Vrije Universiteit, Amsterdam
Faculteit der Exacte Wetenschappen

BWI Werkstuk

Routeren in een netwerk van sensoren

Charlotte Rietveld

Amsterdam, 2007

Voorwoord

Een van de laatste onderdelen van de studie Bedrijfswiskunde en Informatica is het schrijven van een BWI-werkstuk. Hiervoor moet een onderzoek verricht worden over een onderwerp dat binnen twee van de drie disciplines van BWI (Bedrijfskunde, Wiskunde en Informatica) valt.

Voor dit werkstuk heb ik sensornetwerken onderzocht. Met behulp van sensoren kunnen in de netwerken defecten worden gedetecteerd. Als inspiratiebron heb ik Schiphol gebruikt, waar zich een groot netwerk van lampjes, veelal met belangrijke functies, bevindt. De vraag is hoe de informatie van een defecte lamp op een bepaald punt in het netwerk op een effeciënte manier naar een verwerkingsstation kan worden gestuurd zodat de informatie kan worden verwerkt en de defecte lamp kan worden gemaakt.

Ik wil graag Dr. Sandjai Bhulai bedanken voor zijn inzet en enorme enthousiasme tijdens de begeleiding van dit werkstuk. Zijn manier van begeleiden zorgde telkens weer voor een interessante blik op het onderwerp. Mede dankzij zijn inspiratie heb ik met veel plezier aan dit werkstuk gewerkt.

Charlotte Rietveld,

Augustus 2007

Inhoudsopgave

Hoofdstuk 1

Samenvattingen

Nederlands

In dit werkstuk zijn sensornetwerken van lampen bestudeerd waarin defecte lampen kunnen worden gedetecteerd. Wanneer een defecte lamp in het netwerk wordt geconstateerd moet deze informatie op een efficiënte manier naar een basisstation, dat berichtgeving kan doen van de defecte lamp. Om te kunnen realiseren dat de informatie van de detectiebron tot het basisstation kan komen wordt het netwerk zo ingericht dat communicatie tussen lampen die met elkaar verbonden zijn kan plaatsvinden. Iedere lamp bezit hiervoor een bepaalde hoeveelheid geheugen. Met behulp van dit geheugen en de communicatiemogelijkheden kunnen berichten over defecten worden onthouden en doorgegeven. Deze vorm van meldingen zou het onderhoud van grote netwerken kunnen verbeteren en versnellen.

Om te testen welke methode snel werkt en bovendien zoveel mogelijk kapotte lampen bij het basisstation meldt, zijn drie verschillende methodes gesimuleerd en vergeleken. Voor de eerste methode is een variant gekozen waarbij alle berichten naar iedere buur gaan met een bepaalde kans, ook bekend als *roddelen*. Zodra een lamp stuk gaat zend deze een bericht naar al zijn buurlampen die dit op hun beurt weer doorgeven aan hun buurlampen. Wanneer het basisstation wordt bereikt wordt er een melding gemaakt van de kapotte lamp zodat de lamp gerepareerd kan worden.

In de tweede methode wordt er steeds *één buur* uitgekozen om een aantal berichten naar te verzenden. In tegenstelling tot de vorige methode wordt er bij deze methode plaats gemaakt bij de ontvanger. Met een bepaalde kans is de ontvanger de buur die het dichtst bij het basisstation ligt, in andere gevallen wordt een random buur uitgekozen.

Bij de derde methode wordt zeer selectief een buur uitgekozen om berichten naar te versturen. Dit is de buur die het dichtst bij het basisstation ligt. Berichten die vanwege geheugengebrek niet naar deze buur kunnen, gaan naar de buur die daarna de *kortste route* zal vormen met het basisstation. Berichten worden nooit verwijderd en wachten op hun beurt om doorgegeven te worden, tenzij een lamp stuk gaat en er zowel bij de lamp zelf als bij de buren geen plaats is voor het bericht.

Uit de simulaties, waarin testen zijn gedaan met verschillende netwerkstructuren en parameters, blijkt de derde methode verreweg het best te werken. Bij deze manier komen alle berichten aan en bovendien erg snel. Indien bij deze methode een uitzondering wordt gemaakt voor de eerste stap en daarbij een bericht naar alle buren wordt verzonden, werkt

de methode nog beter. Bij de andere methodes doet zich vaak het probleem voor dat er door het verwijderen van berichten bepaalde boodschappen van kapotte lampen uitsterven. Deze lampen zullen nooit gemaakt worden. Ook loopt het netwerk door de vele verspreide berichten snel vol, waardoor de verwachte wachttijd voor een kapotte lamp erg hoog is. De derde methode werkt behalve betrouwbaar en snel ook erg stabiel. Een vergroting van het netwerk bezorgt geen extreem hoge wachttijden. Deze methode levert dan ook de beste ondersteuning bij het onderhouden van een netwerk van lampen.

English

This thesis is about networks of lights in which failing lights can be detected. If such a failing light is detected, this information has to be efficiently transferred to the base-station. Only the base-station can alarm users on a failing light. To be able to transfer information from the source to the base-station, each light can send messages toward other lights that are connected to it. Therefore, every light has got a certain amount of memory to store messages. Using this memory and communication possibilities, lights are able to communicate messages about failing lights over the network. This system of alarming could be beneficial for the maintenance of the network.

To be able to test which methods are fast and reliable (all broken lights should be reported), three different methods are simulated, and examined. In the first method, each message about a broken light is transmitted, with a certain probability, to all neighbouring lights. This is known as the *gossip* method. As soon as a light fails, it sends a message to each of its neighbours, which pass this message again toward their neighbours. If the message arrives at the base-station, the light can be repaired.

In the second method, the message is transmitted toward one of its neighbours. In contrary to the previous method, where all messages that can be transmitted are transmitted, this method transmits a certain amount of messages and overwrites old messages if there is not enough memory available. With a certain probability the message is sent toward the 'best' recipient, otherwist toward a random neighbour.

In the third method, the recipient is chosen carefully. Messages are only sent to the neighbour that is closest to the base-station. Messages that cannot be stored at the 'best' neighbour, will be sent to the second-best neighbour. Messages will never be overwritten, and will wait untill they are passed down, unless there is no other choice.

Simulations, in which experiments are performed with different shapes of the network, and parameters, show that method three is by far the best option. With this method, all messages are received and delivered at the base station within the least amount of time. If the method is extended by gossiping the first message, the performance is even better. This is because if all messages of a certain broken light are removed, these lights will never be fixed. Furthermore, these methods cause the network to be flooded, due to the massive amount of messages, which increases the time needed for a message to reach the base-station. The third method is not only more reliable, and fast, but also very stable. An increase of the network-size does not lead to high waiting-times. This method is therefore the best system for maintaining such networks of lights.

Hoofdstuk 2

Inleiding

Netwerken zijn er op vele plaatsen. Vaak worden deze gebruikt om een object van het ene naar de andere locatie te sturen. Dit kan van alles zijn. Zo kent ieder huis wel een netwerk van gasleidingen. Maar ook vele zoekmachines op internet zijn opgebouwd uit netwerken. In een sensor netwerk kunnen dingen worden gedetecteerd. Bijvoorbeeld een gaslek in een van de leidingen door het huis [1], of dat ene object waarnaar gezocht werd door een zoekmachine.

Voor dit werkstuk zijn netwerken onderzocht waarin een bepaalde boodschap moet worden doorgegeven via de beschikbare paden in het netwerk. Als inspiratiebron heb ik Schiphol gebruikt, waar een netwerk van meer dan 80.000 lampjes zich uitspreidt over het hele complex. Dat dit een grote hoeveelheid lampjes is blijkt alleen al uit het feit dat het er twee keer zoveel zijn als in een grote stad als San Diego, waar ruim 40.000 lampjes onderhouden moeten worden [2]. De lampjes binnen Schiphol worden onder andere gebruikt als communicatiemiddel of signaal voor piloten. Het wegvallen van bepaalde lampen kan voor veel ongemak zorgen. Kapotte lampen moeten tijdig vervangen worden zodat eventueel ongemak beperkt blijft. Een normale procedure voor het onderhoud van de lampjes is dat constant alle lampjes door mensen worden gecontroleerd op hun werking. Zodra er wordt opgemerkt dat er iets kapot is wordt dit doorgeven aan de technische dienst, zodat de lamp gemaakt kan worden. Verder wordt er vaak enig preventief onderzoek naar de staat van de lampjes gedaan.

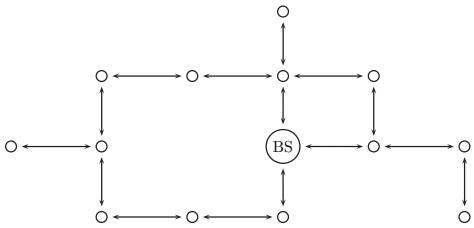
Deze manier van onderhoud is kostbaar en vraagt veel tijd. Bovendien moet een kapotte lamp eerst door iemand worden opgemerkt en worden doorgegeven, hetgeen de betrouwbaarheid en een snelle vervanging niet ten goede komt. Het is dus zaak een kapotte lamp op een andere snelle en betrouwbare manier op te merken. In dit werkstuk zijn verschillende manieren om dit te doen onderzocht. Daarnaast is er een alternatieve manier bedacht die op een efficiënte manier toegepast kan worden op het probleem.

De manieren zijn gebaseerd op boodschappen die kunnen worden doorgegeven via het netwerk van de lampjes. Deze boodschappen kunnen bijvoorbeeld bestaan uit een melding van een kapotte lamp, welke via andere lampjes kan worden doorgegeven aan een basisstation. Vanuit het basisstation kan een melding worden opgesteld voor een reparateur.

Voor het doorgeven van dit soort boodschappen bezit ieder lampje een bepaalde hoeveelheid geheugen. In dit geheugen kan een bepaald aantal boodschappen worden opgeslagen.

