Communicatie tussen processoren via crossbar switch

Seelen, H.A.J.M.

Award date:
1989

Communicatie tussen processoren via crossbar switch

H.A.J.M. Seelen

FI/FIV 89-15

Augustus 1989

Verslag van het afstudeeronderzoek, verricht van oktober 1988 tot augustus 1989, in de vakgroep Fysische Informatica van de faculteit der Technische Natuurkunde van de Technische Universiteit Eindhoven, onder leiding van prof. dr. A.J. van der Wal.

Samenvatting

Voor efficiënte communicatie tussen processoren in een multiprocessor systeem is de architectuur van een crossbar switch ontwikkeld. Met behulp van deze crossbar switch is het mogelijk om directe verbindingen te leggen tussen processoren onderling en het is bovendien mogelijk het netwerk tussen de processoren onderling dynamisch te reconfigureren, dat wil zeggen dat tijdens het uitvoeren van programma's verbindingen tussen processoren omgelegd kunnen worden. Het is essentieel dat er geen datatransport plaatsvindt tijdens reconfiguratie van het netwerk. Om dit te kunnen garanderen is er een protocol ontwikkeld.

Tijdens het afstudeeronderzoek zijn er twee verschillende versies van de implementatie van het protocol voor de single crossbar switch ontwikkeld. Met deze versies zijn simulaties uitgevoerd om de invloed van de belastinggraad van het netwerk en van de lengte van een bericht op de overzendtijd van een bericht te kunnen bepalen.

Inhoudsopgave

Samenvatting

Inhoudsopgave

1. Inleiding

In een multiprocessor omgeving is het van belang dat processoren onderling efficiënt kunnen communiceren. Communicatie tussen processoren kan men globaal in drie klassen indelen.

Ten eerste is communicatie mogelijk via shared bus of shared memory. Door de begrensde communicatie bandbreedte is het aantal processoren dat van de shared bus (of shared memory) gebruik kan maken ook beperkt.

Met de tweede methode is deze beperking komen te vervallen. Door gebruik te maken van point-to-point verbindingen kan men een ongelimiteerd aantal processoren met elkaar verbinden. Het nadeel van dit systeem is dat er alleen een bepaald maximum aantal processoren direct met elkaar kunnen communiceren. Heeft men meer processoren in een systeem dan moeten berichten via andere processoren naar de eindbestemming gedirigeerd worden. Hiervoor moeten routing algoritmen geïmplementeerd worden waardoor de performance van het systeem daalt.

De laatste methode heft dit nadeel op door het routen van berichten uit te laten voeren door onafhankelijke hardware. In plaats van vaste verbindingen is de hardware in staat verbindingen tussen processoren dynamisch te reconfigureren, waardoor tussen alle processoren directe communicatie mogelijk is.

Het afstudeeronderzoek

De methode van het dynamisch reconfigureren van het netwerk heeft als basis gediend voor de crossbar switch. Tijdens de afstudeerperiode is er een protocol ontwikkeld voor communicatie tussen processoren in een crossbar switch en programmatuur ontwikkeld voor de hardware die de verbindingen moet reconfigureren. De opbouw van de crossbar switch staat beschreven in hoofdstuk 3.

Daar de hardware voor crossbar switch nog niet gebouwd was, is er programmatuur geschreven voor de simulatie van de crossbar switch. Alle software is geschreven in de programmeertaal OCCAM 2 en ontwikkeld op het Transputer Development System 2.0 van INMOS (TDS). Het TDS staat beschreven in hoofdstuk 2.

Verder heeft het afstudeeronderzoek zich toegespitst op het nagaan van de invloed van de berichtlengte en de belastinggraad van het netwerk op de overzendtijd van een bericht. Er zijn simulaties gedaan met twee versies van de programmatuur voor de hardware. Deze simulaties met de bijbehorende resultaten worden besproken in hoofdstuk 4.

Ten slotte staan de conclusies die uit de simulaties volgen alsmede een aantal aanbevelingen vermeld in hoofdstuk 5.

2. De TDS opstelling

Alle software tijdens de afstudeerfase is ontwikkeld op het Transputer Development System van INMOS (TDS). Het TDS bestaat uit een hardware gedeelte en een software gedeelte. Het hart van het hardware gedeelte is de transputer, een speciaal soort processor.

In dit hoofdstuk zullen eerst de specifieke eigenschappen van de transputer behandeld worden en daarna de gehele TDS opstelling zoals deze in de vakgroep Fysische Informatica aanwezig is.

2.1 De transputer

Een transputer (ref. 1) is een RISC-processor (Reduced Instruction Set Computer) voorzien van lokaal geheugen en communicatie links. De transputer is ontworpen voor parallelle gegevensverwerking. De transputer kan met behulp van de communicatie links op eenvoudige en efficiënte wijze met andere transputers communiceren.

Mede dankzij instructies om processen op te starten, interproces communicatie op te starten, de prioriteit van een proces te bepalen en een proces te blokkeren, is de transputer zeer krachtig in multitasking. Parallelle processen in een transputer netwerk kunnen dan fysiek concurrent zijn (ieder proces op een eigen transputer) of virtueel concurrent zijn (meerdere processen draaien multitasking op één transputer). Het is dan mogelijk parallelle programma's te schrijven zonder dat het aantal transputers van te voren bekend is. Hierdoor kan de performance van een systeem bijna lineair toenemen naarmate er meer transputers in het netwerk worden toegevoegd.

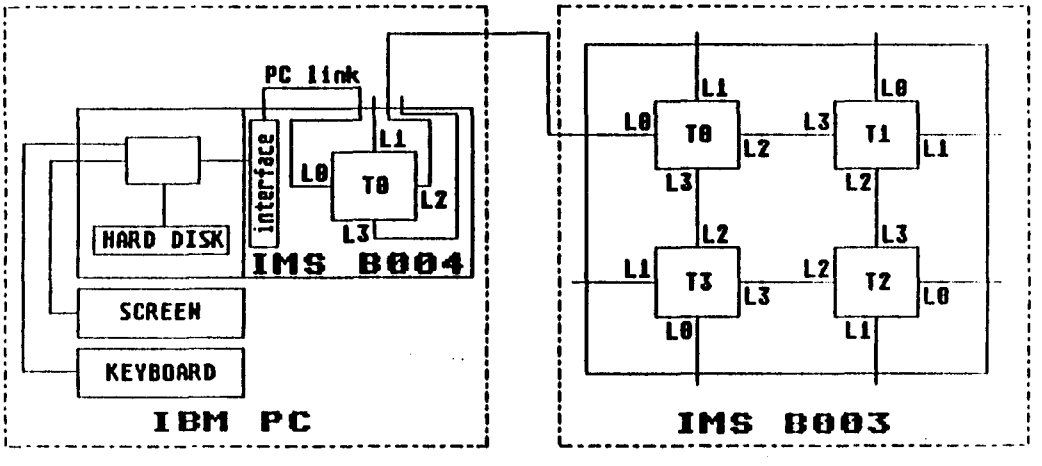
Samen met de transputer is de taal OCCAM (ref. 2, 3) ontworpen. Met OCCAM is het mogelijk om een applicatie te schrijven als een set van parallelle processen die onderling communiceren via kanalen. Mede hierdoor kan OCCAM de mogelijkheid tot multitasking volledig benutten.

2.2 Het Transputer Development System

Het Transputer Development System 2.0 van INMOS bestaat, zoals al is vermeld, uit een hardware en een software gedeelte. Het hardware gedeelte bestaat uit een insteekkaart (IMS B004, ref. 4) voor de IBM-PC. Op deze kaart zit een transputer T414 van INMOS met 1 Mbyte RAM geheugen, een interface naar de PC en een interface naar andere transputer systemen.

De T414 is een 32 bit 10 MIPS transputer met 2 Kbyte cache RAM geheugen, 4 intertransputer links (tot 20 Mbit/sec) en een memory controller. De interface naar de PC zorgt ervoor dat de transputer gebruik kan maken van de I/O-faciliteiten van de PC (scherm, toetsenbord, diskdrives etc.). De interface naar de andere transputer systemen zorgt ervoor dat andere transputers met de transputer in het TDS verbonden kunnen worden. Bij de vakgroep Fysische Informatica is daarvoor een multitransputer kaart (IMS B003, ref. 5) aanwezig met daarop vier T414 transputers met ieder 256 Kbyte RAM geheugen. De gehele opstelling is te zien in figuur 1.

Met behulp van het software gedeelte kan men op de TDS programma's invoeren (in OCCAM 2, verbeterde versie van OCCAM), compileren en uitvoeren (ref. 6). Verder kan men met behulp van bijgeleverde bibliotheek procedures naar het scherm schrijven, van het toetsenbord lezen en naar de schrijven en lezen.



Figuur 1. Schematische weergave van het Transputer Development System (TDS) van INMOS.
T0, T1, T2 & T3: transputer 0, 1, 2 & 3 (INMOS T414);
L0, L1, L2 & L3: link 0, 1, 2 & 3 van transputer;
Interface in IMS B004 zorgt voor connectie tussen transputer en I/O van PC.

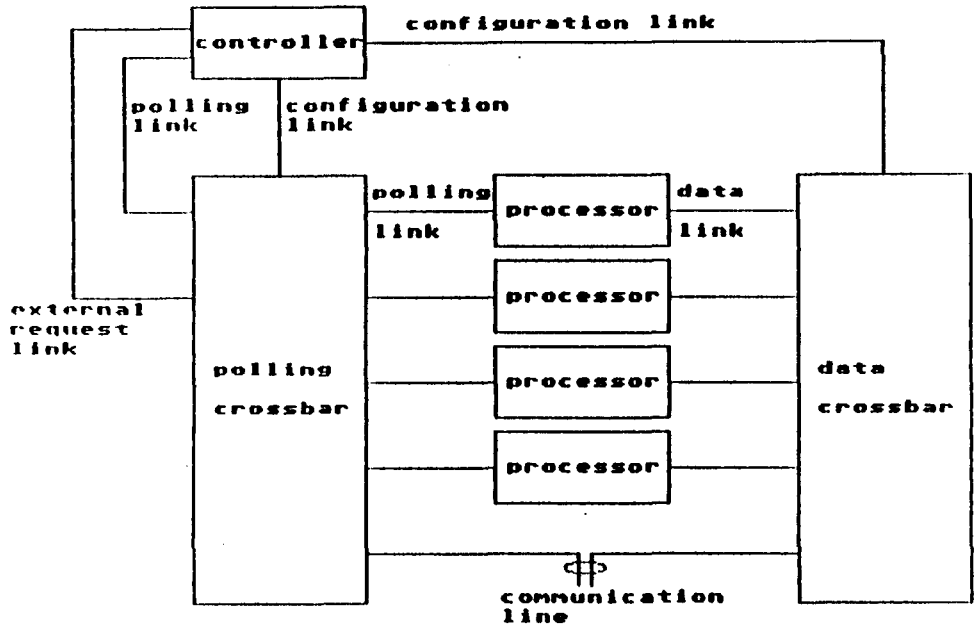
3. De crossbar switch

Voor dynamisch reconfigureerbare netwerken is het nodig dat verbindingen tussen processoren gelegd en verbroken kunnen worden. De firma INMOS produceert het IMS C004 crossbar IC dat de mogelijkheid heeft verbindingen te leggen en te verbreken. Dit IC is ook het uitgangspunt geweest voor de crossbar switch, waarvan de opbouw in paragraaf 3.1 besproken zal worden.

Van belang is natuurlijk dat tijdens het leggen en verbreken van verbindingen tussen twee processoren geen communicatie tussen deze twee plaats vindt. Een protocol (ref. 7) dat beschreven wordt in paragraaf 3.2 zorgt ervoor dat dit vermeden wordt. De implementatie van dit protocol wordt beschreven in paragraaf 3.3.

3.1 Opbouw crossbar switch

Zoals in de bovenstaande paragraaf al vermeld is, wordt de IMS C004 crossbar IC (ref. 8, 9) gebruikt in de crossbar switch (ref 7, 10). Er kunnen 32 bidirectionele communicatie links op dit IC worden aangesloten. Tussen deze 32 communicatie links kunnen verbindingen gelegd en verbroken worden door via een speciale link, de configuration link, commando's te sturen.



Figuur 2. De crossbar switch.

In figuur 2 staat het schema voor de crossbar switch weergegeven. In de crossbar switch worden twee crossbar IC's gebruikt, nl. de polling crossbar en de data crossbar. De configuration links van deze twee crossbars zijn verbonden met een speciale processor, de controller genaamd, en zodoende de twee crossbars kan programmeren en dus bepaald hoe de verbindingen in de crossbars liggen.

De processoren in de crossbar switch zijn met één link aan de polling crossbar en met één link aan de data crossbar verbonden. Indien we voor de processor een transputer gebruiken, dan blijven er van de vier communicatie links van de transputer twee links over. In de tijdens de afstudeerfase onderzochte opstelling blijven deze links ongebruikt. Als een processor met een andere processor wil communiceren dan moet deze processor dit eerst kunnen melden aan de controller, omdat alleen deze de verbindingen kan leggen. Daartoe heeft de controller een link van zichzelf (de polling link) verbonden met de polling crossbar. De controller kan zichzelf zo met de polling link van iedere processor in de crossbar switch verbinden en deze processor ondervragen (pollen). Als na het pollen blijkt dat een processor wil communiceren met een andere processor dan meldt hij dit aan de controller en deze legt dan een verbinding in de data crossbar tussen de twee processoren.

De polling crossbar wordt alleen gebruikt voor het overbrengen van berichten tussen processor en controller, en de data crossbar voor communicatie tussen processoren onderling. Een verbinding tussen twee processoren in de data crossbar heet datapad.

