de *switch* nodig heeft om het doeladres van het pakketje op te zoeken in zijn MAC-adres tabel. Deze tijd is afhankelijk van de fabrikant van de *switch* en is constant voor het model en voor het type. Het meten van deze tijd maakt het mogelijk de totale lengte van de wachtrij te bepalen. Het is niet mogelijk om de *switch* te gebruiken voor *hard real-time* applicaties. Opties zoals *flow control* maken het gedrag van de stromen over de *switch* te onvoorspelbaar.

Uit zowel het voorspellen van de lengte van de wachtrij als de metingen blijkt dat het pakket verlies over de *Ethernet switch* pas een rol gaat spelen wanneer er met meerdere stromen naar een enkel doel adres gestuurd wordt. Zolang er met een enkele stroom naar een enkel doel adres gestuurd wordt treedt er geen pakketverlies op en is de wachttijd in de switch afhankelijk van de pakketgrootte en de constante wachttijd in de switch.

In het geval van meerdere stromen kan de maximale tijdsduur dat een Ethernet pakketje onderweg is ook worden afgeschat, alleen door extra services die de huidige *Ethernet switch*es bieden, kunnen er geen garanties aan ontleend worden. Indien er een vorm van *flow control* aanwezig is zal deze het gedrag van de *Ethernet switch* beïnvloeden en daardoor het gedrag onvoorspelbaar maken. Met de modellen is het mogelijk het moment dat de *flow control* wordt ingeschakeld te voorspellen en te voorkomen dat de *flow control* wordt ingeschakeld. De afzonderlijke stromen door de *Ethernet switch* hebben wel invloed op elkaar, maar niet in die mate dat ze elkaars gedrag beïnvloeden.

Gebruik van een *Ethernet switch* in RTnet Het huidige RTnet maakt gebruik van een enkelvoudige stroom die over een hub gestuurd wordt. Dit is ook mogelijk voor de *Ethernet switch*. Wanneer er *hard real-time* garanties gegeven moeten worden over de stroom is het niet mogelijk meerdere stromen tegelijk te verzenden. De afzonderlijke stromen beïnvloeden elkaar niet, maar de garantie dat bepaalde controle pakketjes op hetzelfde moment verzonden ook op hetzelfde moment aankomen, kan niet gegeven worden. Aangezien sommige controle pakketjes *broadcast* pakketjes zijn kan het voorkomen dat bij een druk bezette poort pakketjes verloren gaan.

Bibliografie

[AOM02] James Aweya, Michel Ouellette, and Delfin Y. Montuno. Interworking of switched ethernet and ATM flow control mechanisms. *International Journal of Network Management*, 12(6):357–366, nov/dec 2002.

[BM96] Scott Bradner and Jim McQuaid. *Benchmarking Methodology for Network Interconnect Devices*. RFC Editor, may 1996. RFC 1944.

[BM99] Scott Bradner and Jim McQuaid. *Benchmarking Methodology for Network Interconnect Devices*. RFC Editor, March 1999. RFC 2544.

[BT00a] Jean-Yves Le Boudec and Patrick Thiran. A short tutorial on network calculus 1: fundamental bounds in communication networks. *Proceedings ISCA2000*, may 2000.

[BT00b] Jean-Yves Le Boudec and Patrick Thiran. A short tutorial on network calculus 2: Min–plus system theory applied to communicatin networks. *Proceedings ISCA2000*, may 2000.

[BT01] Jean-Yves Le Boudec and Patrick Thiran. *Network calculus, A theory of deterministic queuing systems for the Internet*. Springer, first edition, 2001. ISBN 3-540-42184-X.

[Eng] John English. `http://burks.brighton.ac.uk-/burks/pcinfo/hardware/ethernet/switch.htm`.

[Hav98] Boudewijn R. Haverkort. *Performance of computer communication systems*. John Wiley & Sons, Ltd, 1998. ISBN0-470-84192-3.

[HJSM05] Ferdy Hanssen, Pierre G. Jansen, Hans Scholten, and Sape Mullender. RTnet: a distributed real-time protocol for broadcast-capable networks. In *Proceedings Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS/ICNS 2005)*, Papeete, French Polynesia, October 2005. IEEE Computer Society Press. ISBN 0-7695-2450-8.

[IEE99] Institute of Electrical and Electronics Engineers. *IEEE 802.1 P,Q - QoS on the MAC level*, 1999. IEEE Std. 802.1P.

[IEE02] Institute of Electrical and Electronics Engineers. *IEEE Standard for Information Technology– Telecommunications and Information Exchange between Systems– Local and Metropolitan Area Networks– Specific Requirements– Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, 2002. IEEE Std. 802.3-2002.

[KR01] James F. Kurose and Keith W. Ross. *Computer Networking– A Top-Down Approach Featuring the Internet*. Addison-Wesley, first edition, 2001. ISBN 0-201-47711-4.

[KS804] Micrel. *KS8995X integrated 5 port 10/100 QoS switch rev.2.1*, June 2004.

[Loe03] Jork Loeser. Buffer bounds of a FIFO multiplexer. Technical Report TUD-FI03-15, Fakultät Informatik, TU Dresden, Dresden, Germany, November 2003.

[MB76] Robert M. Metcalfe and David R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, July 1976.

[RSV] Rsvp project. `http://www.isi.edu/div7/rsvp/rsvp.html`.

[RTL02] Realtek Semiconductor Corp. *RTL8305S 5-Port 10/100 Mbps Single Chip Switch Controller Data Sheet*, February 2002.

[RTL03a] Realtek Semiconductor Corp. *RTL8305SB Single-chip 5-Port 10/100 Mbps Switch Controller with Dual MII Interfaces Data Sheet*, June 2003.

[RTL03b] Realtek Semiconductor Corp. *RTL8309SB Single-chip 9-Port 10/100 Mbps Switch Controller Data Sheet*, April 2003.

[ZL503] Zarlink Semiconductor Inc. *ZL50409 Managed 9-Port 10/100 M Ethernet Switch Data Sheet*, jan 2003.

[ZL504] Zarlink Semiconductor Inc. *ZLAN-44 Applications of the ZL50400/4/5/7/8/9/10/11 Programming Flow Control Registers Application Note*, December 2004.

Bijlage A

Tools

Voor het meten zijn de volgende programma's en scripts gebruikt:

- smull.ko, Smull is de module die nodig is om in de custom kernel de vier netwerk kaarten an te sturen
- 3c59xt.ko, aangepaste driver gebruikt voor de metingen
- eepro100t.ko, aangepaste driver gebruikt voor de metingen
- snt, Simple Netwerk Tester, user space programma om data van user space naar de Smull module te sturen
- tester, socket programma gebruikt voor de throughput test
- parse, zet de meetdata om naar een histogram
- plot, script om data naar eps bestanden te plotten
- plot_script om data naar png bestanden te plotten
- plot_all.sh script dat een tarball opent en de data plot door gebruik te maken van plot en plot_png