Ieder lampje kan indien daar een vrije geheugenplek voor is, een boodschap ontvangen en doorgeven. Op die manier kan een boodschap door het netwerk worden verstuurd.

Het netwerk van lampjes is gesimuleerd in een programma. Hiervoor wordt een rooster gebruikt waarvan de maten kunnen worden opgegeven. Op ieder snijpunt van lijnen van het rooster staat mogelijk een lampje. Deze punten, met lampje, zijn de knopen van het netwerk. Doordat niet op ieder snijpunt een lamp staat, is de infrastructuur van net netwerk niet volledig vast. Wel staat vast dat een knoop maximaal vier *buren* heeft, en alleen verbindingsstukken heeft tussen knopen die rechtstreeks met elkaar verbonden zijn. Er zijn geen andere lege of bezette snijpunten tussen knopen die rechtstreeks verbonden zijn. Deze directe verbindingen vormen de *paden* van het netwerk. Zie figuur 2.1.



Figuur 2.1: Voorbeeld van een netwerk

In dit werkstuk worden de verschillende manieren om boodschappen door het netwerk te sturen met elkaar vergeleken. Welke manier zorgt voor een snelle bezorging van de boodschap bij het basisstation? En welke manier geeft de grootste de kans dat een boodschap aankomt en is daarmee het betrouwbaarst? Is het voor een groot netwerk efficiënt om een bericht over alle buren te verspreiden, of juist niet?

In het komende hoofdstuk wordt eerst het probleem verder omschreven, waarna er verschillende oplossingsmethoden worden geïntroduceerd. In hoofdstuk 5 wordt het simulatieprogramma beschreven waarin verschillende methodes voor het doorgeven van boodschappen gesimuleerd zijn. Vervolgens zullen de experimenten die gedaan worden, beschreven worden in hoofdstuk 6 en de resultaten en bevindingen naar aanleiding van de experimenten in hoofdstuk 7. Naar aanleiding van de resultaten zal een conclusie volgen in hoofdstuk 8. Vervolgens zal hierop nog een discussie volgen in hoofdstuk 9. Referenties zijn te vinden in het laatste hoofdstuk.

Hoofdstuk 3

Probleemomschrijving

Het ontdekken van een kapotte lamp in een groot netwerk door constant alle lampen te controleren, is vaak onbegonnen werk. Het zou handig zijn als bepaalde boodschappen op een efficiënte manier door het netwerk van lampen kunnen worden gestuurd, door de knopen van een geheugen te voorzien. Zodra er in het netwerk een lamp kapot gaat, moet er een signaal via de andere knopen in het netwerk naar een basisstation gaan. Dit doorgeven van het bericht moet op een efficiënte manier gebeuren, zodat een kapotte lamp tijdig kan worden opgemerkt.

Zeker in een groot netwerk is het lastig om zowel snel een bericht naar het basisstation te brengen, als een hoge zekerheid te bieden dat een bericht ook daadwerkelijk bij het basisstation aankomt, omdat de knopen een beperkte hoeveelheid geheugen hebben. Er kan dus niet eindeloos worden gepropageerd met allerlei verschillende berichten dat er een lamp kapot is, aangezien het geheugen dan snel vol zal raken. Bovendien is het niet handig als iedere knoop het bericht aan iedere buurknoop blijft doorgeven, waardoor er berichten eeuwig rond blijven zwerven in het netwerk.

Er kan een van tevoren ingesteld maximum aantal berichten worden onthouden. Zodra het geheugen vol is, zullen er berichten weggegooid moeten worden. De vraag is welk bericht er in dit geval het best kan worden weggegooid en of er wel meer berichten moeten worden doorgegeven dan de ontvangende knoop kan plaatsen. Het weggooien van berichten kan immers tot gevolg hebben dat het nieuwsbericht uitsterft en dus het basisstation nooit zal bereiken. Het uitsterven van berichten moet geminimaliseerd, of als dat mogelijk is voorkomen worden.

De knopen in het netwerk kennen hun locatie niet, evenals de locatie van het basisstation. Wel kennen zij hun buurknopen, dit zijn tevens de enige knopen waarmee gecommuniceerd kan worden. Alleen met deze knopen kan dus informatie worden uitgewisseld of worden doorgegeven. Lampen die kapot gaan voegen deze informatie toe aan de al rondgaande informatie in het geheugen van de knoop. Andere knopen geven alleen informatie door. Door steeds een bericht door te geven kan de boodschap uiteindelijk bij het basisstation terecht komen waar het verwerkt kan worden. Om te voorkomen dat berichten niet meer bij het basisstation kunnen komen doordat de doorgang wordt belemmerd door een kapotte lamp, wordt aangenomen dat de geheugenplekken op plaatsen van kapotte lampen nog wel functioneren. Ook kunnen er via deze plaatsen nog berichten worden doorgegeven.

Een goede methode voor het doorgeven van berichten zou goed in de praktijk kunnen worden toegepast. Bijvoorbeeld in het lampennetwerk van Schiphol, maar ook in het lampennetwerk van een grote stad of bij lantaarnpalen langs de weg, die al snel een netwerk vormen van duizenden lampen.

Hoofdstuk 4

Methodes

Er zijn vele manieren om berichten door het netwerk te sturen. Zo kan iedere knoop een bericht aan iedere buurknoop doorgeven, dit wordt ook wel roddelen genoemd. Een knoop kan ook selectiever zijn met aan wie hij zijn berichten doorgeeft. Ook bestaan er verschillende ideeën over wanneer iets moet worden gewist uit het geheugen. Als namelijk alles altijd zou blijven staan, zou het doorgeven al snel vastlopen doordat het geheugen van iedere knoop vol zit.

Om te bestuderen wat voor methode goed werkt, zijn verschillende methodes met elkaar vergeleken: een methode waarin alle knopen met een bepaalde kans al hun berichten doorgeven aan alle andere knopen, een methode waarbij een van te voren aantal maximum berichten naar één andere buur wordt gegeven, en als laatste een methode die gebruik maakt van kennis van andere knopen. Hoe de methodes geïmplementeerd zijn is te lezen in hoofdstuk 5 – Systeem. Verder zijn er bij het implementeren algemene aannames gedaan. Deze worden eerst besproken, waarop een uitgebreide beschrijving van de methodes volgt.

4.1 Aannames

Allereerst wordt aangenomen dat alle knopen, eventueel via andere knopen, met het basisstation verbonden zijn. Dit is nodig omdat een bericht anders onmogelijk het basisstation kan bereiken. Verder heeft het basisstation genoeg geheugen om alle binnengekomen berichten te kunnen verwerken. Dit zullen er nooit meer per tijdseenheid zijn dan iedere willekeurige andere knoop kan, omdat er maar één knoop per tijdseenheid berichten doorgeeft. Het verwerken gebeurt met een dusdanige snelheid dat de tijd voor het verwerken van een binnengekomen bericht verwaarloosbaar is.

Doorgeeffuncties van knopen gaan niet stuk. Ook niet als de lamp van een bepaalde knoop kapot is. In dat geval blijft het geheugen en de doorgeeffunctie van de knoop gewoon in tact. Om berichten door te kunnen geven moeten knopen weten wie hun buren zijn. Om die reden is er vanuit gegaan dat dit het geval is. Knopen kennen verder geen positie in het netwerk, en ook geen positie van het basisstation.

De tijd die het doorgeven van berichten in beslag neemt, is verwaarloosbaar. Dit is nodig omdat de methodes anders slecht te vergelijken zijn. Verder geven de knopen

periodiek berichten door. Hiervoor is gekozen omdat dit in praktijk ook vaak het geval is. Als laatste wordt er vanuit gegaan dat knopen niet tegelijkertijd berichten naar elkaar willen versturen. Doorgeven gebeurt steeds door één knoop tegelijk. Indien toch ondanks de random trekkingen knopen op precies hetzelfde tijdstip berichten willen doorgeven, worden deze om beurten afgehandeld. Alle genoemde aannames gelden voor alle in dit werkstuk geïmplementeerde methodes.

4.2 Methode 1: Gereduceerd roddelen

Deze eerste methode wordt ook wel 'gossip' genoemd. Het werd eerder door Haas, Halpern en Li [3] toegepast in een ad-hoc netwerk, een 'multi-hop' netwerk zonder vaste infrastructuur. Maar het zou ook in een netwerk dat gebaseerd is op een rooster goed kunnen werken, aangezien er geen vaste infrastructuur vereist is. De werking van de methode is gebaseerd op het roddelen. Hierbij geeft iedere knoop alle berichten door aan alle andere buren. Op deze manier verspreiden berichten zich op een snelle manier door het hele netwerk.

Bij roddelen worden echter vele berichten onnodig rondgestuurd. Haas, Halpern en Li presenteren een methode waarin berichten met een bepaalde kans worden doorgestuurd naar buren. Hiermee wordt het aantal onnodige berichten enorm gereduceerd. Omdat de locatie van de knopen en ook de coordinaten van het basisstation onbekend zijn, moet worden doorgegaan met doorgeven totdat iedere knoop van het netwerk het bericht ontvangen heeft. Op dat moment kan veilig worden aangenomen dat ook het basisstation het bericht heeft ontvangen. Om ervoor te zorgen dat bijna iedere knoop in het netwerk het bericht ontvangt, moet de kans op doorgeven tussen de 0,6 en 0,8 liggen.[3] Hiermee worden er tot 35% minder berichten rondgestuurd dan met standaard roddelen, hetgeen een behoorlijke vermindering is. In de implementatie is dit niet getest, omdat deze gericht is op de kans en de snelheid waarmee het basisstation wordt bereikt. Alleen deze informatie wordt dus opgeslagen, en daarmee niet de fractie knopen die in totaal bereikt is.