De controller en de twee crossbars noemen we de crossbar switch en zal verder aangeduid worden als switch. Doordat de controller uitsluitend via de polling crossbar met de processoren kan communiceren, kan de controller alleen het linknummer van de polling crossbar nagaan waaraan de polling link van een processor is verbonden, maar niet het linknummer van de data crossbar waaraan de data link van deze processor is verbonden. We veronderstellen dan ook dat het linknummer van de data crossbar gelijk is aan het linknummer van de polling crossbar, waaraan respectievelijk de data link en de polling link van een processor liggen. Een data link van de data crossbar met

bijbehorende polling link van de polling crossbar noemen we een communication line.

Aan een dergelijke communication line kan dus een processor worden verbonden, maar ook een communication line van een andere switch. Hierdoor bestaat de mogelijkheid om meerdere switches te cascaderen en dus het aantal processoren dat direct met elkaar kan communiceren te vergroten.

De controller van een switch kan met een controller van een andere switch communiceren door zijn polling link te verbinden met de polling link van de communication line naar de andere switch. De controller wacht dan net zo lang totdat de andere controller zijn polling link verbindt met de polling link van de communication line, waarna er een directe verbinding tussen de twee controllers is, zodat ze met elkaar kunnen communiceren.

Een deadlock kan optreden indien twee of meer controllers van verschillende switches op elkaar wachten. Om dit te voorkomen is er een tweede link toegevoegd, de external request link. Deze external request link zorgt alleen voor de communicatie tussen de controllers onderling, de polling link zorgt voor het pollen van de communication lines in de switch.

In de rest van dit verslag hebben we het alleen over de single crossbar switch, zodat de external request link kan komen te vervallen en er 31 communication lines in de switch overblijven.

3.2 Protocol voor reconfiguratie crossbar switch

Wanneer in de switch verbindingen gelegd en verbroken worden is het van belang dat er tijdens het leggen en verbreken van deze verbindingen geen data overdracht plaats vindt over deze verbindingen. De controller kan niet direct de data overdracht tussen twee processoren controleren, hetgeen ook wenselijk is in verband met de snelheid. De controller moet daarom met de processoren samenwerken om te zorgen dat er geen data overdracht plaats vindt bij reconfiguratie.

Een processor die met een andere processor wil communiceren, vraagt een datapad aan naar die andere processor aan de controller. Bij dit datapad hoort een mastership. De processor die het mastership heeft, controleert de data overdracht over het

datapad. Geen data overdracht tijdens reconfiguratie van het netwerk wordt gewaarborgd door in achtneming van de volgende drie regels.

(1) De controller creëert een mastership nadat hij een datapad tussen twee processoren heeft aangelegd. Het mastership wordt gegeven aan de processor die het datapad heeft aangevraagd.

(2) De controller vraagt het mastership van een datapad op van de processor die het mastership heeft, wanneer hij een verzoek tot verbreken van het datapad heeft ontvangen. De processor geeft na afloop van de data overdracht over het datapad het mastership aan de controller. De controller verbreekt het datapad en vernietigt daarna het mastership.

(3) Een controller mag altijd verbindingen gebruiken die nog niet in een datapad gebruikt worden. Als een controller een verbinding wil gebruiken, die wel in een datapad gebruikt wordt, zal eerst dat datapad verbroken moeten worden.

3.3 Implementatie protocol in crossbar switch

De implementatie van het in de voorafgaande paragraaf beschreven protocol op de crossbar switch zal hier beschreven worden. De controller pollt de processoren in de switch sequentieel af. Als de controller pollt dan kunnen we zeven verschillende situaties waarin de processoren verkeren onderscheiden.

(1) De processor heeft geen datapad en wil niet communiceren met een andere processor. De processor geeft dit door aan de controller en de controller zal verder gaan met pollen.

(2) De processor heeft geen datapad en wordt benodigd in communicatie met andere processor. Na melding van de processor aan de controller, pollt de controller verder.

(3) De processor heeft geen datapad en wil communiceren met een andere processor, die we destination processor zullen noemen. De processor die wil zenden noemen we de requesting processor. De processor die net gepolld is, is dus de requesting processor. De requesting processor geeft door aan de controller dat hij wil communiceren. De controller vraagt aan de requesting processor het adres van de destination

processor op. Dit adres noemen we het destination address.
De controller kijkt of beide processoren vrij zijn, d.w.z. zowel de requesting als de destination processor hebben geen datapad lopen naar een andere processor. Als dit het geval is, legt de processor een datapad tussen beide processoren en geeft de requesting processor het mastership en de requesting processor kan dus gaan communiceren. Hierna zal de controller verder pollen.
Is er echter één van de twee of zijn allebei de processoren niet vrij (de requesting processor kan aan een datapad liggen doordat hij voorheen een destination processor was, of vanwege multitasking voorheen berichten stuurde naar een andere processor), dan gaat de controller eerst de datapaden van de processoren verbreken. Nadat allebei de processoren vrij zijn wordt het datapad gelegd en het mastership gegeven aan de requesting processor. Hierna zal de controller verder pollen.
Als de controller een datapad moet verbreken dan verbindt de controller zichzelf met de processor die het mastership over het datapad heeft (als de controller hierna weer verder moet pollen, dan gaat de controller weer terug naar de processor die volgt op de als eerst gepollde (requesting) processor waar hij gebleven was). De controller geeft dan de processor de opdracht de communicatie zo spoedig mogelijk te staken en het mastership te overhandigen. Na het stoppen van de communicatie geeft de processor bij de eerst volgende keer dat de processor gepolld wordt het mastership aan de controller. De controller kan nu het datapad verbreken.

(4) De processor heeft een datapad en heeft niet het mastership van dit datapad en is bezig met ontvangen van berichten over dit datapad. Na melding aan de controller, pollt de controller verder.

(5) De processor heeft een datapad en heeft het mastership van dit datapad en is bezig met het verzenden van berichten over dit datapad. Deze situatie kunnen we nog in twee gevallen onderverdelen.

(5.1) De controller heeft het mastership niet opgevraagd. Na melding aan de controller, pollt de controller verder.

(5.2) De controller heeft het mastership opgevraagd. De processor meldt aan de controller dat hij nog niet klaar is met communiceren. De controller pollt daarna verder.

(6) De processor heeft een datapad en heeft niet het mastership van dit datapad en is klaar met ontvangen van berichten over dit datapad. Na melding aan de controller, pollt de controller verder.

(7) De processor heeft een datapad en heeft het mastership van dit datapad en is klaar met verzenden van berichten over dit datapad. Deze situatie kunnen we eveneens op de zelfde manier onderverdelen als in de vijfde situatie.

(7.1) De controller heeft het mastership niet opgevraagd. De processor meldt dit aan de controller, en de controller pollt verder.

(7.2) De controller heeft het mastership opgevraagd. De processor geeft aan de controller het mastership van het datapad. Omdat de processor het mastership teruggeeft, weet de controller dat er een aanvraag is geweest om dit datapad te breken en de controller zoekt op van welke requesting processor dit verzoek kwam en de bijbehorende destination processor (indien er meerdere requesting processoren zijn, dan neemt de controller de processor die als eerste het request indiende). Indien beide processoren vrij zijn, legt de controller het datapad en geeft de requesting processor het mastership.

De verschillende berichten zijn als volgt gecodeerd in de programmatuur.

A. Messages van de controller naar de processor

poll (BYTE 0): pollen (controller vraagt status processor op)

send (BYTE 1): controller geeft processor het mastership over datapad (processor mag dus gaan communiceren)

break (BYTE 2): controller vraagt processor het mastership op (processor moet zo spoedig mogelijk stoppen met communicatie)

B. Messages van processor naar controller

no.request (BYTE 0): - geen verzoek voor communicatie (situaties 1, 2, 6 of 7.1) òf
- is al bezig met communicatie (situaties 4 of 5.1) òf
- controller heeft mastership opgevraagd, maar communicatie nog niet afgelopen (situatie 5.2)

request (BYTE 1): verzoek voor communicatie, processor geeft nog adres destination processor (BYTE dest) (situatie 3)

broken (BYTE 2): controller krijgt mastership terug (situatie 7.2)

4. Simulaties

Op het TDS zijn simulaties gemaakt van de single crossbar switch. In paragraaf 4.1 wordt het doel van deze simulaties besproken, waarna in 4.2 de uitvoering van de simulaties op het TDS wordt beschreven. De programmatuur van de implementatie van het protocol dat op de processoren wordt gedraaid, staat beschreven in 4.3. Er zijn twee versies van de programmatuur voor de controller gemaakt, deze worden te samen met de bijbehorende resultaten van de simulaties besproken in paragrafen 4.4 en 4.5.

4.1 Doel simulaties

Het doel van de simulaties is het bepalen van de overzendtijden van berichten tussen twee processoren. Met de overzendtijd wordt het tijdsverschil bedoeld tussen het tijdstip dat een processor een bericht wil versturen en het tijdstip dat een andere processor het totale bericht ontvangen heeft. Deze tijdsduur kan men in drie gedeelten opsplitsen. In het eerste gedeelte zit de wachttijd totdat de processor in kwestie gepolld wordt. In het tweede gedeelte zit de tijd dat de controller over het maken van de verbinding (en verbreken van eventuele andere verbindingen) tussen de twee processoren doet. De tijdsduur voor het werkelijke overzenden van bericht zit in het derde gedeelte.

De tijdsduren van deze drie gedeelten zijn afhankelijk van de belasting van de crossbar switch en de berichtlengte. Door simulaties te maken met variërende belastinggraad en berichtlengte kan men bepalen hoe de overzendtijd afhangt als functie van de berichtlengte en de belastinggraad en met welke berichtlengte de overzendtijden minimaal zijn bij variërende belastinggraad.

4.2 Uitvoering simulaties

In deze paragraaf wordt de uitvoering van de simulaties behandeld. In 4.2.1 wordt de opzet van de simulaties besproken en in 4.2.2 de uitvoering van de simulaties op het TDS.

4.2.1 Opzet simulaties

Om aan het doel van de simulaties te voldoen, moeten we een groot genoeg aantal berichten over de crossbar switch zenden met een bepaalde berichtlengte en een bepaalde belastinggraad van het netwerk.

De berichtlengte is een van te voren vastgestelde grootheid. Is een bericht groter dan berichtlengte dan wordt dit bericht opgedeeld in blokken ter lengte berichtlengte. Deze blokken worden sequentieel over de crossbar switch verzonden. Een verbinding kan alleen worden afgebroken indien een processor een geheel aantal blokken ter lengte berichtlengte heeft verzonden.

De belastinggraad van het netwerk wordt in de simulaties bepaald door het aantal processoren dat naar één en dezelfde processor (de ontvanger) wil zenden (sternetwerk). Het zenden naar één processor zorgt ervoor dat voor ieder bericht van een andere processor de crossbar switch de huidige verbinding van de ontvanger moet verbreken en een nieuwe verbinding naar de ontvanger moet leggen. Het aantal processoren zorgt ervoor dat indien er meerdere aanvragen zijn, de processor moet wachten totdat alle vorige aanvragen behandeld zijn.

De maximale belastinggraad voor de single crossbar switch is 30 processoren naar 1 ontvanger. Indien de belastinggraad minder is, blijven er processoren "over" op de switch. Op deze processoren draait een proces dat nooit met andere processoren zal communiceren. De verdeling van de verschillende soorten processoren over de crossbar switch is dan als volgt: op plaats 31 zit de ontvanger; op plaats 0 zit de controller; en op de rest (1-30) zitten de processoren die zenden en de processoren die "over" zijn zo homogeen mogelijk verdeeld.

Voor de tweede versie van de controller zijn er simulaties gedaan met meer dan één cluster. Een cluster is een groep processoren die naar één processor (de ontvanger) zendt met bijbehorende ontvanger. De processoren die zenden van de clusters zijn als volgt over de crossbar verdeeld. De zenders van het eerste cluster worden zo homogeen mogelijk over de crossbar verdeeld. Iedere zender van het tweede cluster komt na iedere zender van het eerste cluster. Zenders van het derde cluster komen weer na zenders van het tweede cluster, enzovoort. De

ontvangers van ieder cluster worden over de nog aanwezige plaatsen verdeeld. Op de overgebleven plaatsen komen dan de processoren die "over" zijn.

De opzet van de simulaties is zo dat de tijdsduur van het overzenden niet gemeten wordt in de normale tijdseenheid (de seconde) maar in zogenaamde kloktikken. Een kloktik is de tijdsduur dat een bericht van één byte nodig heeft om over een transputer link verzonden te worden. Hierdoor worden de simulaties onafhankelijk van de specifieke eigenschappen van de hardware en kunnen we de simulaties onderling vergelijken. Indien men bijvoorbeeld tijdens een simulatie uitvoer naar het scherm zou sturen, dan zal dit vertragend werken op de rest van het programma, waardoor twee simulaties met dezelfde berichtlengte en belastinggraad, doch de één met I/O-uitvoer en de andere niet, niet dezelfde resultaten opleveren.

Het meten van de overzendtijd gaat nu als volgt. Het aantal kloktikken dat de controller nodig heeft om één keer alle processoren in de switch te pollen noemen we een pollcyclus. Een processor die berichten wil verzenden, wacht een random tijd, getrokken uit de uniforme verdeling, genormeerd op de pollcyclus en vraagt dan pas een verbinding aan. Wanneer de processor het gehele bericht, dat bestaat uit één blok van de grootte berichtlengte, verzonden heeft, wacht hij wederom een random tijd voor hij het volgende bericht gaat versturen. De tijdsduur tussen het moment dat het wachten is afgelopen en het moment dat het gehele bericht is overgezonden wordt gemeten en is de overzendtijd. Van de overzendtijden van alle processoren, die berichten hebben verzonden, wordt het gemiddelde, de standaard afwijking en het maximum bepaald en deze drie waarden worden na afloop tezamen met de totale simulatieduur in kloktikken en seconden bewaard. De pollcycli blijken afhankelijk van de berichtlengte en belastinggraad te zijn. In appendix B zijn de pollcycli voor de eerste versie en in appendix C voor de tweede versie van de controller berekend.

