Practicum 2 Synchronisatie bij vuurvliegjes

Peter van Dijk & Elizabeth Schermerhorn

29 juni 2014

Inhoudsopgave

1 Inleiding

In de natuur zijn er verschillende vormen van synchronisatie. Aangezien synchronisatie een belangrijk punt is in de werking tussen verschillende nodes in een netwerk is het van belang om een goed algoritme hiervoor te hebben. Een bekende uit de natuur zijn vuurvliegjes die met elkaar synchroniseren wat zich uit in samen knipperen. Dit principe dat vuurvliegjes synchroon kunnen knipperen wordt geïmplementeerd in een multihop netwerk. Door een algoritme te ontwikkelen moet het mogelijk zijn om nodes tegelijk te laten knipperen net zoals een groep vuurvliegjes.

2 Probleemstelling

De probleemstelling welke hier beantwoord dient te zijn aan het einde van dit experiment is: *"het synchroniseren van nodes in een multihop netwerk met als doel ze tegelijk te laten knipperen."*. Er zijn eisen waaraan de synchronisatie moet voldoen:

- Wanneer twee gesynchroniseerde netwerken worden samengevoegd dienen ze te synchroniseren.
- Wanneer een node uitvalt dient de synchronisatie door te gaan.
- Wanneer nodes toegevoegd worden aan het netwerk gaan deze met dezelfde frequentie knipperen als de andere nodes in het netwerk.
- Wanneer nodes uit sync raken moeten ze opnieuw synchroniseren.

De hypothese is dat dit gaat lukken. Er is veel gerelateerd werk waarin wordt beschreven hoe multi hop netwerken synchroniseren. NOG IETS TOEVOEGEN????!!!!!!!!!!

3 Gerelateerd werk

Er zijn verschillende onderzoeken gedaan naar het onderwerp synchronisatie in een multihop netwerk. Een paar papers welke dit onderwerp aansnijden zijn:

- Clock synchronization for wireless sensor networks: a survey
- Wireless sensor network survey
- Academic Press Library in Signal Processing, Chapter 2 Synchronization

De bovenstaande drie papers gaan over synchronisatie in draadloze netwerken. De eerste paper - Clock synchronization for wireless sensor networks: a survey - evalueert bestaand kloksynchronisatie algoritmen gebasseerd op factoren zoals precisie, complexiteit, nauwkeurigheid en kosten. De tweede paper, Wireless

sensor network, geeft een overzicht van bestaande draadloze netwerk implementaties op verschillende niveaus en als laatste geeft het boek Academic Press Library in Signal Processing geeft een kort overzicht van netwerk tijd synchronisatie.

In Sectie 3.3 van de paper "Clock synchronization for wireless sensor networks: a survey"wordt gesproken over de verschillende mogelijkheden bij de implementatie en hoe deze in hun werk gaan. De volgende keuzes moeten gemaakt worden wanneer er een draadloos sensor netwerk ontwikkeld wordt.

- Master-slave protocol versus peer-to-peer synchronisatie
- Interne synchronisatie versus externe synchronisatie
- Single-hop netwerk versus multihop netwerk
- probabilistische versus deterministische synchronisatie

In de bovengenoemde paper staan nog meer keuzemogelijkheden, deze zijn niet van belang voor dit experiment en zijn voor het gemak weggelaten. Hier volgt een korte toelichting op de verschillende onderdelen.

Onderdeel	**Uitleg**
Master-slave protocol	Er is een master node in het netwerk waar alle andere nodes naar luisteren, deze bepaald te synchronisatie
peer-to-peer protocol	Elke node in het netwerk heeft contact met elke andere node. Dit elimineert het risico dat wanneer een master node wegvalt het netwerk niet meer synchroniseert.
Interne synchronisatie	De reële tijd is niet beschikbaar. Het doel is het verschil in tijd tussen de lokale klok en het uitlezen van de sensor zo klein mogelijk te maken.
externe synchronisatie	Hier is de tijd wel beschikbaar zoals UTC. Er is een atomische klok die de ware tijd heeft.
single hop netwerk	Elke node in het netwerk heeft contact met elke andere node in het netwerk
multi hop netwerk	Niet elke node in het netwerk heeft contact met een andere node in het netwerk.
probabilistische synchronisatie	Geeft een probabilistische garantie van de maximum offset van de klok.
deterministische synchronisatie	Deze algoritme garanderen een bepaalde maximum offset van de klok.