Om te voorkomen dat de eerste knoop het bericht niet doorstuurt en daarmee het bericht al direct uitsterft, is in te stellen dat er de eerste keer met kans 1 wordt doorgegeven. Verder wordt de kans op doorgeven aangepast als er erg weinig buren zijn. Wederom omdat de kans dat een bericht uitsterft te groot is als het doorgeven van het bericht van één enkele buur afhangt.

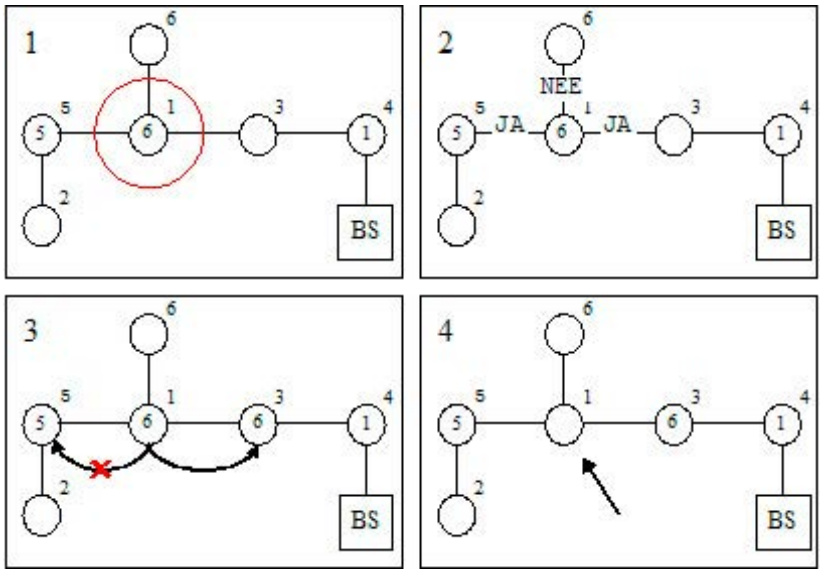
Zodra alle berichten aan alle buren zijn doorgegeven, worden alle berichten uit de actieve knoop gewist. De taak van deze knoop, het aan iedereen doorgeven van het bericht, zit er immers op. Ook als de knoop zijn berichten niet door mocht geven, worden de berichten gewist. Ze hoeven immers niet bewaard te worden als ze niet meer mogen worden doorgegeven. Hierbij ontstaat de kans dat een bericht het basisstation nooit meer zal bereiken omdat het bericht in geen enkel geheugen meer is opgeslagen.

Voor de ontvangende knoop geldt dat als hij reeds berichten in zijn geheugen heeft opgeslagen en niet genoeg plaatsen heeft voor alle nieuwkomende berichten, hij niet alle berichten kan ontvangen. Hij ontvangt er maximaal zoveel als er nog bij past in zijn geheugen. Bij de versturende knoop wordt vervolgens gekeken hoeveel berichten er over blijven om te versturen als alle dubbele berichten (berichten die de ontvanger al in zijn geheugen heeft) eraf worden gehaald. Indien dit er nog steeds teveel zijn, heeft ieder te versturen bericht evenveel kans om een plek bij de ontvanger op te vullen, en worden uiteindelijk alle plaatsen bij de ontvanger gevuld. Zodra met alle buren is gecommuniceerd en zo nodig berichten zijn doorgegeven, worden alle berichten in het geheugen van de versturende knoop gewist. Dit geldt ook voor berichten die aan geen enkele buur doorgegeven zijn.

In figuur 4.1 is schematisch weergegeven hoe de berichten worden doorgegeven. Alle knopen in het netwerk bezitten 1 geheugenplek. Wanneer deze plek bezet is, staat er een cijfer in van de knoop. Dit cijfer geeft aan van welke kapotte lamp het bericht afkomstig is. Ook staan er cijfers naast de knopen. Deze geven de volgorde van doorgeven en het lampnummer aan.

Het eerste plaatje toont het netwerk. Knoop 1 is aan de beurt om door te geven en is aangegeven met een rode cirkel. Allereerst wordt voor alle buren bepaald of hier berichten naar mogen worden doorgegeven. De kans dat dit mag is voor iedere buur gelijk, en is in dit figuur 0,5. In de simulatie worden trekkingen uit de uniforme verdeling gedaan om te bepalen of mag worden doorgegeven of niet. In dit figuur geldt ‘JA’ als ‘er mag worden doorgegeven’, en ‘NEE’ als ‘het mag niet’, zoals in het tweede plaatje te zien is. Bij het derde plaatje wordt er gecommuniceerd tussen de knopen. Hierbij wordt bekeken of er genoeg vrije geheugenplaatsen zijn voor het aantal te versturen berichten, in dit geval één. Het blijkt dat knoop 5 geen plaats meer heeft, maar knoop 3 een bericht kan ontvangen. Het bericht wordt verstuurd naar knoop 3. Nu met alle buurknopen waarmee gecommuniceerd moest worden gecommuniceerd is, en indien mogelijk berichten zijn doorgegeven, is het doorgeven voltooid. De actie eindigt met het leegmaken van het eigen geheugen, zoals in plaatje 4 te zien is. De volgende knoop die berichten door kan geven is knoop 2, die zich links onder bevind. Het doorgeven gebeurt op dezelfde manier als bij knoop 1.

Zodra dus aan alle buren berichten zijn doorgegeven indien dit moest en alle berichten uit de versturende knoop zijn gewist, is het doorgeven voltooid. Alle knopen geven op deze manier, in een vooraf bepaalde willekeurige volgorde, en met vaste tussentijden, berichten door. Doordat de tussentijden vast zijn verandert de volgorde van knopen die berichten door mogen geven niet gedurende een simulatie. Telkens als een knoop zijn berichten mag doorgeven, herhaalt zich het doorgeefproces vanaf het begin.



Figuur 4.1: Methode 1 - Gereduceerd roddelen

4.3 Methode 2: Enkele berichten naar een buur

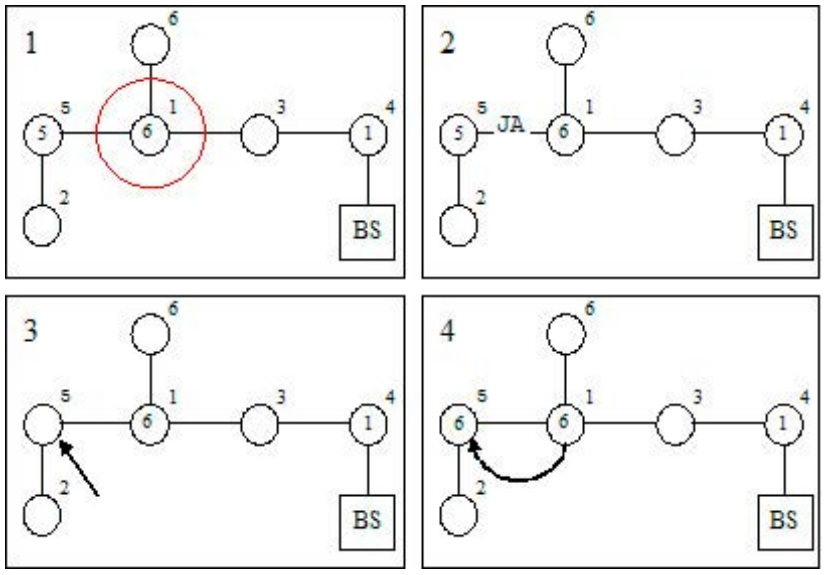
Ook deze methode is gebaseerd op een roddeltechniek ofwel Gossip. Op zich is dit niet zo vreemd, want op Gossip gebaseerde technieken lijken een robuuste, efficiënte en schaalbare oplossing te zijn voor het verspreiden van informatie in peer-to-peer netwerken [2], [3], [4]. Ook deze methode bestaat uit simpele, onafhankelijke one-to-one interacties.

Bij deze methode wordt er bij het doorgeven een bepaalde buur uitgekozen. Met een bepaalde kans is dit de beste buur (zie methode 3), anders wordt een random buur als ontvanger gekozen. Naar de gekozen buur worden een van tevoren ingesteld aantal berichten gestuurd, of als de versturende knoop minder berichten in het geheugen heeft alle berichten. Ook hier worden berichten die bij de buur reeds bekend in het geheugen zijn opgeslagen niet meegeteld en dus ook niet verstuurd. In tegenstelling tot de vorige methode worden berichten niet verwijderd na het doorsturen, maar wordt er bij de *ontvanger* plaats gemaakt voor de verstuurde berichten. Van de berichten die de ontvanger in zijn geheugen heeft moeten er dus genoeg verwijderd worden zodat alles ontvangen kan worden. Bij het verwijderen worden de oudste berichten als eerst verwijderd, omdat de kans voor deze berichten dat ze al een keer zijn doorgegeven het grootst is.

Zodra er plaats is gemaakt voor de nieuwe berichten worden de berichten verstuurd. Als er meer berichten in het geheugen zijn opgeslagen dan er verstuurd mogen worden, exclusief dubbelen, worden er random trekkingen gedaan om een aantal berichten uit te kiezen die verstuurd worden. Ieder bericht heeft evenveel kans om bij de te versturen berichten te horen. Nadat de berichten verstuurd zijn is het doorgeven voltooid. Zo lang berichten dus niet verwijderd worden, kunnen zij de volgende ronde weer naar een andere buur verzonden worden.

Ook bij deze methode geldt dat knopen in een vooraf bepaalde willekeurige maar vaste volgorde berichten door mogen geven. Het doorgeven is geïllustreerd in figuur 4.2.

Wederom bevatten alle knopen 1 geheugenplek. De buur die wordt uitgekozen om mee te communiceren wordt aangeduid met 'JA'. Te zien is dat bij deze methode eerst plaats wordt gemaakt als dit nodig is, alvorens berichten door te sturen, en dat aan het eind van de actie het eigen geheugen niet wordt leeggemaakt. Ook hier is knoop 2 de volgende die berichten kan versturen.