4.2.2 Uitvoering simulaties op het TDS

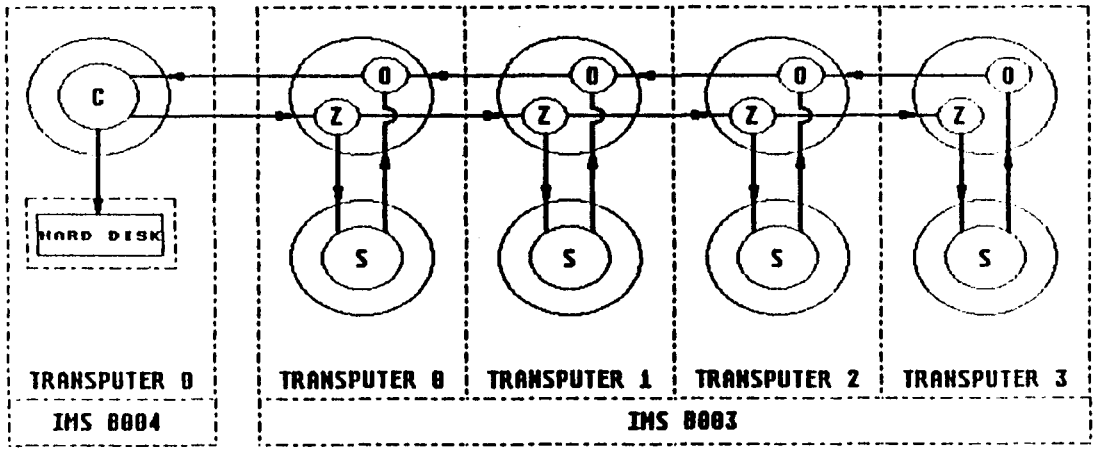
Het in 4.2.1 genoemde overzenden van een aantal berichten over de single crossbar switch met specifieke berichtlengte,

belastinggraad en clusters noemen we een simulatie-opdracht. De specifikaties van alle simulaties die uitgevoerd zijn voor de twee versies van de controller staan in tabel 1.

cv	^{2}log(berichtlengte)	#clusters	#proc./clu.
1	0,1,2,3,4,5,6,7,8,9,10,11,12	1	2,4,8,16,30
2	0,1,2,3,4,5,6,7,8,9,10,11,12	1	2,4,8,16,30
2	0,1,2,3,4,5,6,7,8,9,10	2,3,4,5,6	2
2	0,1,2,3,4,5,6,7,8,9,10	7,8,9,10	2
2	0,1,2,3,4,5,6,7,8,9,10	2,3,4,5,6	4
2	0,1,2,3,4,5,6,7,8,9,10	2,3	8

cv: controller versie
#proc./clu.: aantal processoren per cluster
iedere processor zond 100 berichten naar de ontvanger
#simulaties met controller versie 1: 65
#simulaties met controller versie 2: 241

Tabel 1. Specifikaties van de simulaties.



Figuur 3. Simulatie pipeline op het TDS met de load-balancing methode.
Proces C: controleur simulaties, geven, ontvangen en opslaan van simulaties-opdrachten;
Proces Z: verzenden van simulatie-opdrachten;
Proces O: Ontvangen van simulatie-opdrachten;
Proces S: Uitrekenen van simulatie-opdrachten.

De uitvoering van de simulaties op het TDS staat getekend in figuur 3. Op iedere transputer van het IMS B003 board kan een simulatie-opdracht behandeld worden. Zo is het mogelijk om vier simulatie-opdrachten parallel te verwerken. De controleur van de simulatie-opdrachten (proces C op transputer 0 van het IMS B004 board) voedt een simulatie-opdracht door aan proces Z van transputer 0 van het als pipeline geschakelde IMS B003 board. Dit proces kijkt of de proces S op dezelfde transputer al bezig is

met een simulatie-opdracht. Is de proces S al bezig met een simulatie-opdracht, dan geeft proces Z de simulatie-opdracht door aan proces Z van de volgende transputer, die met de simulatie-opdracht hetzelfde doet als proces Z van transputer 0. Is proces S niet bezig met een simulatie-opdracht, dan geeft proces Z de simulatie-opdracht aan proces S, die de simulatie-opdracht uitwerkt.

Zodra één van de vier processen S een simulatie-opdracht heeft uitgewerkt, geeft proces S de resultaten van de simulatie-opdracht door aan proces 0. Proces 0 zorgt ervoor dat ieder resultaat, ook van vorige transputers richting controleur getransporteerd wordt. Indien proces C een resultaat ontvangt, slaat proces C het resultaat op op de hard disk en geeft aan proces Z van transputer 0 de volgende simulatie-opdracht. De processen Z zorgen er dan voor dat de simulatie-opdracht automatisch terecht komt bij het proces S dat het resultaat had doorgegeven aan de controleur. Deze manier van parallel uitwerken van problemen over meerdere processoren wordt in de literatuur load-balancing genoemd.

4.3 Opzet programmatuur processoren

Voor communicatie tussen processoren onderling zijn er drie processen ontwikkeld op de processoren. Deze processen heten communicate, poll en data. De gebruiker communiceert alleen met communicate. Om een bericht over te zenden naar een processor, geeft hij eerst het commando go.send met het destination address van de processor waar het bericht naar toe gezonden moet worden aan communicate. Hij geeft dan het bericht door aan communicate en na de laatste byte van het bericht geeft hij het commando stop.send.

Communicate communiceert met processen poll en data, die ervoor zorgen dat het bericht uiteindelijk wordt overgezonden. Communicate geeft aan poll de controle data (go.send en stop.send) en aan data het bericht dat verstuurd moet worden. Het bericht wordt door communicate in blokken verdeeld voordat het naar data wordt verstuurd. Deze blokken zijn allemaal van de grootte berichtlengte met uitzondering van het laatste blok, deze mag kleiner zijn. Is bijvoorbeeld berichtlengte 12 bytes groot en

het bericht is 10 bytes groot, dan wordt er één blok van 10 bytes verzonden, is het bericht 25 bytes groot, dan worden er 2 blokken van 12 bytes en één blok van 1 byte verzonden. Het belang van deze blokken is dat een processor een bericht alleen kan afbreken na een geheel aantal blokken.

Poll communiceert met de controller en weet zodoende wanneer er een verbinding ligt en weet dus op ieder tijdstip of er berichten verzonden kunnen worden of niet. Dit geeft poll door aan data en deze kan dan al dan niet blokken bericht van communicate aannemen en versturen. Indien poll van de controller te horen krijgt een bericht zo spoedig mogelijk af te breken, dan geeft poll dit door aan data. Data zal na het overzenden van het blok waar hij nog mee bezig is, het overzenden staken en dit melden aan poll, die dit op zijn beurt weer aan de controller kan duidelijk maken.

Samengevat zorgt proces poll voor de communicatie met de controller, proces data voor communicatie met een andere processor en proces communicate voor communicatie tussen gebruiker en de processen poll en data.

4.4 Simulaties met eerste versie controller

Het protocol is geïmplementeerd op de controller en er zijn simulaties gedraaid. Aan de eerste versie van de controller bleek later met name de fairness verbeterd te kunnen worden. Daarvoor is een tweede versie van de controller gekomen die we in paragraaf 4.5 zullen behandelen. In paragraaf 4.4.1 zal de opzet van de programmatuur van de eerste versie van de controller besproken worden. De resultaten die volgen uit de simulaties zullen in 4.4.2 behandeld worden.

4.4.1 Programmatuur controller (versie 1)

De controller pollt de processoren in de switch sequentieel door zichzelf (de polling link) te verbinden met de processor via de polling crossbar. De controller geeft dan de message poll (BYTE 0), waar de processor op drie mogelijke manieren kan antwoorden:

1. **no.request (BYTE 0).**

   Er zijn dan drie gevallen mogelijk:

   - de processor heeft geen verzoek tot het versturen van een bericht;
   - de processor is al bezig met het versturen van een bericht;
   - de processor heeft opdracht gehad om een bericht zo spoedig mogelijk af te breken, maar het bericht is nog niet afgebroken;

   In alle drie de gevallen is het niet nodig dat de controller actie onderneemt. De controller verbreekt de verbinding van zichzelf naar de processor en pollt verder.

2. **request (BYTE 1).**

   De controller vraagt het adres op van de processor waar de gepollde processor naar toe wil zenden (destination processor). De controller gaat na of de gepollde en/of de destination processor een datapad naar een andere processor hebben liggen. Hebben beide processoren geen datapad dan gaat de controller direct naar fase 2.11 anders gaat de controller naar fase 2.1.

   fase 2.1: Eén of beide processoren hebben een datapad.

   De controller kijkt of de gepollde processor een datapad heeft liggen. Ligt er een datapad dan gaat de controller naar fase 2.2 anders naar fase 2.8.

   fase 2.2: Gepollde processor heeft datapad.

   De controller gaat na wie het mastership van dit datapad heeft en of dit mastership al is opgevraagd. Als het mastership is opgevraagd gaat de controller naar fase 2.3 anders naar fase 2.5.

   fase 2.3: Mastership datapad is opgevraagd.

   De controller kijkt of het verzoek tot opvragen mastership van de gepollde processor kwam. Kwam het verzoek van de gepollde processor dan gaat de controller naar fase 2.4 anders naar fase 2.8.

   fase 2.4: Verzoek van gepollde processor.

   De controller verbreekt de verbinding van zichzelf in de polling crossbar en legt zichzelf aan de processor met het mastership van het datapad en gaat naar fase 2.6.

fase 2.5: Controller nog niet mastership opgevraagd.
De controller verbreekt de verbinding van zichzelf in de polling crossbar en verbindt zichzelf met de processor met het mastership van het datapad. Controller geeft processor de message break (BYTE 2) en de processor probeert het bericht over het datapad zo snel mogelijk af te breken. Controller gaat naar fase 2.6.

fase 2.6: Controller heeft mastership opgevraagd.
De controller geeft de processor (met het mastership van het datapad) de message poll (BYTE 0). De processor kan twee mogelijke antwoorden geven:

- no.request (BYTE 0): bericht niet afgebroken, controller naar fase 2.8;
- broken (BYTE 2): bericht afgebroken, controller naar fase 2.7.

fase 2.7: Het bericht is afgebroken.
De controller verbreekt het datapad en gaat naar fase 2.8.

fase 2.8: Behandelen datapad destination processor.
Controller kijkt of destination processor een datapad heeft liggen. Ligt er een datapad dan gaat de controller naar fase 2.9 anders naar fase 2.10.

fase 2.9: Destination processor heeft datapad.
Herhaal de fasen 2.2 tot en met 2.7 voor het datapad van de destination processor en ga na afloop naar fase 2.10.

fase 2.10: Datapaden beide processoren behandeld.
De controller kijkt of de gepollde en destination processor beide geen datapaden meer hebben. Als dit het geval is dan naar fase 2.11 anders naar fase 2.12.

fase 2.11: Beide processoren hebben geen datapaden.
De controller verbreekt de verbinding met zichzelf in de polling crossbar en verbindt zichzelf met de gepollde processor. De controller maakt ook een datapad tussen de gepollde en de destination processor. De controller geeft nu de gepollde processor het mastership van het datapad door middel van de message send (BYTE 1). De controller gaat naar fase 2.12.

fase 2.12: Eindfase.
De controller verbreekt de verbinding van zichzelf in de polling crossbar en gaat verder met pollen.

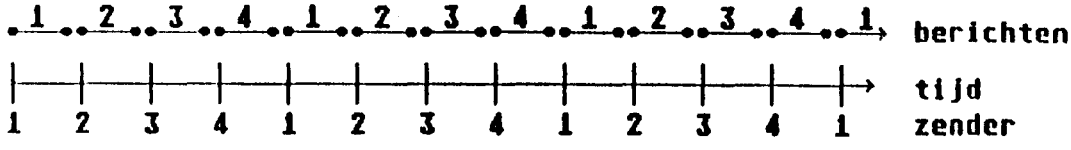
3. **broken (BYTE 2).**

De processor heeft opdracht gehad het mastership van het datapad zo spoedig mogelijk te retourneren, en geeft nu het mastership terug. De controller verbreekt het datapad en gaat na welke processor met bijbehorende destination processor om verbreking datapad verzocht heeft. De controller gaat nu de fasen 2.10 tot en met 2.12 nalopen.

4.4.2 Resultaten simulaties (controller versie 1)

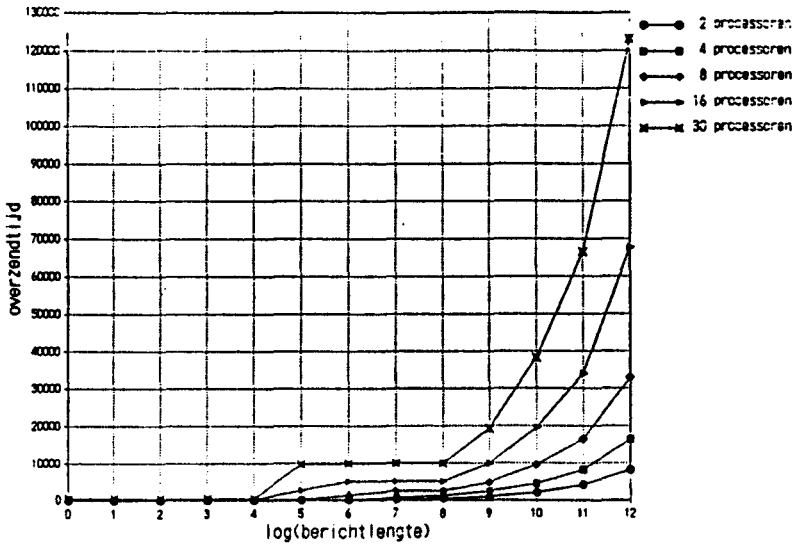
De resultaten van de simulaties met de eerste versie van de controller staan getabelleerd in appendix D. In figuur 5, 6 en 7 zijn de resultaten weergegeven in grafiekvorm. Figuur 5 geeft de overzendtijd (in kloktikken) weer als functie van de logaritme van de berichtlengte (in bytes) voor verschillende waarden van de belastinggraad. Uit figuur 6 kan men het functionele verband tussen de overzendtijd en de berichtlengte halen. In deze grafiek is de logaritme van de overzendtijd (in kloktikken) verticaal weergegeven.