Dit zijn allemaal keuzes die gemaakt moeten worden bij het implementeren van een draadloos sensor netwerk. Afhankelijk van de eisen die aan het netwerk gesteld worden moet er voor bepaalde onderdelen gekozen worden. Een paar voorbeelden hiervan zijn: Er is geen server beschikbaar, het moet zo min mogelijk energie verbruiken zodat de nodes lang meegaan of de nodes staan ver uit elkaar. Dit zijn slechts een aantal voorbeelden.

In de tweede genoemde paper, Wireless sensor network survey, worden in sectie 6.2 meerdere synchronisatie algoritmen uitgelegd. De volgende algoritmen worden beschreven:

- Onzekerheids gedreven benadering
- Lucarelli's algoritme
- Vuurvlieg algoritme

- Tijd synchronisatie protocol voor sensor netwerken
- klok-opname gezamenlijke netwerk synchronisatie
- Tijd synchronisatie
- globale synchronisatie synchronisatie

Het geïmplementeerde algoritme is gebaseerd op het vuurvlieg algoritme. Hoewel het geïmplementeerde algoritme erop gebaseerd is, is het op sommige punten aangepast. Het beschreven algoritme in de paper werkt als volgt: Elke node in het netwerk werkt als een oscillator met een vastgestelde tijd T. Elke node heeft een interne tijd t die opgehoogd wordt totdat deze gelijk is aan T. Op tijd T zal de node een signaal sturen en zijn interne tijd t weer op 0 zetten. Buren van deze node die dit signaal ontvangen zullen de tijd tussen t en T verkleinen. Dit wordt bepaald door middel van een functie en een kleine constante. Na verloop van tijd zullen de nodes gelijk hun signaal versturen. in figuur XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX is dit algoritme schematisch weergegeven. De bedoeling is dat alle nodes tegelijk een signaal gaan versturen na verloop van tijd. PLAATJUUUUUUHHHHHH firefly.png Het grote verschil tussen bovenstaande implementatie en de experimentele implementatie van deze paper is dat de t niet naar 0 wordt gezet maar het verschil wordt bijgehouden hoelang het duurt voor er weer een bericht wordt ontvangen. Afhankelijk hiervan wordt er zolang geslapen waardoor uiteindelijk de nodes synchroniseren. PLAATJEUUUUUUUUHHHH onzeimplementatie.png

4 Ontwerp synchronisatiealgoritme

Het synchronisatie algoritme is gebasseerd op een master-slave structuur.———-plaatje!!——— De master van het netwerk wordt bepaald aan de hand van een random nummer tussen 0 en 1000. Er is gekozen voor deze structuur aangezien er gesynchroniseerd moet worden. Het doel is om de nodes tegelijk te laten knipperen, om dit te laten werken moet de interne klok van de nodes exact gelijk lopen of in dit geval stuurt de master naar alle andere nodes zijn tijd zodat daarop gecorrigeerd kan worden. Binnen deze master-slave structuur is de implementatie verdeeld in twee alternerende delen. In het eerste deel wordt vijf maal achtereen alternerend de methode delay() aangeropen en vervolgens een bericht verstuurd(De struct die hiervoor gebruikt is staat in de bijlage). In het tweede deel wordt er gesynchroniseerd, eerst wordt er voor 1000 milliseconden geluisterd of er berichten zijn ontvangen. Als er berichten zijn ontvangen worden deze verwerkt. Als laatste wordt er voor 1000 milliseconden geslapen en als laatste wordt er een bericht gestuurd. In dit bericht zitten drie elementen welke worden verstuurd.