Figuur 4.2: Methode 2 - Enkele berichten naar één buur

4.4 Methode 3: De kortste weg

Bij deze methode wordt in tegenstelling tot de andere methodes, gebruik gemaakt van bepaalde voorkennis. Deze voorkennis wordt opgebouwd door wanneer het systeem begint, eerst vanuit het basisstation voor iedere knoop te bepalen hoeveel stappen er nodig zijn om bij het basisstation te komen. Het idee is gebaseerd op het algoritme van Dijkstra voor de kortste weg [5]. Hierbij wordt voor een beslissing vanaf het startpunt om een bepaalde kant op te lopen gekeken naar de direct toegankelijke buurknopen. Deze knopen geven aan hoe lang de kortste weg is vanaf deze knopen. Op basis van deze kennis en kennis van hoe ver de buurknopen bij het startpunt vandaan liggen, kan berekend worden hoe lang de kortste weg zal zijn vanaf het startpunt en welke route daarbij hoort.

Het voordeel van deze methode is dat er geen kennis over de positie van de knopen voor nodig is. Knopen weten simpelweg wie hun buren zijn, en hoeveel stappen er vanaf deze buren nog nodig zijn om bij het eindpunt te komen. Op basis van deze informatie hoeven berichten niet eindeloos rond te zwerven in het netwerk, maar wordt er bewust voor een bepaalde weg gekozen, waarbij iedere doorgeefactie het bericht dichter bij het basisstation brengt.

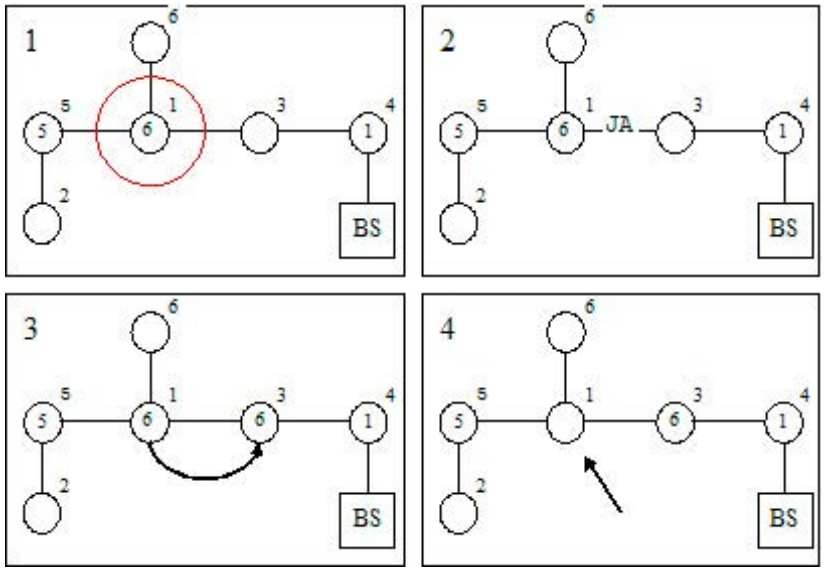
Ook bij deze methode kan het voorkomen dat de ontvangende knoop niet genoeg plek heeft om alle berichten te ontvangen die verstuurd kunnen worden. In dit geval worden er zoveel mogelijk berichten verstuurd. Berichten die niet verstuurd konden worden, worden naar een andere buur gestuurd. Ieder bericht gaat dus naar maximaal 1 buurknoop.

Wanneer berichten niet dichter bij het basisstation kunnen komen door doorgegeven te worden, kunnen er twee dingen gebeuren, die van tevoren kunnen worden ingesteld:

- Er kan voor worden gekozen om in dat geval niets te doen, omdat de situatie er in principe niet op vooruit gaat als het bericht een stap 'terug' gaat. Tijdens de volgende doorgeefronde wordt het bericht dan namelijk toch weer terug geplaatst naar de knoop waar hij vandaan kwam, hetgeen weinig zin heeft.
- Er kan ook voor worden gekozen om wel toe te staan dat een bericht achteruit doorgegeven wordt. Dit zou kunnen werken als er op een bepaalde manier ergens in het netwerk een bepaalde knoop een knelpunt zou vormen. Dit kan het geval zijn wanneer er veel lampen tegelijkertijd in een bepaald gebied kapot gaan en de kortste route van deze lampen altijd via dit een bepaald knelpunt in het netwerk gaat. Het resultaat is vaak te zien in het verkeer: wanneer veel mensen tegelijkertijd via een bepaalde kortste route willen reizen, ontstaat er een file. Deze file zal zich uiteindelijk oplossen: zelfs wanneer alle berichten door een bepaald punt moeten, komt er een moment dat alle lampen stuk zijn en geen nieuwe berichten meer zullen uitzenden. Dit zou de wachttijd echter niet ten goede komen. In dit geval zou het misschien niet altijd slecht werken als er met een bepaalde kans berichten een andere kant op worden gestuurd dan voor hen het kortst zou zijn. Op deze manier zou een alternatieve route ontdekt kunnen worden.

Het verschil met methode twee is dat bij deze methode voor ieder bericht een kortste weg wordt gezocht. Berichten worden niet verwijderd als ze nog niet zijn doorgegeven, en wachten op hun beurt als dit nodig is. Dat moet ook, want van ieder bericht is er maar één in het netwerk aanwezig. Hierdoor moet dit bericht ook aankomen. Dit wordt vergemakkelijkt doordat er geen berichten een verkeerde kant op gaan en het aantal berichten in het netwerk wordt gereduceerd tot het kleinst mogelijke.

Wanneer een bericht succesvol wordt doorgegeven, wordt het bericht bij de versturende knoop verwijderd. Het bericht is immers een stap dichterbij het basisstation en een tweede bericht bij de versturende knoop voegt niets meer toe. Ter illustratie is het verloop van een doorgeefactie van een knoop weergegeven in figuur 4.3. Ook in dit figuur is knoop 1 aan de beurt om berichten door te geven. Er wordt bekeken welke knoop het dichtst bij het basisstation ligt, dat is in dit geval knoop 3. Vervolgens wordt met deze knoop gecommuniceerd of deze plaats heeft voor het aantal berichten dat verstuurd moet worden, in dit geval één. Indien dit niet mogelijk was geweest, zou hiermee de doorgeefactie zijn beëindigd. Te zien is dat er in dit voorbeeld wel een vrije geheugenplek is, waardoor het bericht kan worden doorgegeven. Vervolgens wordt het bericht uit het eigen geheugen gewist, en wordt daarmee de actie beëindigd.



Figuur 4.3: Methode 3 - De kortste weg

Hoofdstuk 5

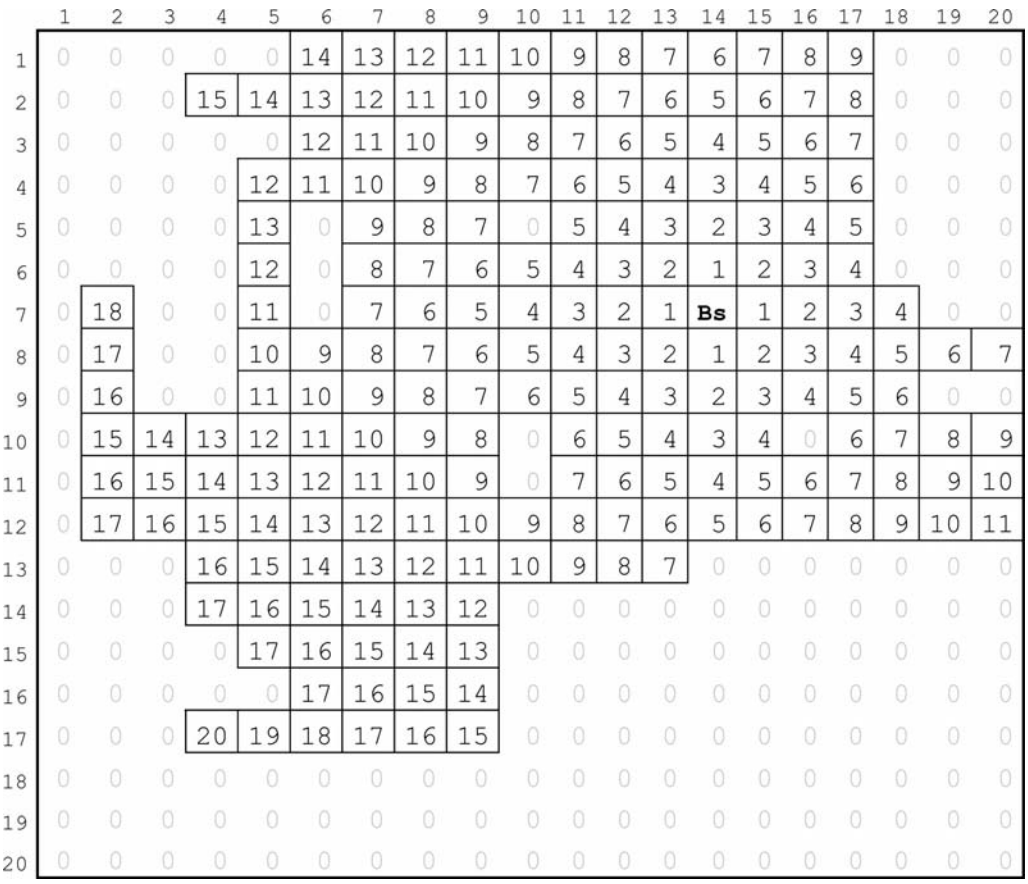
Systeem

Om het doorgeven van berichten te simuleren is er een programma geschreven in C++. De opbouw in grote lijnen van dit programma wordt in dit hoofdstuk beschreven.