Lopend van links naar rechts door de grafiek in figuur 6 kunnen we vier gebieden onderscheiden. In het eerste gebied blijft de overzendtijd constant tot een bepaalde berichtlengte. Hierna wordt de overzendtijd met een sprong groter. Dit gebied bevat maar één meetpunt. In het derde gebied blijft de overzendtijd constant tot en met een berichtlengte van 2^8 bytes. In het laatste gebied neemt de overzendtijd evenredig toe met de berichtlengte en evenredig met de belastinggraad. Het ontstaan van deze verschillende gebieden zal hieronder verklaard worden.

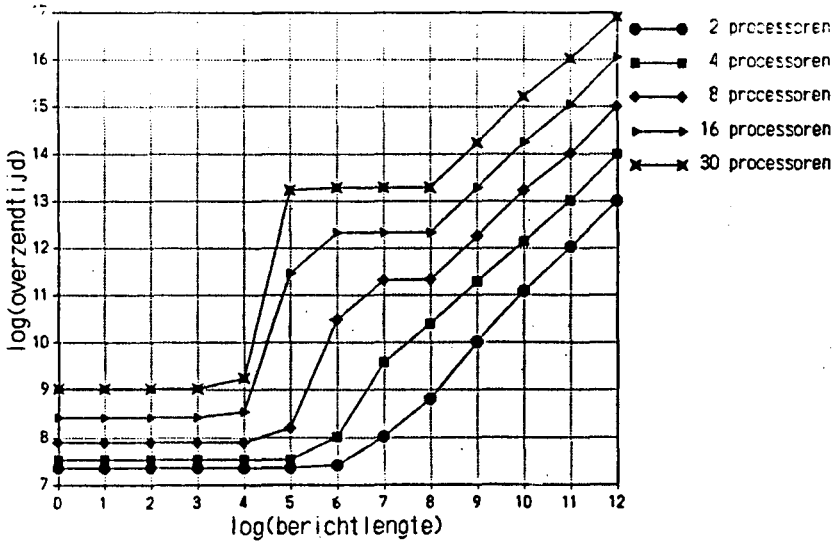


Figuur 4. Voorbeeld overzendstructuur op de data crossbar voor gebied I. In dit voorbeeld zenden vier processoren naar één ontvanger. De maatstreepjes op de tijdschaal geven aan wanneer zender één, twee, drie of vier gepolld wordt. Daarboven staan de berichten met aanduiding van de zender getekend, die op dat moment plaatsvinden.

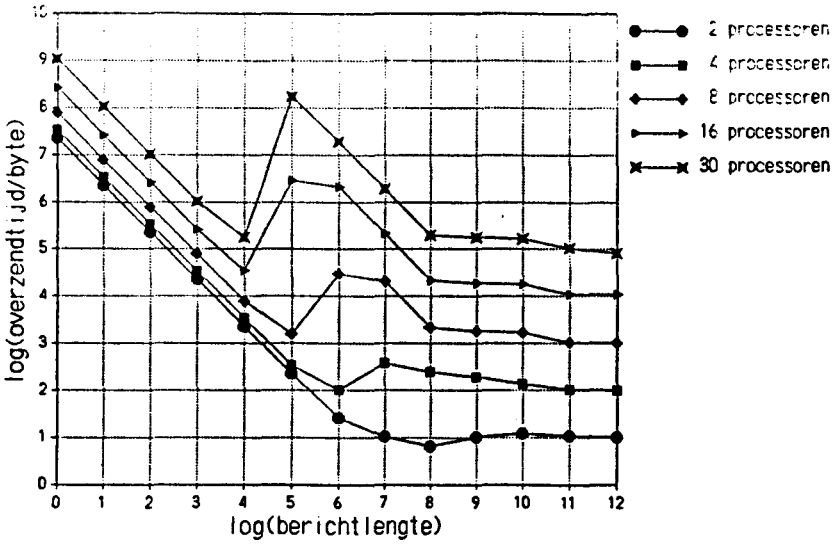
Gebied I. Overzendtijd constant voor kleine berichtlengte. Voor deze kleine berichtlengten geldt dat het bericht al



Figuur 5. Overzendtijden uitgezet tegen logaritme van de berichtlengte (controller versie 1).

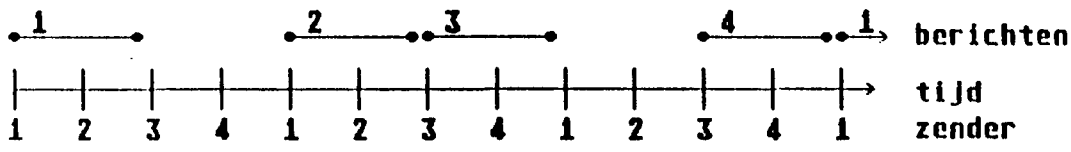


Figuur 6. Logaritme van overzendtijden uitgezet tegen de logaritme van de berichtlengte (controller versie 1).



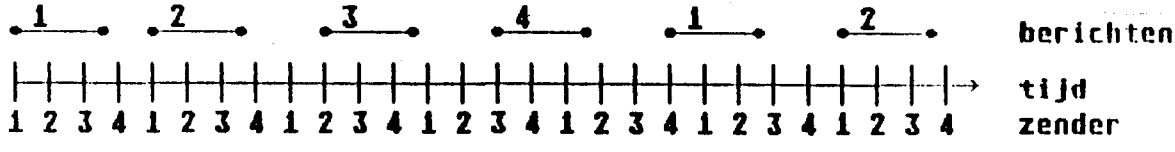
Figuur 7. Logaritme van de overzendtijd/byte uitgezet tegen de logaritme van de berichtlengte (controller versie 1).

afgelopen is voor de volgende processor die wil zenden gepolld wordt, waardoor de volgende zender direct een verbinding kan krijgen. Figuur 4 geeft een weergave van de gebeurtenissen tijdens het pollen. Indien de tijd nodig voor het overzenden van een bericht kleiner is dan de polltijd van 1 naar 2 dan zal voor die berichtlengten de overzendtijd constant blijven.



Figuur 8. Voorbeeld overzendstructuur op de data crossbar voor gebied II. Ook in dit voorbeeld zenden vier processoren naar één ontvanger.

Gebied II. Berichtlengte ligt tussen tijdsduur pollen van twee en drie opeenvolgende processoren. We krijgen een overzendstructuur zoals in figuur 8. Wanneer 2 wordt gepolld, is 1 nog aan het zenden en 2 zal dus moeten wachten. In de volgende pollcyclus wordt 1 gepolld en 1 is klaar met zenden en meldt dit aan de controller. De controller kan nu de verbinding voor 2 leggen en 2 mag gaan zenden. Dan wordt 2 gepolld en die geeft de message no.request. Wanneer 3 gepolld wordt, is 2 al klaar met zenden en 3 krijgt dus direct een verbinding en kan gaan zenden. Als 4 gepolld wordt, is 3 nog niet klaar met zenden en 4 zal dus moeten wachten. De overzendtijd wordt nu groter omdat men nu meerdere pollcycli moet wachten voordat men een verbinding krijgt.



Figuur 9. Voorbeeld overzendstructuur op de data crossbar voor gebied III. Wederom in dit voorbeeld zenden vier processoren naar één ontvanger.

Gebied III. Berichtlengte groter dan de tijd voor pollen van drie opeenvolgende processoren, maar kleiner dan 1 pollcyclus. Zoals in figuur 9 te zien is kan er tijdens een pollcyclus maar één verbinding gemaakt worden, omdat geen enkele processor op tijd met overzenden klaar is. De overzendtijd is weer groter

geworden, omdat er maar één verbinding gemaakt kan worden per pollcyclus. De overzendtijd blijft constant totdat de berichtlengte groter dan één pollcyclus wordt. De pollcyclus is ongeveer 300 kloktikken, dus tot aan een berichtlengte van 2^8 bytes zal de overzendtijd constant blijven.

Gebied IV. Berichtlengte groter dan één pollcyclus. Processoren zullen het aantal pollcycli moeten wachten dat een bericht groot is. De overzendtijd is dus evenredig met de berichtlengte. Ook is de berichtlengte evenredig met de belastinggraad, doordat een processor die als laatste een aanvraag indient, moet wachten totdat alle voorgaande processoren één bericht hebben verzonden.

In appendix B staat precies berekend welke berichtlengten en belastinggraden horen bij ieder van deze vier gebieden. In figuur 6 staat de logaritme van de overzendtijd per byte uitgezet tegen de logaritme van de berichtlengte. De overzendtijd per byte wordt bepaald door de overzendtijd te delen door de berichtlengte. Hieruit kan men voor iedere belastinggraad de waarden voor de berichtlengten halen voor de optimale overzendtijd per byte.

In de grafiek zien we een eerste minimum liggen bij berichtlengten 8, 6, 5, 4 en 4 voor respektievelijk 2, 4, 8, 16 en 30 processoren. Een tweede minimum ligt bij grotere berichtlengte (de curven vertonen een continu dalende lijn voor berichtlengten groter dan 2^{10} bytes). Voor deze berichtlengten is echter de overzendtijd veel te groot (uit figuur 6 op te maken). We moeten dus het eerste optimum hebben. Aangezien in reële omstandigheden de belastinggraad onbekend is, nemen we voor de optimale berichtlengte 2^4 bytes (optimaal voor de grootste belastinggraad).

4.5 Simulaties met de tweede versie van de controller

Voor de implementatie van het protocol op de controller is een tweede versie gekomen die op enkele punten verbeterd werd ten opzichte van de eerste versie van de controller. In paragraaf 4.5.1 wordt de implementatie van de verbeteringen behandeld en in paragraaf 4.5.2 worden de resultaten besproken van de simulaties met de tweede versie van de controller.

4.5.1 Programmatuur controller (versie 2)

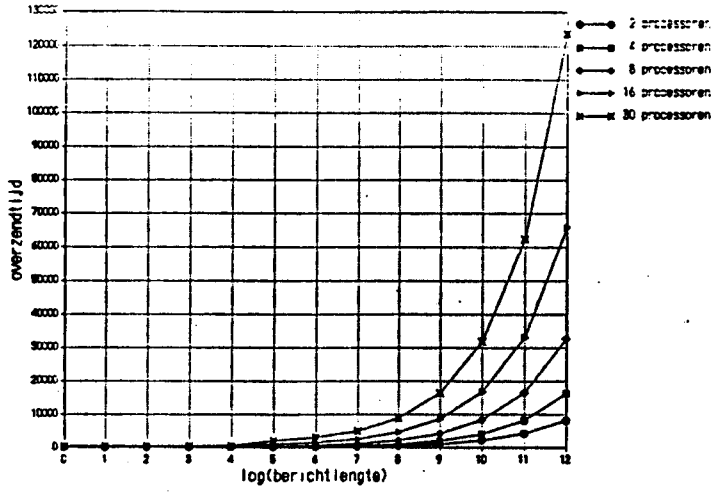
Simulaties en testprogramma's wezen uit dat de implementatie van het protocol op de controller nog verbeterd kon worden. Het eerste punt was de fairness van de controller ten opzichte van de aanvragen van de processoren. Hoewel het onmogelijk is om volledige fairness te garanderen, doordat de processoren niet tegelijk, maar sequentieel afgevraagd worden, was het mogelijk de fairness van de controller te verbeteren. De eerste versie van de controller gaf na het maken van een datapad, de eerste processor die dit datapad opvroeg de voorkeur, terwijl er misschien meerdere processoren voor het leggen van de datapad al een aanvraag voor dit datapad hadden lopen bij de controller.

Het tweede probleem was, zoals uit figuren 7, 8 en 9 blijkt, dat hoewel er voldoende aanvragen voor een datapad zijn, er grote tijdsintervallen zijn dat er geen berichten overgezonden worden. Dit komt doordat de controller maar maximaal twee keer per pollcyclus kijkt of het bericht al afgelopen is.

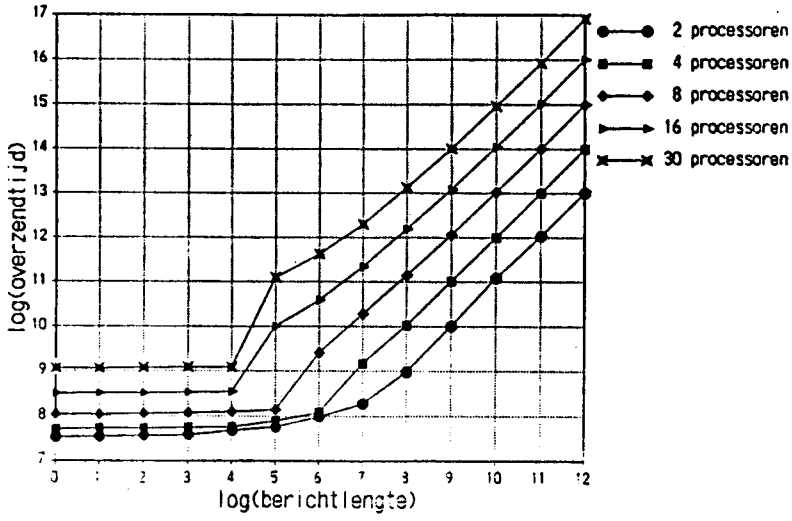
De fairness op de tweede versie van de controller is verbeterd door alle aanvragen voor verbindingen gesorteerd op tijd in een lijst komen (van twee dezelfde aanvragen van de dezelfde processor maar in tijd verschillen wordt alleen de oudste aanvraag bewaard). Na het maken van een verbinding wordt eerste deze lijst raadgeslagen voor de controller verder gaat pollen. De effectiviteit van de crossbar switch werd verbeterd door de controller het tijdstip te laten onthouden wanneer hij het mastership van een datapad opvroeg. Als na de tijdsduur om een blok met de grootte berichtlengte over te zenden plus overhead het mastership nog niet aan de controller teruggeven is, pollt de controller de processor met het mastership. Deze processor zal op dit tijdstip het bericht inmiddels hebben afgebroken en dus het mastership aan de controller teruggeven. Dit tussentijds pollen van een processor om het mastership terug te vragen, kan alleen gedaan worden na afloop pollen van huidige processor door controller.