- Eigen ID van de node
- De ID van de tot dan toe hoogst bekende ID, de eigen ID wordt verstuurd als er geen hogere bekend is

- De messageID houdt bij hoeveelste bericht wordt verstuurd.

De messageID is toegevoegd om te kunnen detecteren wanneer er loops voorkomen in het netwerk. In de komende twee subsecties zullen deze twee onderdelen verder toegelicht worden.

4.1 Het eerste deel, slapen en versturen

Het eerste deel bestaat uit alternerend slapen en berichten versturen.

```
int sendMax = 5;
int i = 0;
while(i<sendMax)
        sleepFor(1000)
        toggleLed();
        sendMessage();
        i++
```

De bovenstaande pseudo code wordt vijf maal herhaald voordat er door wordt gegaan met het tweede deel.

4.2 Het tweede deel, synchroniseren

Het tweede deel gaat over het synchroniseren van de nodes om ze tegelijk te laten knipperen. De synchronisatie gebeurt door de volgende acties:

```
listenFor(deltT-night)       //Listen for radio messages
if(message_received){
        toggleLed();
} else {
        toggleLed();
        sendMessage();
}
sleepFor(1000 milliseconds);
toggleLed();
sendMessage();
```

Op het moment dat er in de methode listenFor() een nieuw bericht wordt ontvangen zal dit bericht verwerkt worden in de huidige variabelen. De synchronisatie gebeurt op basis van een master-slave netwerk waarbij de master wordt gekozen op basis van zijn random nummer. Het is belangrijk om alleen naar de berichten te luisteren die van de master komen. Wanneer er een bericht van de master ontvangen wordt dan zal de tijd waarop het LEDje knippert aangepast. De deltaT wordt geïnitieerd met 1000 aan het begin van het programma en wordt elke keer aangepast aan de tijd waarop er een bericht van de master wordt ontvangen. Juist doordat elke node in het netwerk exact dezelfde code heeft werkt deze strategie omdat alle nodes uiteindelijk met dezelfde deltaT komen te zitten en dus na verloop van tijd allemaal tegelijk knipperen. Nadat

deze synchronisatie ronde is gedaan wordt het LEDje getoggled en zal er door de node geslapen worden voor 1000 milliseconden. Hierna zal het LEDje weer getoggled worden en er zal een nieuw bericht gestuurd worden naar de nodes in het netwerk die luisteren. Nadat deze synchronisatie ronde is voltooid zal er weer met het eerste deel overnieuw begonnen worden.

5 Testopstellingen en resultaten van de tests

Verschillende testopstellingen die getest kunnen worden:

- Er kan getest worden hoelang het gemiddeld duurt voor de nodes voor de eerste keer gesynchroniseerd zijn en hoeveel synchronisatie rondes dit neemt
- Testen wanneer de nodes dicht bij elkaar staan of ze beter synchroniseren dan wanneer ze ver weg staan. (Meten hoelang het duurt voor ze synchroniseren op verschillende afstanden. 1meter, 3meter, 5 meter, 10 meter)

Meetresultaten:

6 Conclusie

7 Aanbevelingen

Tijdens dit onderzoek was er slechts de beschikbaarheid over drie nodes. In een vervolgonderzoek kan er meer uitgebreid getest worden of het ook werkt met veel meer nodes. Vragen die hierbij gesteld kunnen worden zijn:

- Is er een maximum aan het aantal nodes in het netwerk waarop er nog gesynchroniseerd kan worden? Aangezien er gebruik wordt gemaakt van een random getal tussen 0 en 1000 is het van belang om te weten wanneer er vaak dubbele getallen voorkomen.
- veranderd de synchronisatie tijd die nodig is om te synchroniseren naarmate er meer nodes in het netwerk aanwezig zijn?
- De mogelijkheden onderzoeken om in plaats van een gedecentraliseerd netwerk te ontwikkelen een gecentraliseerd netwerk te synchroniseren. Gebruik makende van de code en hier enkele aanpassingen aan te maken.

8 Bronnen