Constanten
Allereerst is er een aantal constanten waarmee gewerkt wordt. Deze kunnen per simulatie veranderd worden. De belangrijkste constanten zijn waarden voor de afmetingen van het rooster, het aantal runs dat gedraaid wordt, de simulatietijd, het aantal geheugenplaatsen, de parameter lambda voor de tijden tussen twee lampen die opeenvolgend stuk gaan, verschillende kansen voor het doorgeven, de maximale tijd dat er geen lamp kapot gaat en de frequentie van doorgeven door middel van het aantal tussenliggende seconden op te geven. Verder kan worden opgegeven of in de eerste stap berichten naar alle buren moeten en of andere doorgeef acties naar alle buren gaan of niet (optie *gossipEersteStap* en optie *gossip*).

Rooster
Zodra het programma opstart wordt er een rooster opgebouwd en worden er zogenaamde lampen op verschillende coördinaten gezet. Dit gebeurt met een algoritme waarin vanuit het basisstation, dat random wordt gekozen, een pad wordt gevormd. Bij iedere stap worden van een bepaalde knoop de buren bepaald en wordt vervolgens op recursieve wijze het netwerk uitgebreid. Op deze manier kunnen er geen eilanden van knopen ontstaan (knopen die op geen enkele wijze verbonden kunnen worden met het basisstation) en zit alles aan elkaar vast. Wanneer het netwerk gevormd is, wordt voor alle bestaande knopen bepaald hoeveel stappen er nodig zijn om bij het basisstation te komen. Dit is nodig om de beste buur te kunnen bepalen. Het gevormde netwerk kan eruit zien als in figuur 5.1.

Lijst met gebeurtenissen
De simulatie zelf werkt met een lijst waarin allerlei gebeurtenissen worden opgeslagen. De gebeurtenissen staan in volgorde van het tijdstip waarop zij plaats vinden. Er zijn twee soorten gebeurtenissen: een lamp die kapot gaat en een knoop die berichten gaat doorgeven. Bij het initiëren van het programma wordt er voor alle knopen een doorgeefactie ingepland, vervolgens wordt er steeds een nieuwe doorgeefactie voor een bepaalde knoop ingepland zodra er door die knoop moet worden doorgegeven. De tussentijden voor het doorgeven zijn voor alle knopen hetzelfde. Deze is van tevoren ingesteld. Voor het kapot gaan van lampen gebeurt iets soortgelijks: er wordt bij het initiëren ingepland



Figuur 5.1: Een netwerk, met getallen die de afstanden tot het basisstation voorstellen

wanneer de eerste lamp stuk gaat en vervolgens wordt er steeds zodra er een lamp stuk gaat een nieuwe stukgaande lamp ingepland.

Gebeurtenis kapotte lamp

Zodra de gebeurtenis van een kapotte lamp zich voordoet en deze lamp al stuk blijkt te zijn, gebeurt er niets. Wanneer de lamp nog niet stuk is, wordt de simulatietijd opgeslagen (om zo later terug te kunnen zoeken wat de wachttijd van de lamp was). Wanneer er voor een lamp een tijd is opgeslagen geeft dit direct aan dat de lamp stuk is. Er wordt een nieuwe lamp (random) getrokken die stuk gaat en deze wordt ingepland. Afhankelijk van of *gossipNaarAlleBuren* aanstaat gaat er vervolgens een bericht naar één of naar alle buren met de coördinaten van de lamp en de tijd dat deze kapot ging als informatie. Afhankelijk van het algoritme dat gebruikt wordt kan de boodschap mogelijk niet bij alle buren, of zelfs bij geen enkele buur worden opgeslagen vanwege een vol geheugen. In dit geval wordt het bericht bij de knoop van de kapotte lamp opgeslagen. Indien ook hier alle geheugenplaatsen vol zijn wordt er bij de knoop van de kapotte lamp plaats gemaakt door een bericht te wissen.

Gebeurtenis doorgeven

Wanneer de gebeurtenis *doorgeven* optreedt, worden er berichten doorgegeven. Afhankelijk van de methode gebeurt dit op een bepaalde manier. Er wordt zo nodig gecontroleerd op dubbele berichten, een bericht verwijderd, of een te versturen bericht geselecteerd.

Wanneer de knoop waaraan berichten worden doorgegeven het basisstation is, wordt de lamp waar het bericht vandaan kwam gemaakt en de informatie, zoals wachttijd, verwerkt. In de simulatie wordt aangenomen dat de tijd die nodig is om berichten te verwerken verwaarloosbaar is.

Opwarmperiode

De simulatie is opgedeeld in meerdere runs. Doordat er gebruik wordt gemaakt van meerdere runs kan een betrouwbaarheidsinterval voor de resultaten worden opgesteld. Zolang de vooraf ingestelde tijd voor een run niet is verstreken, loopt de simulatie door. Wanneer de tijd van een run is verstreken worden er wel tussenresultaten opgemaakt en worden er statistieken verzameld, maar de status van de lampjes blijft hetzelfde. Dit is nodig omdat anders iedere run zou beginnen met allemaal lampen die heel zijn en hierdoor de resultaten worden beïnvloedt. Door de simulatie te starten met een run die niet meegeteld wordt in de resultaten, wordt een beginsituatie gecreëerd die wel realistisch is.

Om resultaten op te maken die over de totale simulatie gaan, zoals de berekening van het totaal aantal lampen dat stuk ging en het totaal aantal lampen dat gemaakt werd, wordt na de laatste run de simulatie nog enige tijd langer door gedraaid. Er komen dan geen nieuwe kapotte lampen meer bij, maar wel eventuele nieuwe doorgeefacties. Op deze manier kan bekeken worden hoeveel lampen er stuk zijn waarvan er geen berichten meer in het netwerk zitten. Deze berichten zijn uitgestorven en de defecte lampen zullen dus niet gemaakt worden.

Verificatie/validatie

Om te testen of het programma werkt zoals het bedacht is en om te bekijken of op deze manier inderdaad de methodes te simuleren zijn, is allereerst het netwerk dat wordt opgebouwd bekeken. Ook is bekeken of het aantal stappen dat een knoop van het basisstation verwijderd klopt. Door middel van het stap voor stap volgen van het programma is het proces gevalideerd en geverifiëerd. Er is onder andere bekeken of berichten niet zomaar onnodig verwijderd worden en goed doorgegeven worden, of dubbele berichten goed worden opgemerkt, of het basisstation de informatie goed en op het goede moment verwerkt en of berichten zich zo verspreiden zoals de methode dit voorschreef. Ook is het kapot gaan van lampen gevolgd. Er kan geconcludeerd worden dat het programma werkt zoals het moet werken.

Hoofdstuk 6

Experimenten

Om te bekijken hoe goed de verschillende methodes het doen zijn verschillende experimenten uitgevoerd. Voor deze experimenten worden steeds andere methodes of parameters gebruikt. De uitkomsten van de experimenten zijn te vinden in hoofdstuk 7 – Resultaten.

Allereerst worden er experimenten met verschillende methodes gedaan om berichten door te geven. Dit om de methodes later met elkaar te kunnen vergelijken. Verder worden de volgende parameters per experiment aangepast:

- De grootte van het rooster waar de lampen zich in bevinden. Er is gewerkt met twee groottes: in totaal 400 mogelijke lampen, of in totaal 1600 mogelijke lampen.
- De vorm van het basisrooster. Er worden drie verschillende vormen aangenomen: vierkant (20x20 en 40x40), rechthoekig (40x10 en 80x20) en een smalle rechthoek met een korte kant van 2 (dus 200x2 en 800x2). De smalle rechthoek kan vergeleken worden met een lange weg waaraan bijvoorbeeld lantaarnpalen staan. Ook in dit soort netwerken kunnen er meldingen worden doorgegeven van kapotte lampen met behulp van de methodes. Vandaar dat ook deze laatste variant een interessante kijk op de werking van de methodes op kan leveren.
- De volgende parameter die kan worden aangepast is λ. Met behulp van λ en de exponentiële verdeling (parameter $1/\lambda$) word de tussentijd tussen twee kapotte lampen bepaald. Hoe groter λ is, hoe meer lampen er kapot gaan per tijdseenheid. Door λ te variëren, kan getest worden hoe de verschillende methodes presteren op momenten dat er weinig kapot is en er vaak geheugenplaatsen vrij zijn en op momenten dat er veel kapot gaat en er vaak een tekort zal zijn aan geheugenplaatsen. Er is gewerkt met een λ van 0.3 en een λ van 3.
- De volgende parameter die veranderd wordt per experiment is p. Met behulp van p wordt voor de 2e methode, waarin met een bepaalde kans berichten naar de beste buur worden doorgegeven, de kans bepaald dat berichten naar deze beste buur gaan. Door p te variëren kan bekeken worden in welke gevallen deze methode goed werkt. De gebruikte waarden voor p zijn 0.2, 0.4, 0.6, 0.8 en 1. Methode 2 met een p van 1 is niet per definitie gelijk aan methode 3. Er kunnen nog steeds berichten worden gewist en er kunnen dubbele berichten ontstaan.

- Als laatste worden bij methode 3 simulaties gedraaid waarbij iedere kapotte lamp 1 bericht uitzend en simulaties waarbij er alleen bij de eerste stap een bericht wordt gestuurd naar alle buren, voor zover hier geheugenplaatsen voor zijn. Hierbij worden er dus 1 tot 4 berichten het netwerk in gestuurd. Dit aantal berichten wordt niet verder vergroot. Het meerdere berichten in het netwerk sturen zou kunnen werken in drukke netwerken waarbij de kans aanwezig is dat er geen vrije geheugenplekken zijn voor een lamp die stuk gaat en er hiervoor een bericht wordt verwijderd. In het geval dat er meerdere berichten zijn van een kapotte lamp is de boodschap hierbij niet direct uitgestorven. Ook zou er op deze manier een snellere route gevonden kunnen worden die een knelpunt in de kortste route vermijdt. De winst in wachttijd zou op kunnen wegen tegen het verlies door het grotere aantal berichten in het netwerk.