Deze verbeteringen worden in de tweede versie van de controller gebruikt, die verder op de zelfde wijze werkt als de eerste versie van de controller.

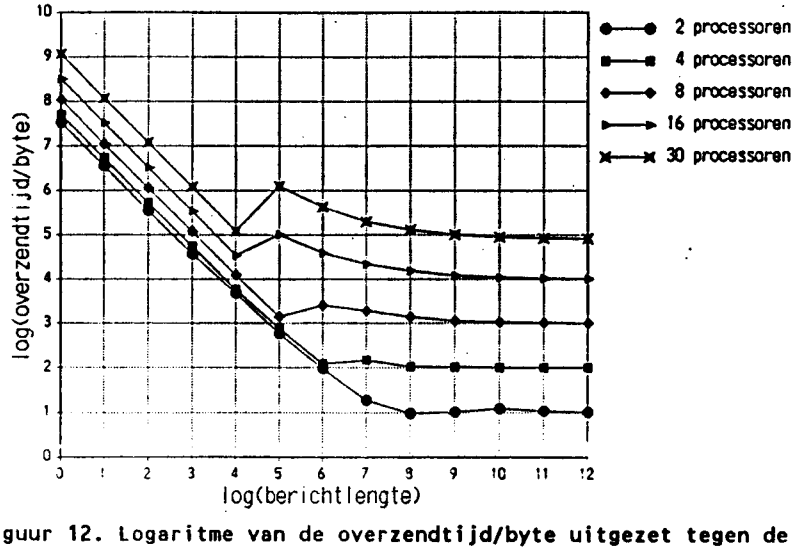
4.5.2 Resultaten simulaties (tweede versie controller)



Figuur 10. Overzendtijden uitgezet tegen logaritme van de berichtlengte (controller versie 2).



Figuur 11. Logaritme van overzendtijden uitgezet tegen de logaritme van de berichtlengte (controller versie 2).



Figuur 12. Logaritme van de overzendtijd/byte uitgezet tegen de logaritme van de berichtlengte (controller versie 2).

De numerieke resultaten van de simulaties met de tweede versie van de controller staan getabelleerd in appendix E. In figuur 10 staat de overzendtijd (in kloktikken) en in figuur 11 de logaritme van de overzendtijd (in kloktikken) uitgezet tegen de logaritme van de berichtlengte (in bytes).

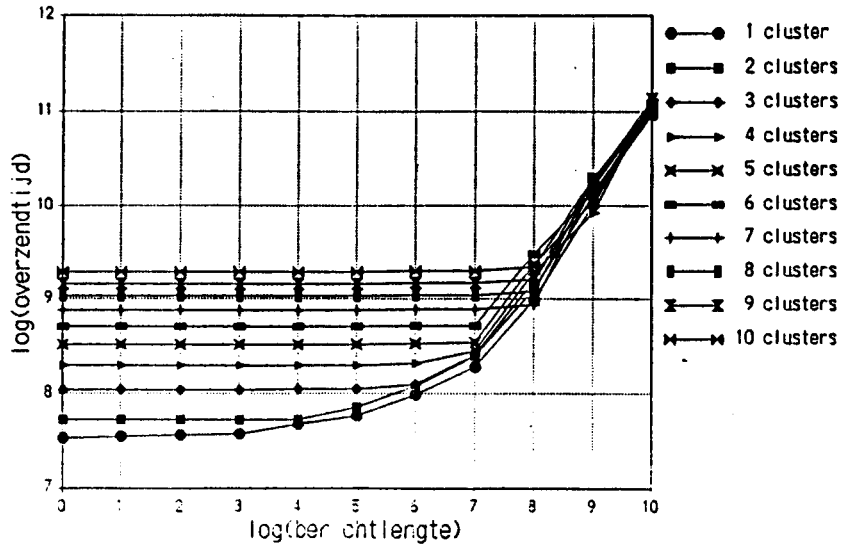
De verbetering van de effectiviteit van de crossbar switch is te in figuur 11. Lopend van links naar rechts zien we nu maar twee gebieden. In het eerste gebied blijft de overzendtijd nagenoeg constant, in het tweede gebied neemt de overzendtijd lineair toe met de berichtlengte en de belastinggraad. Bij de overgang tussen de twee gebieden zien we een sprong in de overzendtijd.

In het eerste gebied geldt nog steeds dat de berichtlengte zo klein is dat het bericht al afgelopen is voordat de volgende zender gepolld wordt. In het tweede gebied is dit niet meer het geval, maar wanneer het bericht is afgelopen (de controller weet dit doordat hij tijdstip heeft onthouden wanneer hij de message break heeft gegeven en dus zeker weet dat na berichtlengte + overhead kloktikken het bericht afgebroken is) ondervraagt de controller de processor met het mastership van het datapad en de controller krijgt het mastership. De controller kan direct het datapad verbreken en een nieuw datapad aanleggen, waardoor de overzendtijd evenredig met de berichtlengte zal zijn. In appendix C staat berekend welke berichtlengte en belastinggraad hoort bij ieder van de twee gebieden.

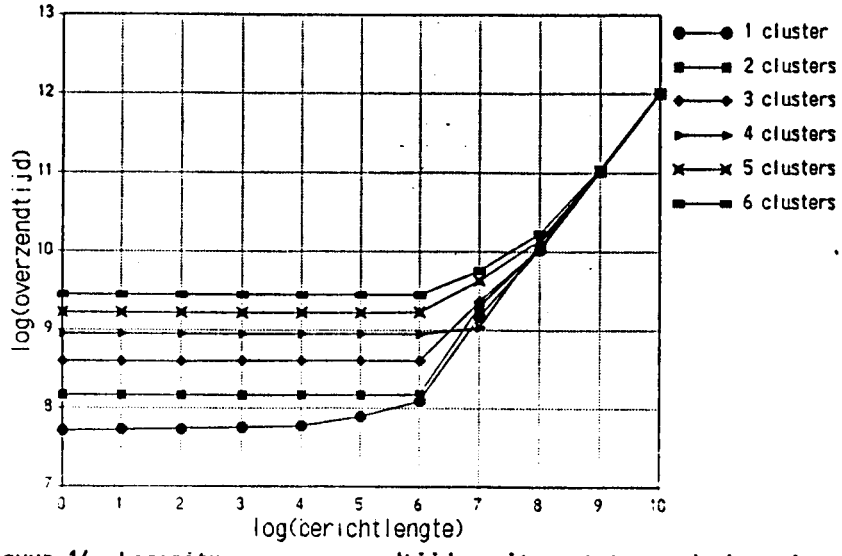
In grafiek 12 staat de logaritme van de overzendtijd per byte weer uitgezet tegen de logaritme van de berichtlengte. Hieruit blijkt dat het optimum voor overzendtijden te vinden is bij een berichtlengte van 2^4 bytes.

Bij de tweede versie van de controller zijn er ook simulaties gedraaid voor meerdere clusters. De resultaten hiervan zijn weergegeven in de figuren 13, 14 en 15. In deze figuren is de logaritme van de overzendtijden (in kloktikken) uitgezet tegen de logaritme van de berichtlengte (in bytes) voor verschillende aantallen clusters bij een belastinggraad van 2 processoren (figuur 13), 4 processoren (figuur 14) en 8 processoren (figuur 15).

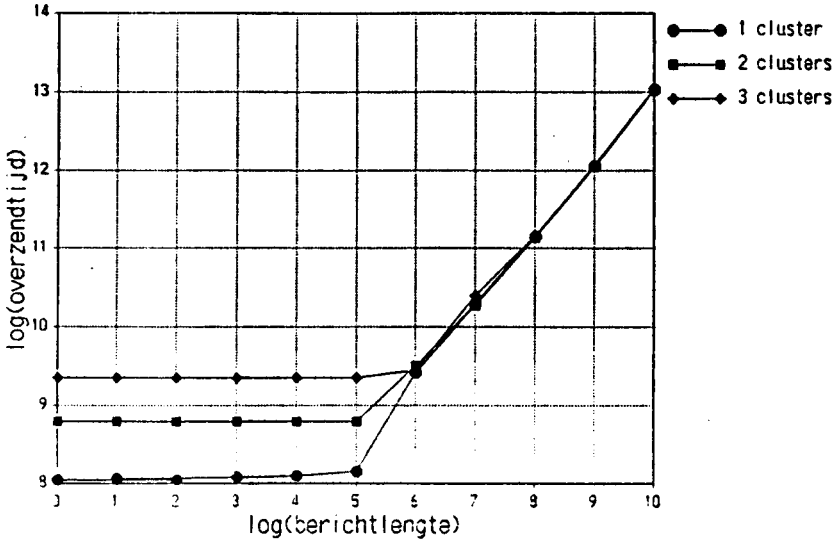
In de figuren is te zien dat er door meer clusters toe te voegen de overzendtijd groter wordt. Dit komt doordat er bij een



Figuur 13. Logaritme van overzendtijden uitgezet tegen logaritme van de berichtlengte voor verschillende aantallen clusters van 2 processoren (controller versie 2).



Figuur 14. Logaritme van overzendtijden uitgezet tegen de logaritme van de berichtlengte voor verschillende aantallen clusters van 4 processoren (controller versie 2).



Figuur 15. Logaritme van overzendtijden uitgezet tegen de logaritme van de berichtlengte voor verschillende aantallen clusters van 8 processoren (controller versie 2).

groter aantal clusters er in totaal meer zenders op de crossbar zitten, waardoor de pollcyclus van de crossbar groter wordt. Hierdoor wordt de overzendtijd groter, want de processoren worden minder frequent gepolld en de kans is dus groter dat de processor langer moet wachten.

Ook blijkt uit de figuren dat er bij sommige berichtlengten door vergroting van het aantal clusters juist een verkleining van de overzendtijd plaatsvindt. Dit vindt plaats bij die berichtlengte, waar de overgang tussen de twee gebieden, nl. overzendtijd ongeveer constant (gebied I) en overzendtijd evenredig met berichtlengte en belastinggraad (gebied II). Door vergroting van het aantal clusters wordt niet alleen de pollcyclus groter, maar ook de tijdsduur nodig voor het pollen van twee opeenvolgende zenders van dezelfde cluster. Hierdoor wordt de berichtlengte, die hoort bij de grens tussen de twee gebieden, groter. En de overzendtijd zal dientengevolge ook langer constant blijven.

5. Conclusies en aanbevelingen

Uit de resultaten van simulaties voor overzenden van berichten tussen processoren op de crossbar switch kunnen we de volgende conclusies trekken:

- Maximale belastinggraad voor de crossbar switch is 30 processoren naar één ontvanger.
- Optimale berichtlengte voor de berichten is 2^4 bytes. Dit levert voor de worst case (maximale belastinggraad) een overzendtijd van 2^5 bytes per overgezonden byte, d.w.z. een reductie in de maximale transmissiesnelheid van de communicatie lijnen met een factor 2^5. Voor een transputer link zijn er twee standaard transmissiesnelheden, namelijk 10 Mbaud en 20 Mbaud (1 baud is 1 bit/s). Een transputer link heeft voor het versturen van een byte (8 bits) twee startbits en één stopbit nodig. In theorie resulteert dit in een overzendsnelheid van 28 kbyte/s voor de 10 Mbaud en 57 kbyte/s voor de 20 Mbaud transputer link voor de worst case. Deze overzendsnelheden kunnen oplopen tot respectievelijk 0.9 Mbyte/s en 1.8 Mbyte/s voor de best case (één processor naar één ontvanger). Deze getallen zijn berekend uit de maximale overzendtijd van 1 byte per overgezonden byte bericht met maximale transmissiesnelheden.
- Het toevoegen van meerdere clusters heeft voor berichten met berichtlengten die in gebied I vallen (voor de tweede versie van de controller) een minder dan lineair effect op de overzendtijd. Voor bijvoorbeeld simulaties met clusters van twee processoren geldt dat als men van 3 naar 8 clusters gaat (factor 2.7) de overzendtijd van 2^8 naar 2^9 kloktikken gaat (factor 2). Naar mate de berichtlengte toeneemt, neemt de invloed van het toevoegen van clusters op de overzendtijd steeds meer af. Voor zeer grote berichtlengte (2^{10} bytes) heeft het toevoegen van clusters zelfs geen invloed op de overzendtijd van een bericht.