Omdat er vele combinaties ontstaan bij het kiezen van verschillende parameters, kon niet iedere parameter afzonderlijk getest worden. Om deze reden is bijvoorbeeld niet bekeken bij welke kans van doorgeven aan een buur methode 1 (het gereduceerd roddelen) goed werkt, aangezien uit literatuur bleek dat dit tussen de 0.6 en 0.8 moest zitten voor een optimale werking van de methode [3]. In de simulaties is met 0.7 gewerkt.

Verder zijn de netwerken in de roosters opgebouwd door met een bepaalde kans een bepaalde kant op te kunnen. Voor een vierkant is voor alle richtingen de kans 0.8 genomen, voor een rechthoek de kans 0.8 in de lengte en 0.6 in de breedte, en voor de zeer langwerpige rechthoek met dikte 2 de kans 1 in de lengte, en 0.4 in de breedte. Enkele andere vaste parameters zijn de doorgeeftijd, die is ingesteld op 5 tijdseenheden, het aantal geheugenplaatsen dat is ingesteld op 30 en de doorgeefkansen naar de buren bij methode 1: 1 bij minder dan 2 buren en 0,7 bij minimaal 2 buren.

Met alle combinaties van bovenstaande instellingen zijn simulaties gedraaid. Voor al deze simulaties heb ik een aantal zaken bijgehouden:

- Het aantal keren dat een bericht op het moment dat een lamp stuk gaat nergens kan worden opgeslagen en hiervoor een bericht verwijderd moet worden;
- De verwachte wachttijd voor een kapotte lamp op het moment dat deze stuk gaat tot het moment dat deze gemaakt wordt;
- Een 95% betrouwbaarheidsinterval voor de verwachte wachttijd;
- De langste wachttijd die voor is gekomen;
- Het aantal lampen dat in totaal in de simulatie stuk ging;
- Het aantal lampen dat in totaal in de simulatie gemaakt werd;
- De ratio van het totaal aantal lampen dat gemaakt is in de simulatie / totaal aantal keren dat een lamp stuk ging in de simulatie.

Na een aantal experimenten te hebben uitgevoerd, blijkt er iets vreemds te gebeuren. Als illustratie staan de resultaten van een rooster van 20x20 in tabel 6.1.

Rooster	Meth.	λ	Variatie	Verw wachttijd	BTI	Max wachttijd	# gemaakt	# kapot	Ratio gemaakt/kapot
20x20	1	3	-	5,8	[5,3 ; 6,2]	84	127691	127530	0.9987
20x20	2	3	0.2	102,0	[95,0 ; 108,9]	131531	117094	116988	0,9991
20x20	2	3	0.4	88,1	[75,9 ; 100,3]	164904	154523	154443	0,9995
20x20	2	3	0.6	83,4	[76,6 ; 90,1]	156070	152493	152404	0,9994
20x20	2	3	0.8	84,3	[75,8 ; 92,9]	162572	173453	173377	0,9996
20x20	2	3	1	77,1	[69,2 ; 85,0]	152346	191128	191050	0,9996
20x20	3	3	noGossip	15.4	[15.3 ; 15.4]	52	376971	376971	1
20x20	3	3	Gossip	13.5	[13.5 ; 13.6]	52	385333	385333	1

Tabel 6.1: De resultaten van een rooster van 20x20 voor alle methodes

Bij methode 1 wordt niet alles gemaakt, maar zijn er wel hele korte wachttijden. Deze wachttijden zijn gemiddeld gezien erg vreemd, aangezien berichten van een kapotte lamp van ver bij het basisstation vandaan er toch een minimale tijd over zouden moeten doen om het basisstation te kunnen bereiken. Ook gaan er weinig lampen stuk, terwijl er met dezelfde λ is gewerkt. Dit alles zou te verklaren kunnen zijn doordat berichten van kapotte lampen die ver van het basisstation zitten, het basisstation niet gemakkelijk kunnen bereiken. Door de roddelmethode loopt het hele netwerk vol met berichten. Doordat het zo vol is, is de kans groot dat berichten uitsterven. In dat geval blijft de lamp altijd stuk en kan deze ook niet opnieuw stuk gaan. Hiermee is dus ook te verklaren waarom er veel minder kapotte lampen zijn geweest in het experiment. De korte wachttijd is te verklaren doordat de wachttijd steeds wordt berekend zodra er een bericht bij het basisstation binnenkomt. Berichten die hier nooit aankomen, worden dus ook niet meegenomen in de berekening. Alleen berichten van kapotte lampen in de buurt van het basisstation komen aan, hetgeen een korte wachttijd veroorzaakt. De methode lijkt op het eerste gezicht goed te werken, maar doet het ondertussen dus niet erg best. Ook aan de ratio aantal gemaakt/aantal kapot is niet veel te zien. Juist omdat sommige lampen maar 1 keer stuk gaan en nooit meer gemaakt worden, en de simulatietijd erg lang is, weegt dit niet al te zwaar mee in het geheel.

Om dit effect mee te nemen in te resultaten wordt in de volgende serie experimenten steeds aan het eind van iedere run bekeken welke lampen stuk zijn, en hoe lang deze dit al zijn. Deze tijd wordt opgeteld bij de totale wachttijd. Om de totale berekening niet te verstoren, wordt de tijd dat de lamp kapot ging verzet naar de huidige simulatietijd. Op deze manier worden er geen wachttijden dubbel geteld. De lamp blijft wel kapot en aan de berichten die wellicht nog onderweg zijn wordt ook niets veranderd. Indien een bepaald bericht is uitgestorven en de lamp nooit meer gemaakt zal worden, is dit nu ook terug te zien als een soort boetekosten in de wachttijd, die iedere run weer hoog is. Dit is niet onterecht; er zijn lampen die al erg lang stuk zijn en al erg lang wachten op een reparateur. De experimenten die gedaan zijn met deze verandering geven een betere indicatie van de kwaliteit van de methoden. Hierdoor zijn de methoden beter met elkaar te vergelijken.

Verder blijkt dat bij methode 3, waarbij de kortste weg voor ieder bericht wordt gezocht, dat de optie om berichten naar achter te sturen (dus naar een knoop die verder bij het basisstation vandaan ligt dan de knoop waar het bericht vandaan zou komen) erg slecht werkt. Dit was te verwachten, aangezien berichten uiteindelijk toch weer proberen via dezelfde route naar het basisstation te komen. Deze optie wordt dan ook niet meegenomen in de experimenten die verder besproken zullen worden.

Hoofdstuk 7

Resultaten

De drie methodes zijn getest in roosters van verschillende *vorm* en *grootte*. Verder zijn de parameters λ en p gevarieerd per simulatie. De uitkomsten van deze simulaties zijn te vinden in Bijlage A. De eerste tabel is ook te zien in tabel 7.1. De kolom Variatie geeft bij methode 2 de kans dat het bericht naar de beste buur gaat (p) aan en bij methode 3 of er de eerste stap wordt doorgegeven aan alle buren.

Rooster	Meth.	λ	Variatie	Verw wachttijd	BTI			Max wachttijd	# gemaakt	# kapot	Ratio gemaakt/kapot
20x20	1	3	-	203.7	[183.1	;	224.3]	84	127691	127530	0.9987
20x20	2	3	0.2	204.6	[197.9	;	211.3]	3993	117094	116988	0.9991
20x20	2	3	0.4	137.3	[132.1	;	142.6]	3974	154523	154443	0.9995
20x20	2	3	0.6	143.7	[140.2	;	147.2]	3998	152493	152404	0.9994
20x20	2	3	0.8	113.6	[108.7	;	118.6]	3994	173453	173377	0.9996
20x20	2	3	1	96.9	[92.8	;	101.1]	3986	191128	191050	0.9996
20x20	3	3	noGossip	15.4	[15.3	;	15.4]	52	376971	376971	1
20x20	3	3	Gossip	13.5	[13.5	;	13.6]	52	385333	385333	1

Tabel 7.1: De resultaten van een rooster van 20x20 voor alle methodes

Er valt op in de tabellen dat methode 3 (de kortste weg) een erg korte wachttijd heeft in vergelijking met de andere methodes. Zowel bij de variant waarbij bij de eerste stap de berichten als gossip worden verspreid als bij de variant dat er van iedere kapotte lamp maar 1 bericht in het netwerk is, worden de laagste gemiddelde wachttijden behaald, waarbij de methode het snelste werkt als er de eerste stap op de gossip manier wordt doorgegeven. Methode 2 wordt beter naarmate de kans op doorgeven aan de beste buur groter wordt. Methode 1 werkt in kleinere netwerken (max. 400 lampen) ook nog wel redelijk, maar bij grotere netwerken (max. 1600 lampen) doet deze methode het altijd het slechtst. Opvallend is dat in het figuur dat 2 lampen breed is, de zogenaamde ladder, alle gemiddelde wachttijden onder de 300 liggen. Voor dit figuur zijn alle methoden goed te gebruiken, al is methode 3 wel de meest effectieve.

Het aantal kapotte lampen

Te zien is ook dat bij methode 1 de minste lampen stuk gaan en gemaakt worden. Dit komt doordat de wachttijd bij deze methode langer is dan bij de andere methodes en lampen daardoor minder vaak stuk kunnen gaan. Het gebeurt dus vaak dat op het moment dat de lamp wordt ingepland om stuk te gaan, nog stuk is. Indien de boodschap van een kapotte lamp uitsterft, zal de lamp altijd stuk blijven en kan deze nooit meer opnieuw stuk gaan. Ook hierdoor blijft het aantal kapotte lampen laag. Bij methode 3 is het effect groot: voor sommige roosters geldt dat er bijna 4x zoveel lampen stuk gaan als bij methode 1.