Aanbevelingen voor nader onderzoek zijn:

- De datapaden in de crossbar switch worden nu nog unidirectioneel gebruikt, d.w.z. de zender kan alleen naar de ontvanger berichten zenden. Als men deze bidirectioneel zou kunnen gebruiken dan is het niet meer nodig dat de ontvanger aan de controller een datapad moet aanvragen aan de controller voor

eventuele bevestigingen van berichten. Het probleem is dat dan bij verbreken van een datapad de controller dan twee masterships moet terugvragen, één van de ontvanger en één van de zender. Door gebruik te maken van het datapad dat er al ligt, dan kan de controller volstaan met het terugvragen van één mastership.
Wanneer de controller een datapad legt tussen twee processoren dan geeft hij de requesting processor het mastership. De requesting processor geeft na ontvangst van het mastership een message door aan de destination processor dat de destination processor verbonden is met de requesting processor en dat de destination processor berichten kan overzenden naar de requesting processor. De requesting processor kan nu ook berichten overzenden naar de destination processor.
Wanneer de controller het datapad wil verbreken dan vraagt hij het mastership terug van de processor met het mastership van het datapad (dit was de requesting processor). De requesting processor geeft dan de destination processor een message door dat de destination processor zo spoedig mogelijk het zenden van berichten naar de requesting processor moet stoppen. Gelijkertijd zal ook de requesting processor zo snel mogelijk het overzenden staken. Wanneer de destination processor klaar is met overzenden dan geeft hij de requesting processor de message dat het overzenden gestopt is. Wanneer èn de requesting processor gestopt is met zenden èn de requesting processor van de destination processor de message heeft ontvangen dat de destination processor ook gestopt is met oversturen van berichten, dan geeft de destination processor het mastership aan de controller.
Via tokens kunnen de processoren onderscheid maken tussen een bericht en een message.
- Het implementeren van een protocol om meerdere crossbar switches via de communication lines te cascaderen en zo het aantal processoren dat direct met elkaar kan communiceren te vergroten.

Referenties

[1] Transputer Reference Manual,
INMOS Ltd., oktober 1986.

[2] Occam 2 Reference Manual,
C.A.R. Hoare,
Prentice Hall, 1988.

[3] Parallel Processing. Occam and the Transputer,
A. Carling,
Sigma Press, 1988.

[4] IMS B004 IBM PC add-in board,
S. Ghee,
INMOS Ltd. technical note 11, februari 1987.

[5] IMS B003 Design of a multi-transputer board,
A. Vadher, P. Walker,
INMOS Ltd. technical note 10, februari 1987.

[6] Occam program development using the IMS D701 transputer development system,
M. Poole,
INMOS Ltd. technical note 16, januari 1987.

[7] Dynamisch reconfigureerbare multiprocessor systemen,
C.R.P. Wijnen,
afstudeerverslag TUE, FI/FIV 88-13.

[8] IMS C004 programmable link switch,
INMOS Ltd. preliminary datasheet, april 1987.

[9] Design and applications for the IMS C004,
G. Hill,
INMOS Ltd. technical note 19, april 1987.

[10] Integratie van een transputersysteem in een multiprocessor omgeving,
H.J.J.H. Schepers,
stageverslag TUE, FI/FIV 88-16.

Appendix A. Berekening polltijden

In deze appendix worden de polltijden berekend die kunnen optreden bij verschillende antwoorden van processoren op de message poll van de controller.

1. processor vraagt geen datapad aan (n: niets)
   - maken verbinding controller - processor : 4
   - controller geeft message poll : 1
   - processor geeft message no.request : 1
   - verbreken verbinding controller - processor : 3

   totale polltijd: 9 kloktikken. (A.1)

2. processor vraagt datapad aan, maar controller onderneemt geen actie, omdat de controller al bezig is met een eerdere aanvraag naar dezelfde destination processor (vn: vraag-niets) Idem aan A.1 doch in plaats van message no.request geeft de processor de message request + destination address (1 kloktik extra).

   totale polltijd: 10 kloktikken (A.2)

3. processor vraagt en krijgt datapad (vbg: vraag-break-go)
   - maken verbinding controller - processor : 4
   - controller geeft message poll : 1
   - processor geeft message request + dest. address : 2
   - verbreken verbinding controller - processor : 3
   - maken verbinding controller - processor met mastership : 4
   - controller geeft message break : 1
   - controller geeft message poll : 1
   - processor geeft message broken : 1
   - verbreken datapad : 3
   - verbreken verbinding controller - processor : 3
   - maken verbinding controller - polling processor : 4
   - maken datapad : 4
   - controller geeft message send : 1
   - verbreken verbinding controller - processor : 3

   totale polltijd: 35 kloktikken (A.3)
   polltijd tot message poll: 16 kloktikken (A.4)
   polltijd tot message send: 32 kloktikken (A.5)

4. processor vraagt datapad, maar krijgt datapad niet want processor geeft mastership niet terug (vb: vraag-break)
   - maken verbinding controller - processor : 4
   - controller geeft message poll : 1
   - processor geeft message request + dest address : 2
   - verbreken verbinding controller - processor : 3
   - maken verbinding controller - processor met mastership : 4
   - controller geeft message break : 1
   - controller geeft message poll : 1
   - processor geeft message no.request : 1
   - verbreken verbinding controller - processor : 3

   totale polltijd: 20 kloktikken (A.6)

5. processor vraagt datapad aan, heeft dit al eerder gedaan en krijgt datapad (vag: vraag-ask-go)
   Idem A.3 doch geen controller die de message break geeft (1 kloktik minder)

   totale polltijd: 34 kloktikken (A.7)

6. processor vraagt datapad aan, heeft dit al eerder gedaan maar krijgt datapad niet (va: vraag-ask)
   Idem A.6 doch geen controller die de message break geeft (1 kloktik minder)

   totale polltijd: 19 (A.8)

7. processor geeft message broken (bg: broken-go)

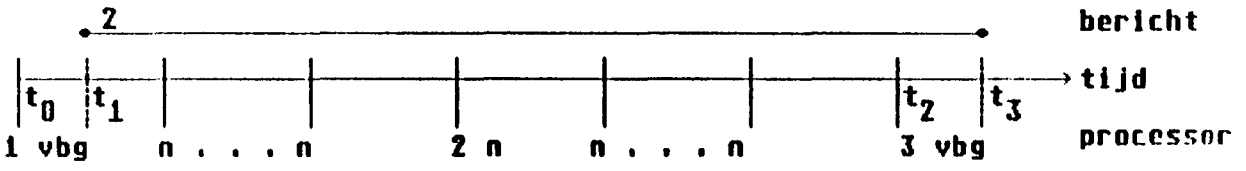
- maken verbinding controller - processor	:	4
- controller geeft message poll	:	1
- processor geeft message broken	:	1
- verbreken datapad	:	3
- verbreken verbinding controller - processor	:	3
- maken verbinding controller - processor die datapad aanvroeg	:	4
- maken datapad	:	4
- controller geeft message send	:	1
- verbreken verbinding controller - processor	:	3

   totale polltijd: 24 kloktikken (A.9)
   polltijd tot message poll: 5 kloktikken (A.10)
   polltijd tot message send: 21 kloktikken (A.11)

8. controller ondervraagt processor die de message break heeft gehad en nu klaar met overzenden zou moeten zijn, hierna maakt controller datapad en geeft mastership aan processor, maar vraagt mastership direct weer op omdat er nog meer aanvragen waren (processor geeft mastership echter pas terug nadat hij een volledig blok heeft overgezonden) (agvb: ask-go-vraag-break, alleen mogelijk met controller versie 2)

- maken verbinding controller - processor	:	4
- controller geeft message poll	:	1
- processor geeft message broken	:	1
- verbreken datapad	:	3
- verbreken verbinding controller - processor	:	3
- maken verbinding controller - processor die datapad aanvroeg	:	4
- maken datapad	:	4
- controller geeft message send	:	1
- controller geeft message break	:	1
- controller geeft message poll	:	1
- processor geeft message no.request	:	1
- verbreken verbinding controller - processor	:	3

   totale polltijd: 27 kloktikken (A.12)



Figuur B4. Maximale berichtlengte voor gebied II. 1, 2 en 3 zijn de zenders, n zijn de processoren die "over" zijn.
t_0: tijdstip dat 1 gepolld wordt;
t_1: tijdstip dat 2 de message send krijgt;
t_2: tijdstip dat 3 gepolld wordt;
t_3: tijdstip dat 2 de message poll krijgt.

Minimale berichtlengte in dit gebied is de maximale berichtlengte van gebied I. Maximale berichtlengte is te halen uit het feit dat berichtlengte kleiner moet zijn dan tijdsduur pollen twee opeenvolgende zenders (zie figuur B4). We krijgen dan als bovengrens voor de berichtlengte:

$$b_{grens} = (t_3-t_1) = (t_2-t_0) - (t_1-t_0) + (t_3-t_2)$$

Aantal processoren dat "over" is tussen drie opeenvolgende zenders bij zo homogeen mogelijke verdeling is ((60 div z) - 2).

t_2-t_0 = 1*[bg] + ((60 div z) - 2)*[n] + 1*[n] = 15 + 9*(60 div z)
t_1-t_0 = 21 (A.11)
t_3-t_2 = 16 (A.4)

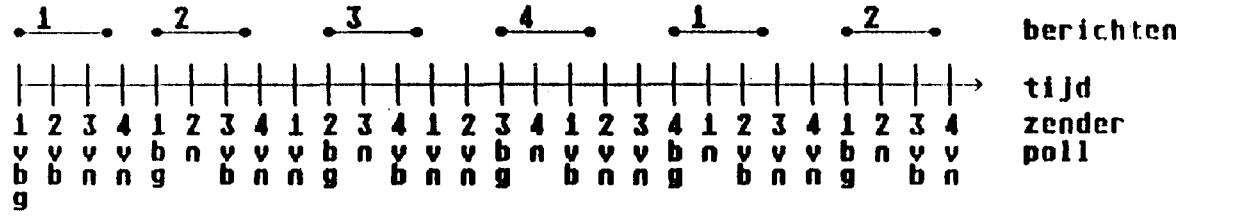
Dit ingevuld levert:

$$b_{grens} = 10 + 9*(60 \text{ div } z)$$

En de volgende grenswaarden voor gebied II:

z	pc	b_{grens}	$^2\log(b_{grens})$ (geheel)	
2	316	280	8	
4	318	145	7	
8	322	73	6	
16	330	37	5	
30	knv	28	4	opm.: knv: komt niet voor

Merk op dat voor z=30 er geen gebied II is, dit komt doordat voor z=30 de maximale bovengrens van gebied I gelijk is aan de maximale bovengrens van gebied II.



Figuur B5. Pollcyclus voor berichten uit gebied III. In dit voorbeeld z=4.

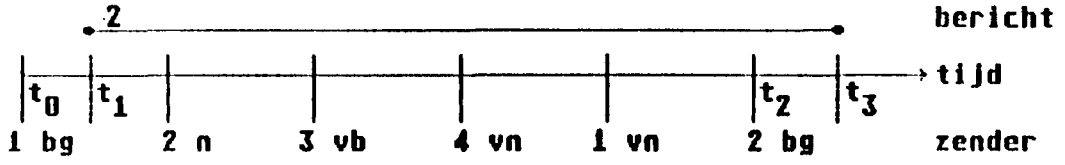
3. Gebied III. Berichtoverzendtijd is kleiner dan één pollcyclus maar groter dan de berichtoverzendtijd uit gebied II. In figuur B5 is te zien dat de data overdracht afhankelijk van de berichtlengte nu kan stoppen tijdens polltijd [vb] of tijdens één van de (z-2) polltijden [vn].

a. data overdracht stopt tijdens polltijd [vb]

$$pc := (z-2)*[vn] + 1*[bg] + 1*[n] + (31-z)*[n]$$
$$= 292 + z$$

b. data overdracht stopt tijdens k-de [vn] ($1 \leq k \leq (z-2)$).

$$pc := (z-2-k)*[vn] + 1*[bg] + 1*[n] + 1*[vb] + (k-1)*[vn] + (31-z)*[n]$$
$$= (z-3)*[vn] + 1*[bg] + 1*[n] + 1*[vb] + (31-z)*[n]$$
$$= 302 + z$$



Figuur B6. Maximale berichtlengte voor gebied III. 1, 2, 3 en 4 zijn de zenders, n zijn de processoren die "over" zijn.
t_0: tijdstip dat 1 gepolld wordt;
t_1: tijdstip dat 2 de message send krijgt;
t_2: tijdstip dat 2 gepolld wordt;
t_3: tijdstip dat 2 de message poll krijgt.

Minimale berichtlengte voor gebied IIIa is de maximale berichtlengte voor gebied II. Maximale berichtlengte is te halen uit het feit dat de data overdracht gestopt moet zijn als de controller bij polltijd [vn] is.

$$b_{grens} = 1*([bg]-(t_1-t_0)) + 1*[n] + 1*[vb] + ((120 \text{ div } z)-3)*[n]$$

(t_1-t_0)=21 (A.11, dit is het gedeelte dat er geen overdracht kan plaatsvinden tijdens polltijd [bg] omdat de zender nog geen message send heeft gehad)

Gemiddeld aantal processoren dat tussen 4 opeenvolgende zenders zit is ((120 div z)-3)

$$b_{grens} = 2 + 9*(120 \text{ div } z)$$

Dit geeft de volgende tabel voor gebied IIIa:

z	pc	b_{grens}	$^2\log(b_{grens})$ (geheel)
2	knv	542	9
4	296	272	8
8	300	137	7
16	308	65	6
30	322	38	5

opm.: knv: komt niet voor

Dat z=2 niet voor komt, blijkt uit de bovengrens voor gebied III. Deze is voor z=2 gelijk aan de bovengrens van gebied II waardoor er voor z=2 helemaal geen gebied III bestaat.