Wachttijden nader bekeken

Als naar de maximale wachttijd gekeken wordt, valt op dat deze voor methode 2 altijd hoger is dan voor de andere methodes. In sommige gevallen zelfs er veel hoger. Er worden wachttijden bereikt van bijna 4000, de maximale simulatietijd per run. Dit is te verklaren doordat na iedere run de opgebouwde wachttijd van kapotte lampen wordt verwerkt en de wachttijd op 0 wordt gezet. Het kan zijn dat deze berichten al meerdere runs in het netwerk zaten. Wel komen deze berichten uiteindelijk aan. De maximale wachttijd wordt bepaald onder berichten die aangekomen zijn. De maximale wachttijd van methode 1 is nooit erg hoog. Maar niet ieder bericht komt aan. Dit betekent dat als berichten er eenmaal niet doorheen komen, waarschijnlijk nooit meer aankomen. Dat de gemiddelde wachttijd veel hoger is komt dan ook doordat hierin ook berichten die nog niet zijn aangekomen worden meegenomen. Bij methode 3 is dit anders. De maximale wachttijd is niet erg hoog, maar is wel altijd hoger dan de gemiddelde wachttijd. Dat was te verwachten, want bij deze methode komen alle berichten aan.

Verder valt bij de wachttijden op, dat de wachttijden bij een hoge λ, en dus veel lampen die stuk gaan, er een lagere gemiddelde wachttijd is voor dezelfde simulaties met een lage λ. Dit lijkt vreemd, maar is te verklaren doordat lampen rond het basisstation eerder worden gemaakt dan andere lampen. Deze kunnen dus ook weer eerder stuk gaan. Op deze manier gaan er veel vaker lampen met een korte reparatietijd stuk dan lampen met een lange reparatietijd en tellen de korte reparatietijden dus zwaarder. Dit haalt de gemiddelde wachttijd naar beneden.

Dit effect is ook te zien doordat de gemiddelde wachttijd voor bijvoorbeeld reparaties bij methode 3 lager is dan zou worden verwacht als naar de grootte van het rooster gekeken wordt en berekend wordt hoe lang het minimaal zou moeten duren voordat een bericht van een lamp van ver bij het basisstation vandaan aan zou komen. Het is te verklaren door het frequente kapot gaan van de lampen rond het basisstation.

De maximale wachttijd bij methode 1, het roddelen, verschilt amper als er gesimuleerd wordt met verschillende λ. Het had gekund dat er meer berichten van ver aan zouden komen als het aantal lampen dat kapot gaat per tijdseenheid naar beneden zou worden geschroefd. Dit is echter niet het geval. Doordat berichten zich zo snel verspreiden, lopen alsnog de geheugenplekken rond het basisstation snel vol. Er zijn immers in het geval van 30 geheugenplaatsen maar 30 verschillende berichten nodig om het geheugen vol te krijgen, dus er hoeven in principe maar 30 lampen stuk te gaan. Zodra de geheugens vol zitten doet het probleem zich voor dat berichten van ver er niet meer doorheen komen, hetgeen dus al snel het geval zal zijn.

Stabiliteit

Het valt op dat het percentage lampen dat gemaakt wordt erg constant is, voor alle drie de methodes. Voor methode 3 is dit percentage altijd 1, hetgeen betekent dat altijd alle lampen worden gemaakt. Ook valt bij alle methodes op dat de wachttijden niet extreem toenemen zodra de grootte van het netwerk toeneemt. Dit geeft aan dat alle methodes op een stabiele manier werken. Alleen in een vierkant neemt bij methode 1 de wachttijd behoorlijk toe. Dit kan komen doordat als er een lamp bij komt, het totaal aantal lampen aan berichten bij komt. Dit in tegenstelling van methode 3, waarbij er hoogstens 4 berichten bij komen indien er de eerste stap met gossip wordt gewerkt.

Hoofdstuk 8

Conclusies

Na de simulatieresultaten te bekijken blijkt methode 3, waarin berichten via de kortste weg naar het basisstation gaan, het best te werken. Deze methode had niet alleen de kortste verwachte wachttijd, maar ook de beste betrouwbaarheid: alle kapotte lampen werden gemaakt. Vooral wanneer bij deze methode de eerste stap bij een kapotte lamp naar alle buren indien mogelijk een bericht gaat, werkt de methode snel. Blijkbaar loont het om meerdere berichten te versturen en daarmee de kans op het vinden van een snelle route te vergroten. De snelste route hoeft hierbij niet de kortste te zijn. Omdat berichten niet verwijderd worden kan er bij knelpunten lang worden gewacht. Wanneer berichten op dezelfde knoop uitkomen worden niet beide berichten opgeslagen. Hierdoor kan het aantal berichten worden gereduceerd en is het nadeel van meerdere berichten in het netwerk compenseerbaar.

De andere methodes werken redelijk. Een groot gedeelte van de berichten komt aan, maar dit is ook te wijten aan het feit dat een lamp die eenmaal kapot is niet nogmaals kapot kan gaan. Het aantal lampen dat nooit gemaakt zal worden weegt niet op tegen het grote aantal keren dat lampen in totaal stuk gaan. Om de wachttijden van lampen die nooit gemaakt worden wel mee te kunnen nemen was het noodzakelijk om per simulatierun ook van de nog niet gemaakte lampen de wachttijd te bekijken. Indien dit niet gedaan wordt, worden de resultaten verstoord doordat de enige lampen die gemaakt worden lampen dicht bij het basisstation blijken te zijn.

Over het algemeen zijn de wachttijden in een vierkant het langst, gevolgd door de wachttijden in een rechthoek, gevolgd door de wachttijden in een ladder. Dit geldt voor zowel grote als kleine netwerken en voor zowel een grote als een kleine λ. Bij een kleinere λ (en daarmee minder vaak kapotte lampen) was de gemiddelde wachttijd niet langer. Dit is te verklaren doordat lampen die snel gemaakt worden ook weer snel stuk kunnen gaan en op deze manier veel vaker mee kunnen worden genomen in het gemiddelde. Juist met een hoge λ is dit effect sterk aanwezig, waardoor de gemiddelde wachttijden naar beneden worden getrokken.

Hoofdstuk 9

Discussie

Uit de resultaten bleek dat het doorgeven van berichten via de kortste route naar het basisstation een betrouwbare en snelle methode is. Deze methode werkt het best als de beschikbare informatie van lampen juist en volledig is. Het kan voorkomen dat er lampen uit het netwerk worden gehaald of dat het netwerk wordt uitgebreid met nieuwe lampen. In deze gevallen moet het systeem goed aan te passen zijn.

Een extra lamp kan bij het bekend maken van de buurknopen direct berekenen hoe lang de route vanaf de knoop die bij de lamp hoort minimaal is. Maar er is voor andere knopen wellicht een kortere route ontstaan met de komst van deze nieuwe lamp. Een soortgelijk probleem ontstaat als er een lamp wegvalt uit het netwerk.

Verder kan onderzocht worden hoe drukke knelpunten in het netwerk op een effectieve manier ontlast kunnen worden. Dit zou de wachttijd kunnen verbeteren en zou er toe kunnen leiden dat er een minder groot aantal geheugenplaatsen nodig is.

Voor een andere kijk op de wachttijd kan de wachttijd worden gedeeld door de afstand die de boodschap minimaal af heeft moeten leggen. Deze kortste afstand is bekend en gemakkelijk na te zoeken. Het effect van het vaker stukgaan en gemaakt worden van lampen die dicht bij het basisstation liggen, en de gevolgen hiervan op de wachttijd, kunnen hiermee nader bekeken worden.

Als laatste kan onderzocht worden of het vaker uitzenden van het bericht dat er een lamp stuk is verbeteringen biedt. In de simulaties die voor dit werkstuk gedraaid werden, was er altijd maar één moment waarop een kapotte lamp berichten in het netwerk stuurde. Dit zou nog een aantal keer, met bepaalde tussentijden, herhaald kunnen worden, of zelfs door kunnen gaat tot het moment dat de lamp gemaakt is, hetgeen simpel kan worden nagegaan met behulp van de detector bij iedere lamp. Het nadeel van het vaker uitzenden van berichten zou kunnen zijn dat het netwerk nog voller raakt met berichten en hierdoor de situatie verslechterd. Het voordeel zou echter kunnen zijn dat op deze manier de kans op uitsterven van een bericht afneemt en daarmee de betrouwbaarheid van de methode toeneemt.