Minimale berichtlengte voor gebied III is weer de maximale berichtlengte van gebied II. De maximale berichtlengte is te halen uit het feit dat de berichtlengte kleiner moet zijn dan een hele pollcyclus plus tijdsduur pollen twee opeenvolgende zenders.

$$b_{grens} = (t_3-t_1) = (t_2-t_0) - (t_1-t_0) + (t_3-t_2)$$

Gemiddeld aantal processoren dat "over" is dat tussen twee opeenvolgende zenders zit is ((30 div z) - 1).

$$t_2-t_0 = 1*[bg] + 1*[n] + 1*[vb] + (z-2)*[vn] + (31-z)*[n] + ((30 \text{ div } z)-1)*[n] = 303 + z + 9*(30 \text{ div } z)$$
$$t_1-t_0 = 21 \text{ (A.11)}$$
$$t_3-t_2 = 5 \text{ (A.10)}$$

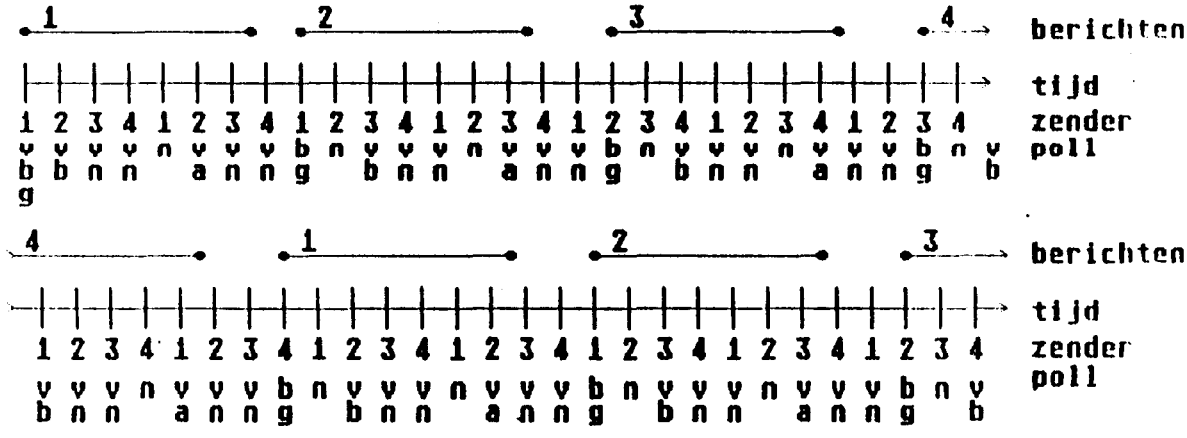
Dit geeft de volgende bovengrens voor de berichtlengte:

$$b_{grens} = 287 + z + 9*(30 \text{ div } z)$$

En de volgende tabel voor gebied III (de polltijden voor gebied IIIb):

z	pc	b_{grens}	$^2\log(b_{grens})$ (geheel)	
2	knv	424	8	opm.: knv: komt niet voor
4	knv	354	8	
8	310	322	8	
16	318	312	8	
30	332	326	8	

Merk op dat voor z=2 er geen gebied III is, dit komt doordat voor z=2 de maximale bovengrens van gebied II gelijk is aan de maximale bovengrens van gebied III. Voor z=4 is bovengrens gebied IIIa gelijk aan ondergrens gebied IIIb en komt dus ook niet voor.



Figuur B7. Pollcycli voor berichten groter dan één pollcyclus. In dit voorbeeld is z=4.

Gebied IV. Berichtoverzendtijd groter dan 1 pollcyclus. Uit figuur B7 vallen de verschillende pollcycli af te lezen.

a. data overdracht stopt tijdens polltijd [va]

pc := (z-2)*[vn] + 1*[bg] + 1*[n] + (31-z)*[n] = 292 + z

b. data overdracht stopt tijdens k-de [vn] ($1 \leq k \leq (z-2)$)

pc := (z-2-k)*[vn] + 1*[bg] + 1*[n] + 1*[vb] + (k-1)*[vn] +
+ (31-z)*[n]
= (z-3)*[vn] + 1*[bg] + 1*[n] + 1*[vb] + (31-z)*[n]
= 302 + z

c. data overdracht stopt tijdens polltijd [bg]

pc := 1*[n] + 1*[vb] + (z-2)*[vn] +(31-z)*[n] = 288 + z

De minimale berichtlengte is de maximale berichtlengte voor gebied III.

Hieronder volgt een tabel met alle in deze appendix berekende gegevens. In de linkerkolom staan de berichtlengten (b). In de horizontale kolommen staan voor vijf waarden van z (aantal zenders) de pollcyclus (pc) uitgezet en tot welk van de vier gebieden de overzendtijd zal behoren.

	2	4	8	16	30
b	pc g	pc g	pc g	pc g	pc g
0	331 I	383 I	487 I	695 I	1059 I
1	331 I	383 I	487 I	695 I	1059 I
2	331 I	383 I	487 I	695 I	1059 I
3	331 I	383 I	487 I	695 I	1059 I
4	331 I	383 I	487 I	695 I	1059 I
5	331 I	383 I	487 I	330 II	322 IIIa
6	331 I	383 I	322 II	308 IIIa	332 IIIb
7	331 I	318 II	300 IIIa	318 IIIb	332 IIIb
8	316 II	296 IIIa	310 IIIb	318 IIIb	332 IIIb
9	304 IV	306 IV	310 IV	318 IV	332 IV
10	304 IV	306 IV	310 IV	318 IV	332 IV
11	304 IV	306 IV	310 IV	318 IV	332 IV
12	304 IV	306 IV	310 IV	318 IV	332 IV

Appendix C. Berekening pollcycli (controller versie 2)

In deze appendix worden de pollcycli voor de tweede versie van de controller berekend. Zoals in paragraaf 4.5.2 al is vermeld, kunnen we de overzendtijden indelen in twee gebieden. In gebied I is de overzendtijd constant als functie van de berichtlengte, in gebied II neemt de overzendtijd lineair toe met de berichtlengte en de belastinggraad.

Gebied I. De overzendtijd is constant als functie van de berichtlengte.
In dit gebied is de berichtlengte zo klein dat het bericht al afgelopen is voordat de controller de volgende zender gepolld heeft. Hierbij horen dezelfde pollcycli als voor gebied I met de eerste versie van de controller zoals berekend in appendix B.

Gebied II. Overzendtijd lineair met berichtlengte en belastinggraad.
Dit gebied laat zich onderverdelen in een gebied II-A en gebied II-B. In gebied II-A is de berichtlengte kleiner dan één pollcyclus, in gebied II-B is de berichtlengte groter dan één pollcyclus. Dit levert verschillende waarden voor de pollcycli.

Gebied II-A. Berichtlengte kleiner dan één pollcyclus.
In dit gebied worden meerdere berichten per pollcyclus overgezonden. De controller weet nu wanneer een bericht is afgelopen, ondervraagt dan de processor met het mastership. Deze geeft het mastership terug aan de controller en de controller verbreekt het datapad, legt een nieuw datapad en geeft de processor de message send. Direct daarna probeert de controller dit datapad te verbreken omdat er meerdere processoren dit datapad willen hebben. De controller ondervraagt de processor nadat deze tijd genoeg heeft gehad om één bericht van de grootte berichtlengte over te zenden plus een overhead tijd van 20 kloktikken.
We hebben n zenders, hiervan is er steeds 1 aan het zenden (polltijd [n]), één heeft een aanvraag bij de controller lopen die in behandeling is genomen (polltijd [va]) en de rest van de zenders vragen een verbinding aan, maar worden niet door de controller behandeld (polltijd [vn]). Normaal zou de pc dus worden:

$$\text{"pc"} := 1*[n] + 1*[va] + (z-2)*[vn] + (31-z)*[n] = 287 + z$$

Tijdens deze pollcyclus pollt de controller nog om te kijken of een bericht afgelopen is (polltijd [agvb]). Het aantal keer dat dit voor komt is de oude pc gedeeld door de berichtlengte plus 20 (overhead). Dit geeft voor iedere keer een extra polltijd van [agvb]. Na enig uitwerken levert dit:

$$pc := 1*[n] + 1*[va] + (z-2)*[vn] + (31-z)*[n] + k*[agvb]$$

met

$$k = \{\ 1*[n] + 1*[va] + (z-2)*[vn] + (31-z)*[n] - 1*[agvb]\ \} \text{ div } (b + 20 - [agvb])$$

Gebied II-B. Berichtlengte groter dan één pollcyclus.
Nu duurt een bericht een aantal pollcycli, waardoor er aan het eind van een bericht de pollcyclus maar één keer een extra polltijd van [agvb] komt. Hierdoor kunnen we voor de pc dezelfde formule gebruiken als in gebied II-B, met voor k=1.

Dit levert de volgende tabel op:

	2	4	8	16	30
b	pc g	pc g	pc g	pc g	pc g
0	331 I	383 I	487 I	695 I	1059 I
1	331 I	383 I	487 I	695 I	1059 I
2	331 I	383 I	487 I	695 I	1059 I
3	331 I	383 I	487 I	695 I	1059 I
4	331 I	383 I	487 I	695 I	1059 I
5	331 I	383 I	487 I	600 II-A	614 II-A
6	331 I	383 I	403 II-A	411 II-A	452 II-A
7	331 I	345 II-A	349 II-A	357 II-A	371 II-A
8	316 II-B	318 II-B	318 II-B	330 II-B	333 II-B
9	316 II-B	318 II-B	318 II-B	330 II-B	333 II-B
10	316 II-B	318 II-B	318 II-B	330 II-B	333 II-B
11	316 II-B	318 II-B	318 II-B	330 II-B	333 II-B
12	316 II-B	318 II-B	318 II-B	330 II-B	333 II-B

Appendix D. Resultaten simulatie (controller versie 1)

#pr.	b.	#ber.	clock.tick	E(x)	S.A.	max(x)	time
2	0	100	33509	164	94	330	70
2	1	100	33509	164	94	330	70
2	3	100	33518	164	94	330	72
2	2	100	33509	164	94	330	71
2	4	100	33518	164	93	330	78
2	5	100	33545	164	91	330	81
2	6	100	34175	171	87	409	88
2	7	100	43121	260	111	566	123
2	8	100	61526	451	90	609	195
2	9	100	119312	1028	96	1187	382
2	10	100	234849	2180	120	2343	759
2	11	100	438149	4208	176	4376	1456
2	12	100	842776	8244	307	8422	2847
4	0	100	39112	186	109	383	82
4	1	100	39112	186	109	383	83
4	2	100	39112	186	109	383	84
4	3	100	39121	186	109	383	86
4	4	100	39130	186	108	383	90
4	5	100	39139	186	107	383	98
4	6	100	46527	257	150	650	130
4	7	100	95033	769	110	945	265
4	8	100	151105	1340	133	1507	450
4	9	100	267468	2496	201	2671	830
4	10	100	472377	4531	329	4718	1531
4	11	100	853667	8321	581	8531	2875
4	12	100	1668419	16418	1133	16679	5684
8	0	100	49538	239	140	494	106
8	1	100	49538	239	140	487	106
8	2	100	49538	239	140	487	109
8	3	100	49547	239	140	487	112
8	4	100	49556	239	140	487	121
8	5	100	55265	296	183	768	149
8	6	100	163241	1443	165	1627	411
8	7	100	277047	2583	259	2767	720
8	8	100	277167	2584	252	2767	854
8	9	100	513130	4922	453	5127	1625
8	10	100	985056	9597	866	9847	3167
8	11	100	1693195	16612	1476	16927	5764
8	12	100	3345177	32977	2921	33447	11465

#pr. : aantal processoren
b. : log(berichtlengte)
#ber. : aantal berichten
clock.tick : simulatieduur in kloktikken
E(x) : gemiddelde overzendtijd
S.A. : standaard afwijking overzendtijd

$$\text{met } (S.A.)^2 = \frac{\Sigma(x^2) - N*(E(x))^2}{N - 1}$$

max(x) : maximum overzendtijd
time : simulatieduur in seconden

#pr.	b.	#ber.	clock.tick	E(x)	S.A.	max(x)	time
16	0	100	70620	346	204	695	152
16	1	100	70620	346	204	695	154
16	2	100	70620	346	204	695	158
16	3	100	70629	346	204	695	166
16	4	100	73559	375	221	983	189
16	5	100	304373	2838	280	3039	717
16	6	100	538609	5174	475	5383	1287
16	7	100	538666	5174	472	5383	1422
16	8	100	538799	5175	464	5383	1692
16	9	100	1023552	9979	877	10231	3273
16	10	100	1993056	19586	1710	19927	6454
16	11	100	3447571	34000	2947	34471	11770
16	12	100	6841094	67627	5856	68407	23451
30	0	100	107458	521	311	1059	234
30	1	100	107458	521	311	1059	238
30	2	100	107458	522	311	1059	245
30	3	100	107493	522	310	1059	261
30	4	100	116396	610	343	1366	312
30	5	100	996268	9701	860	9962	2294
30	6	100	1026998	10005	889	10269	2488
30	7	100	1027295	10007	874	10519	2742
30	8	100	1027425	10008	867	10519	3249
30	9	100	1978355	19431	1670	20346	6356
30	10	100	3880007	38281	3288	39387	12571
30	11	100	6733315	66550	5686	69481	22919
30	12	100	12439455	123094	10493	128443	43707

#pr. : aantal processoren
b. : log(berichtlengte)
#ber. : aantal berichten
clock.tick : simulatieduur in kloktikken
E(x) : gemiddelde overzendtijd
S.A. : standaard afwijking overzendtijd

$$\text{met } (S.A.)^2 = \frac{\Sigma(x^2) - N*(E(x))^2}{N - 1}$$

max(x) : maximum overzendtijd
time : simulatieduur in seconden

Appendix E. Resultaten simulatie (controller versie 2)