Appendices

Bijlage A

Resultaten

Rooster	Methode	λ	Variatie	Verw wachttijd	BTI			Max wachttijd	# gemaakt	# kapot	Ratio gemaakt/kapot
20x20	1	3	-	203.7	[183.1	;	224.3]	84	127691	127530	0.9987
20x20	2	3	0.2	204.6	[197.9	;	211.3]	3993	117094	116988	0.9991
20x20	2	3	0.4	137.3	[132.1	;	142.6]	3974	154523	154443	0.9995
20x20	2	3	0.6	143.7	[140.2	;	147.2]	3998	152493	152404	0.9994
20x20	2	3	0.8	113.6	[108.7	;	118.6]	3994	173453	173377	0.9996
20x20	2	3	1	96.9	[92.8	;	101.1]	3986	191128	191050	0.9996
20x20	3	3	noGossip	15.4	[15.3	;	15.4]	52	376971	376971	1
20x20	3	3	Gossip	13.5	[13.5	;	13.6]	52	385333	385333	1
20x20	1	0.3	-	548.9	[492.7	;	605.1]	81	26429	26312	0.9956
20x20	2	0.3	0.2	1410.6	[1390.8	;	1430.4]	3946	11963	11812	0.9874
20x20	2	0.3	0.4	1410.9	[1387.0	;	1434.8]	3979	11907	11758	0.9875
20x20	2	0.3	0.6	1220.3	[1202.4	;	1238.2]	3989	13947	13810	0.9902
20x20	2	0.3	0.8	1068.1	[1049.6	;	1086.7]	3991	15917	15782	0.9915
20x20	2	0.3	1	412.4	[405.3	;	419.5]	3979	29452	29375	0.9974
20x20	3	0.3	noGossip	16.8	[16.7	;	16.9]	52	47108	47108	1
20x20	3	0.3	Gossip	15.6	[15.5	;	15.7]	52	47186	47186	1
40x10	1	3	-	202.4	[187.7	;	217.2]	56	81434	81336	0.9988
40x10	2	3	0.2	72.8	[71.3	;	74.3]	3998	158743	158714	0.9998
40x10	2	3	0.4	63.9	[61.2	;	66.6]	3999	170488	170456	0.9998
40x10	2	3	0.6	58.1	[56.2	;	60.0]	3909	182934	182905	0.9998
40x10	2	3	0.8	55.0	[52.5	;	57.5]	3933	186837	186814	0.9999
40x10	2	3	1	52.8	[50.4	;	55.1]	3998	194016	193983	0.9998
40x10	3	3	noGossip	11.3	[11.3	;	11.3]	45	345223	345223	1
40x10	3	3	Gossip	9.4	[9.4	;	9.5]	45	358621	358621	1
40x10	1	0.3	-	795.6	[721.1	;	870.0]	53	15119	15028	0.9940
40x10	2	0.3	0.2	541.6	[529.5	;	553.8]	3974	18763	18699	0.9966
40x10	2	0.3	0.4	438.9	[432.0	;	445.8]	3988	21514	21456	0.9973
40x10	2	0.3	0.6	365.9	[359.4	;	372.4]	3673	23862	23812	0.9979
40x10	2	0.3	0.8	335.5	[327.6	;	343.5]	3913	25098	25052	0.9982
40x10	2	0.3	1	171.7	[168.9	;	174.5]	3978	33250	33227	0.9993
40x10	3	0.3	noGossip	13.4	[13.3	;	13.5]	45	47297	47297	1
40x10	3	0.3	Gossip	12.3	[12.2	;	12.3]	45	47433	47433	1
200x2	1	3	-	11.8	[11.1	;	12.6]	19	154906	154896	0.9999
200x2	2	3	0.2	5.7	[5.7	;	5.7]	271	224266	224266	1
200x2	2	3	0.4	5.4	[5.3	;	5.4]	263	226640	226640	1
200x2	2	3	0.6	4.8	[4.8	;	4.9]	150	228957	228957	1
200x2	2	3	0.8	4.8	[4.8	;	4.8]	148	254283	254283	1
200x2	2	3	1	4.5	[4.5	;	4.5]	204	256085	256085	1
200x2	3	3	noGossip	3.4	[3.4	;	3.4]	24	280222	280222	1
200x2	3	3	Gossip	2.7	[2.7	;	2.7]	24	301781	301781	1
200x2	1	0.3	-	83.9	[81.8	;	86.1]	23	19685	19675	0.9995
200x2	2	0.3	0.2	9.2	[9.1	;	9.3]	188	41454	41454	1
200x2	2	0.3	0.4	8.2	[8.1	;	8.4]	140	42494	42494	1
200x2	2	0.3	0.6	7.5	[7.4	;	7.7]	158	43519	43519	1
200x2	2	0.3	0.8	7.1	[7.0	;	7.2]	123	43717	43717	1
200x2	2	0.3	1	7.5	[7.4	;	7.6]	252	42323	42323	1
200x2	3	0.3	noGossip	6.0	[6.0	;	6.1]	24	43902	43902	1
200x2	3	0.3	Gossip	5.5	[5.5	;	5.6]	24	44268	44268	1

Rooster	Methode	λ	Variatie	Verw wachttijd	BTI			Max wachttijd	# gemaakt	# kapot	Ratio gemaakt/kapot
40x40	1	3	-	644.0	[584.5	;	703.4]	65	52027	51822	0.9961
40x40	2	3	0.2	199.5	[192.1	;	206.8]	3993	127609	127493	0.9991
40x40	2	3	0.4	194.7	[188.2	;	201.3]	3999	127255	127147	0.9992
40x40	2	3	0.6	161.1	[154.5	;	167.7]	4230	147935	147832	0.9993
40x40	2	3	0.8	149.2	[144.8	;	153.6]	4075	155370	155268	0.9993
40x40	2	3	1	150.6	[145.2	;	156.0]	3988	154104	153992	0.9993
40x40	3	3	noGossip	21.3	[21.2	;	21.3]	59	347358	347358	1
40x40	3	3	Gossip	19.0	[19.0	;	19.1]	59	355876	355876	1
40x40	1	0.3	-	1536.0	[1391.8	;	1680.2]	62	12946	12757	0.9854
40x40	2	0.3	0.2	1406.2	[1378.8	;	1433.7]	3994	12768	12610	0.9876
40x40	2	0.3	0.4	1412.4	[1395.6	;	1429.2]	3998	12719	12560	0.9875
40x40	2	0.3	0.6	1272.4	[1257.7	;	1287.2]	3909	14222	14072	0.9895
40x40	2	0.3	0.8	1015.7	[1007.3	;	1024.1]	4248	17473	17339	0.9923
40x40	2	0.3	1	642.7	[634.9	;	650.5]	3970	24127	24021	0.9956
40x40	3	0.3	noGossip	23.3	[23.2	;	23.5]	59	46885	46885	1
40x40	3	0.3	Gossip	22.0	[21.9	;	22.2]	59	46963	46963	1
80x20	1	3	-	250.6	[225.7	;	275.4]	84	127137	126936	0.9984
80x20	2	3	0.2	271.6	[259.4	;	283.7]	3990	110575	110424	0.9986
80x20	2	3	0.4	254.8	[242.4	;	267.1]	4000	116076	115933	0.9988
80x20	2	3	0.6	216.2	[209.7	;	222.7]	3985	130928	130801	0.9990
80x20	2	3	0.8	184.8	[177.5	;	192.0]	3996	146852	146736	0.9992
80x20	2	3	1	146.1	[141.2	;	151.0]	3996	173859	173738	0.9993
80x20	3	3	noGossip	18.3	[18.3	;	18.3]	49	380894	380894	1
80x20	3	3	Gossip	16.4	[16.4	;	16.4]	49	388096	388096	1
80x20	1	0.3	-	781.1	[703.2	;	858.9]	85	24194	24034	0.9934
80x20	2	0.3	0.2	1895.5	[1861.0	;	1930.0]	3964	9728	9525	0.9791
80x20	2	0.3	0.4	1849.4	[1823.0	;	1875.8]	3971	10061	9861	0.9801
80x20	2	0.3	0.6	1719.7	[1697.0	;	1742.4]	3996	11102	10907	0.9824
80x20	2	0.3	0.8	1608.7	[1576.4	;	1641.0]	3993	12075	11888	0.9845
80x20	2	0.3	1	769.4	[754.5	;	784.3]	3885	23220	23093	0.9945
80x20	3	0.3	noGossip	19.9	[19.8	;	20.0]	49	48391	48391	1
80x20	3	0.3	Gossip	18.6	[18.5	;	18.7]	49	48453	48453	1
800x2	1	3	-	12.1	[11.4	;	12.7]	18	152396	152386	0.9999
800x2	2	3	0.2	5.6	[5.6	;	5.6]	227	224416	224416	1
800x2	2	3	0.4	5.2	[5.1	;	5.2]	161	241461	241461	1
800x2	2	3	0.6	4.8	[4.8	;	4.9]	132	239503	239503	1
800x2	2	3	0.8	4.9	[4.8	;	4.9]	135	247120	247120	1
800x2	2	3	1	4.7	[4.7	;	4.7]	229	241047	241047	1
800x2	3	3	noGossip	3.4	[3.4	;	3.4]	24	278751	278751	1
800x2	3	3	Gossip	2.7	[2.7	;	2.7]	24	301232	301232	1
800x2	1	0.3	-	81.4	[80.3	;	82.6]	28	20174	20164	0.9995
800x2	2	0.3	0.2	9.3	[9.2	;	9.5]	175	41868	41868	1
800x2	2	0.3	0.4	8.1	[8.0	;	8.2]	155	42221	42221	1
800x2	2	0.3	0.6	7.5	[7.4	;	7.6]	158	43220	43220	1
800x2	2	0.3	0.8	6.9	[6.9	;	7.0]	120	43543	43543	1
800x2	2	0.3	1	7.5	[7.4	;	7.6]	256	43492	43492	1
800x2	3	0.3	noGossip	6.0	[6.0	;	6.1]	24	43665	43665	1
800x2	3	0.3	Gossip	5.6	[5.5	;	5.6]	24	44026	44026	1

Bibliografie

[1] P. Korteweg, M. Nuyens, R. Bisseling, T. Coenen, H. van den Esker, B. Frenk R. de Haan, B. Heydenreich, R. van der Hofstad, J. in t Panhuis, L. Spanjers, and M. van Wieren. Math saves the forest. *Mathematics in Industry*, 2006. [op p. 5]

[2] D. Gavidia, S. Voulgaris, and M. Steen. A gossip-based distributed news service for wireless mesh networks. *WONS (INRIA - CCSd - CNRS)*, 3:59–67, 2006. [op p. 5, 12]

[3] Z.J. Haas, J.Y. Halpern, and L. Li. Gossop-based ad hoc routing. *IEEE INFOCOM*, 14(3), 2006. [op p. 10, 12, 20]

[4] C.L. Barret, S.J. Eidenbenz, L.Kroc, M. Matathe, and J.P. Smith. Parametric probabilistic sensor network routing. *WSNA*, 2003. [op p. 12]

[5] E.W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959. [op p. 13]