#pr.	b.	#clu.	clock.tick	E(x)	S.A.	max(x)	time
2	0	1	36014	185	92	329	65
2	1	1	36528	187	91	329	67
2	2	1	36402	189	91	329	67
2	3	1	36690	191	91	338	68
2	4	1	38599	205	88	345	80
2	5	1	40380	219	86	364	85
2	6	1	43452	255	83	403	96
2	7	1	48492	310	79	473	122
2	8	1	67106	510	86	738	193
2	9	1	119083	1029	94	1253	352
2	10	1	234699	2182	116	2343	702
2	11	1	437438	4204	162	4496	1351
2	12	1	841286	8233	271	8846	2643
2	0	2	41912	213	95	381	76
2	1	2	41912	213	95	381	77
2	2	2	41912	213	95	381	79
2	3	2	41921	213	95	381	80
2	4	2	41921	213	95	381	85
2	5	2	41892	232	98	412	93
2	6	2	44522	270	98	458	116
2	7	2	52588	338	100	525	166
2	8	2	72888	550	89	803	272
2	9	2	137985	1102	95	1284	533
2	10	2	240306	2011	111	2249	1007
2	0	3	47624	264	99	592	87
2	1	3	47624	264	96	433	88
2	2	3	47624	264	96	433	89
2	3	3	47633	264	96	433	94
2	4	3	47633	265	96	433	99
2	5	3	47651	265	96	433	114
2	6	3	47670	274	98	503	140
2	7	3	52835	340	116	572	204
2	8	3	78526	583	97	837	355
2	9	3	146028	1175	95	1330	695
2	10	3	250613	2128	113	2290	1321

#pr. : aantal processoren
b. : log(berichtlengte)
#ber. : aantal berichten
clock.tick : simulatieduur in kloktikken
E(x) : gemiddelde overzendtijd
S.A. : standaard afwijking overzendtijd

$$\text{met } (S.A.)^2 = \frac{\Sigma(x^2) - N*(E(x))^2}{N - 1}$$

max(x) : maximum overzendtijd
time : simulatieduur in seconden

#pr.	b.	#clu.	clock.tick	E(x)	S.A.	max(x)	time
2	0	4	53336	316	98	485	98
2	1	4	53336	316	98	485	99
2	2	4	53336	316	98	485	101
2	3	4	53345	316	97	485	105
2	4	4	53345	316	97	485	115
2	5	4	53363	316	97	485	132
2	6	4	53408	319	95	485	169
2	7	4	53386	349	113	623	239
2	8	4	84178	620	96	870	438
2	9	4	126006	968	97	1346	803
2	10	4	262940	2241	116	2399	1639
2	0	5	59048	367	99	537	108
2	1	5	59048	367	99	537	110
2	2	5	59048	367	99	537	113
2	3	5	59057	367	99	537	119
2	4	5	59057	367	99	537	132
2	5	5	59075	368	98	537	153
2	6	5	59120	370	95	537	197
2	7	5	59147	371	96	589	286
2	8	5	89882	662	90	822	522
2	9	5	135642	1058	94	1407	968
2	10	5	264378	2252	118	2411	1936
2	0	6	64760	419	105	822	119
2	1	6	64760	419	101	589	121
2	2	6	64760	419	101	589	124
2	3	6	64769	419	101	589	131
2	4	6	64769	419	100	589	145
2	5	6	64787	419	100	589	172
2	6	6	64832	422	96	589	226
2	7	6	64850	423	96	621	335
2	8	6	95414	712	92	868	607
2	9	6	141213	1130	94	1285	1130
2	10	6	246689	2066	110	2259	2208

#pr. : aantal processoren
b. : log(berichtlengte)
#ber. : aantal berichten
clock.tick : simulatieduur in kloktikken
E(x) : gemiddelde overzendtijd
S.A. : standaard afwijking overzendtijd

$$\text{met } (S.A.)^2 = \frac{\Sigma(x^2) - N*(E(x))^2}{N - 1}$$

max(x) : maximum overzendtijd
time : simulatieduur in seconden

#pr.	b.	#clu.	clock.tick	E(x)	S.A.	max(x)	time
2	0	7	70472	470	103	641	130
2	1	7	70472	470	103	641	134
2	2	7	70472	470	103	641	136
2	3	7	70481	470	102	641	145
2	4	7	70481	470	102	641	160
2	5	7	70499	470	101	641	192
2	6	7	70544	474	96	641	256
2	7	7	70553	474	96	653	383
2	8	7	73496	491	92	886	642
2	9	7	149088	1201	95	1357	1298
2	10	7	256621	2158	113	2316	2532
2	0	8	76184	521	105	693	141
2	1	8	76184	521	105	693	143
2	2	8	76184	522	105	693	148
2	3	8	76193	522	104	693	157
2	4	8	76193	522	104	693	176
2	5	8	76211	522	103	693	212
2	6	8	76256	525	97	693	285
2	7	8	76256	526	96	693	431
2	8	8	76769	543	92	918	723
2	9	8	152812	1235	96	1391	1457
2	10	8	264274	2248	118	2408	2853
2	0	9	81896	573	107	745	152
2	1	9	81896	573	107	745	154
2	2	9	81896	573	107	745	161
2	3	9	81905	573	107	745	170
2	4	9	81905	573	106	745	191
2	5	9	81923	573	106	745	232
2	6	9	81959	577	97	745	315
2	7	9	81959	578	97	745	479
2	8	9	82463	594	92	950	809
2	9	9	154156	1248	96	1403	1611
2	10	9	246844	2066	109	2382	3124

#pr. : aantal processoren
b. : log(berichtlengte)
#ber. : aantal berichten
clock.tick : simulatieduur in kloktikken
E(x) : gemiddelde overzendtijd
S.A. : standaard afwijking overzendtijd

$$\text{met } (S.A.)^2 = \frac{\Sigma(x^2) - N*(E(x))^2}{N - 1}$$

max(x) : maximum overzendtijd
time : simulatieduur in seconden

#pr.	b.	#clu.	clock.tick	E(x)	S.A.	max(x)	time
2	0	10	87608	624	110	797	163
2	1	10	87608	624	110	797	166
2	2	10	87608	624	109	797	171
2	3	10	87608	624	109	797	183
2	4	10	87608	624	109	797	206
2	5	10	87617	625	108	797	252
2	6	10	87653	629	98	797	344
2	7	10	87662	629	97	797	528
2	8	10	88157	646	92	982	895
2	9	10	155673	1259	99	1432	1765
2	10	10	254227	2157	112	2423	3445
4	0	1	43140	210	106	384	79
4	1	1	42530	212	106	386	78
4	2	1	42530	212	106	386	79
4	3	1	42881	216	105	391	82
4	4	1	43036	219	104	394	86
4	5	1	45460	238	103	418	99
4	6	1	49637	273	96	455	124
4	7	1	76900	575	104	834	208
4	8	1	122656	1049	111	1318	360
4	9	1	225113	2072	119	2267	686
4	10	1	427726	4090	190	4353	1335
4	11	1	839798	8195	373	8397	2646
4	12	1	1658007	16347	727	16868	5269
4	0	2	52913	287	111	486	97
4	1	2	52913	287	111	486	98
4	2	2	52913	287	111	486	101
4	3	2	52922	287	111	486	105
4	4	2	52922	287	111	493	114
4	5	2	52940	287	111	513	131
4	6	2	52976	289	110	519	167
4	7	2	82691	629	111	922	292
4	8	2	134935	1053	105	1428	532
4	9	2	248537	2083	117	2331	1029
4	10	2	472319	4110	190	4376	2019

#pr. : aantal processoren
b. : log(berichtlengte)
#ber. : aantal berichten
clock.tick : simulatieduur in kloktikken
E(x) : gemiddelde overzendtijd
S.A. : standaard afwijking overzendtijd

$$\text{met } (S.A.)^2 = \frac{\Sigma(x^2) - N*(E(x))^2}{N - 1}$$

max(x) : maximum overzendtijd
time : simulatieduur in seconden

#pr.	b.	#clu.	clock.tick	E(x)	S.A.	max(x)	time
4	0	3	64345	389	113	590	118
4	1	3	64345	389	113	590	120
4	2	3	64345	389	113	590	123
4	3	3	64354	390	113	590	132
4	4	3	64354	390	113	590	144
4	5	3	64372	390	112	590	170
4	6	3	64390	391	111	590	224
4	7	3	91948	659	132	1015	383
4	8	3	136518	1063	103	1458	683
4	9	3	248526	2084	120	2369	1328
4	10	3	473541	4117	190	4348	2621
4	0	4	75777	493	115	694	140
4	1	4	75777	493	115	694	142
4	2	4	75777	493	115	694	147
4	3	4	75786	493	115	694	155
4	4	4	75786	493	115	694	174
4	5	4	75804	493	114	701	210
4	6	4	75804	495	112	710	282
4	7	4	79477	524	115	1004	432
4	8	4	138346	1102	107	1398	834
4	9	4	251011	2099	120	2375	1631
4	10	4	475329	4128	192	4329	3224
4	0	5	87200	595	120	798	161
4	1	5	87200	595	120	798	164
4	2	5	87200	595	120	798	170
4	3	5	87209	595	120	798	181
4	4	5	87218	596	117	817	204
4	5	5	87227	597	117	837	250
4	6	5	87227	599	114	846	340
4	7	5	107101	795	121	1139	558
4	8	5	143265	1137	123	1469	994
4	9	5	250072	2096	125	2385	1934
4	10	5	474810	4132	191	4404	3834

#pr. : aantal processoren
b. : log(berichtlengte)
#ber. : aantal berichten
clock.tick : simulatieduur in kloktikken
E(x) : gemiddelde overzendtijd
S.A. : standaard afwijking overzendtijd

$$\text{met } (S.A.)^2 = \frac{\Sigma(x^2) - N*(E(x))^2}{N - 1}$$

max(x) : maximum overzendtijd
time : simulatieduur in seconden

#pr.	b.	#clu.	clock.tick	E(x)	S.A.	max(x)	time
4	0	6	98632	698	124	902	183
4	1	6	98632	698	124	902	186
4	2	6	98632	698	124	902	193
4	3	6	98641	698	124	902	208
4	4	6	98641	698	123	902	235
4	5	6	98641	698	123	902	290
4	6	6	98650	701	118	902	399
4	7	6	115004	865	108	1168	649
4	8	6	149725	1204	128	1617	1159
4	9	6	254146	2118	147	2531	2249
4	10	6	475189	4139	204	4384	4454
8	0	1	52808	265	135	489	97
8	1	1	52808	265	135	489	98
8	2	1	52988	267	134	492	101
8	3	1	53643	270	134	495	107
8	4	1	56027	275	132	505	119
8	5	1	55871	284	129	521	136
8	6	1	90191	680	126	1042	234
8	7	1	144279	1251	123	1607	403
8	8	1	246770	2273	153	2505	731
8	9	1	446325	4259	236	4505	1378
8	10	1	858706	8365	432	8682	2696
8	11	1	1678127	16524	855	16803	5337
8	12	1	3315045	32821	1682	33464	10620
8	0	2	74789	444	143	694	138
8	1	2	74789	444	142	694	140
8	2	2	74789	444	142	694	145
8	3	2	74798	444	142	694	154
8	4	2	74798	444	142	694	171
8	5	2	74816	444	142	694	207
8	6	2	103390	725	148	1166	333
8	7	2	156895	1235	126	1583	577
8	8	2	269739	2260	150	2581	1076
8	9	2	488281	4231	230	4555	2061
8	10	2	944798	8377	437	8652	4068

#pr. : aantal processoren
b. : log(berichtlengte)
#ber. : aantal berichten
clock.tick : simulatieduur in kloktikken
E(x) : gemiddelde overzendtijd
S.A. : standaard afwijking overzendtijd

$$\text{met } (S.A.)^2 = \frac{\Sigma(x^2) - N*(E(x))^2}{N - 1}$$

max(x) : maximum overzendtijd
time : simulatieduur in seconden

#pr.	b.	#clu.	clock.tick	E(x)	S.A.	max(x)	time
8	0	3	97661	651	145	902	181
8	1	3	97661	651	145	902	184
8	2	3	97661	651	145	902	191
8	3	3	97661	651	145	902	204
8	4	3	97661	651	145	902	231
8	5	3	97661	652	145	902	285
8	6	3	100703	699	129	1167	398
8	7	3	169128	1346	199	1736	748
8	8	3	272012	2283	150	2615	1381
8	9	3	494412	4293	236	4641	2677
8	10	3	949184	8429	453	9016	5286
16	0	1	74814	365	194	697	139
16	1	1	75397	366	193	697	143
16	2	1	75510	368	193	697	147
16	3	1	74562	371	191	701	154
16	4	1	76243	376	188	705	174
16	5	1	127251	1028	143	1465	306
16	6	1	177374	1556	140	1916	469
16	7	1	280986	2610	176	2973	802
16	8	1	490923	4700	279	5037	1473
16	9	1	896542	8738	485	9010	2789
16	10	1	1718517	16920	931	17207	5455
16	11	1	3355722	33215	1814	33598	10781
16	12	1	6629007	65794	3591	66424	21428
30	0	1	112355	539	294	1059	212
30	1	1	112321	539	294	1059	216
30	2	1	112150	541	293	1059	224
30	3	1	114153	545	292	1068	244
30	4	1	114463	546	288	1058	276
30	5	1	247958	2216	196	2687	602
30	6	1	340637	3167	221	3607	908
30	7	1	527549	5038	315	5472	1526
30	8	1	912960	8890	509	9180	2783
30	9	1	1675572	16497	930	16782	5295
30	10	1	3220396	31871	1794	32245	10362
30	11	1	6282948	62348	3498	62868	20447
30	12	1	12436398	123586	6936	124400	40752

#pr. : aantal processoren
b. : log(berichtlengte)
#ber. : aantal berichten
clock.tick : simulatieduur in kloktikken
E(x) : gemiddelde overzendtijd
S.A. : standaard afwijking overzendtijd

$$\text{met } (S.A.)^2 = \frac{\Sigma(x^2) - N*(E(x))^2}{N - 1}$$

max(x) : maximum overzendtijd
time : simulatieduur in seconden