UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

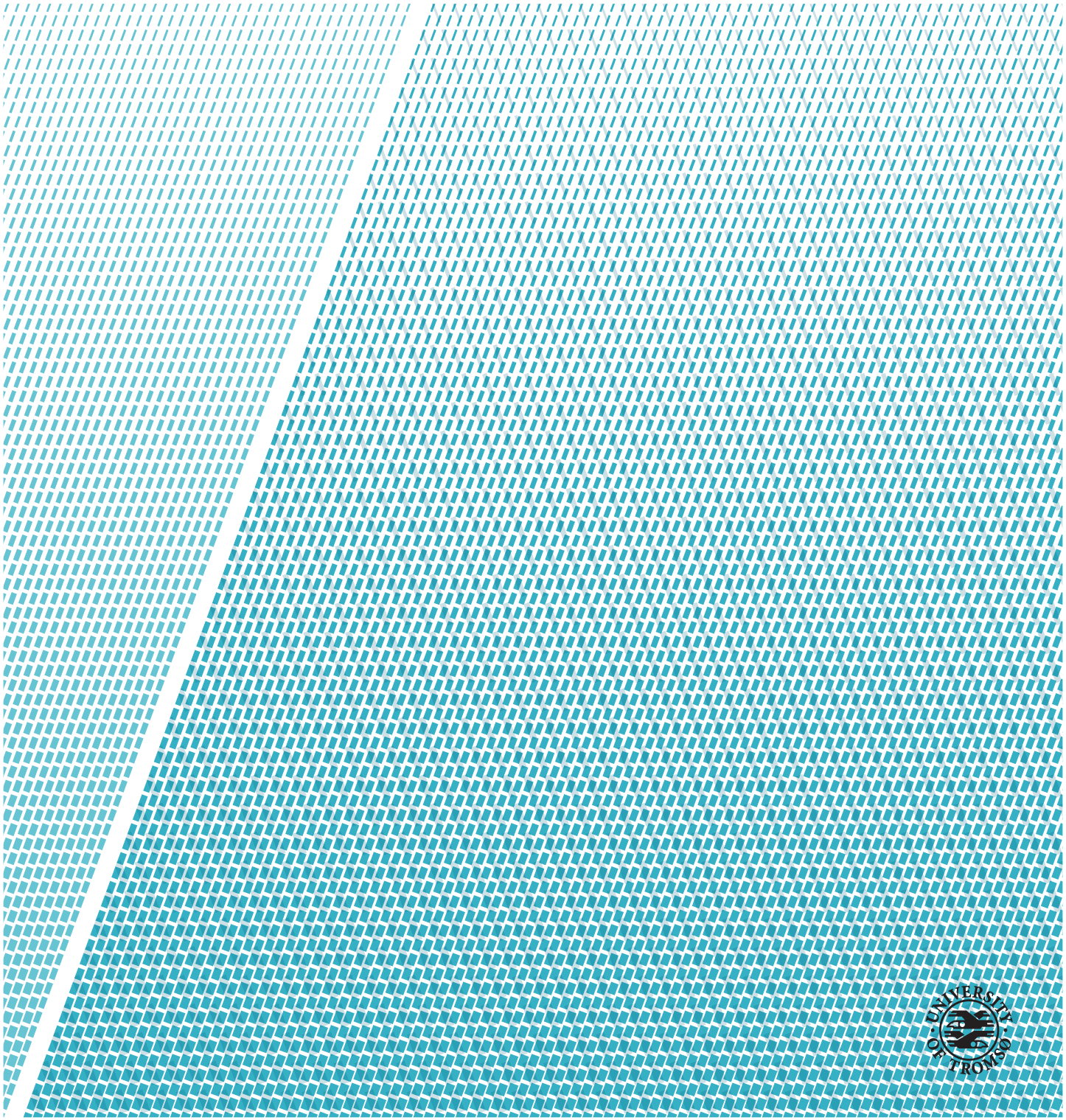
Faculty of Science and Technology
Department of Computer Science

Peer Observations of Observation Units

—

Camilla Stormoen

INF-3981 Master's Thesis in Computer Science ... May 2018



Abstract

What is wrong with the world? Motivation 1-3 sentences, Arch, Des, Imp, Exp 1,2-3 sentences, results and main conclusion.

Acknowledgements

Contents

List of Figures

List of Tables

1 Introduction

FRA CAPSTONE: *The Arctic tundra in the far northern hemisphere is challenged by climate changes in the world today and is one of the ecosystems that are most affected by these changes[10]. The Climate-ecological Observatory for Arctic Tundra (COAT) is a long-term research project developed by five Fram Center[1] institutions. Their goal is to create robust observation systems which enable documentation and understanding of climate change impacts on the Arctic tundra ecosystems. COAT was in autumn 2015 granted substantial funding to establish research infrastructure which allowed them to start up a research infrastructure during 2016-2020[10].*

WSN is a system that consists of hundreds or thousands of low-cost micro-sensor nodes. These nodes monitor and collect physical and environmental conditions. The various activities in the sensor nodes consume lots of energy and the battery of the sensor node is difficult to recharge in wireless scenarios and also because the sensor nodes are located at remote areas in the Arctic tundra.

This thesis presents the architecture, design and implementation of a peer observation that can observe and accumulate data from in-situ Observation Unit (OU)s.

1. http://www.framsenteret.no/english

1.1 Motivation

The motivation behind this project is...

The purpose is to fetch and accumulate data observed by OUs for further use.

1.2 Contributions

The dissertation makes the following contributions:

- An introduction to WSNs and routing techniques in WSNs.
- An implementation and description of a system ...
- An evaluation of the system.
- Thoughts on future work and further improvements to the current system.

1.3 Assumptions

Avgrense viktig!

1.4 Limitations

Avgrense viktig!

1.5 Outline

This thesis is structured into X chapters including the introduction.

Chapter 2 describes ..

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Chapter X

2

Routing Techniques in WSNs

WSN is a system that consists of hundreds or thousands of low-cost micro-sensor nodes which monitor and collect physical and environmental conditions. Figure 2.1 shows how a WSN architecture can be.

WSNs main task is to periodically collect information of the interested area and broadcast the information to a Base Station (BS). An easy approach to achieve this task is to make each sensor node transmit their data directly to the BS. But the problem is that the BS can be far away from the sensor node so a direct data transmission would not be possible, or if the routing path from the sensor node to the BS is long, the sensor node may/will die due to energy consumption. There are multiple hierarchical protocols that has been proposed as a solution to this problem.

2.1 Routing Protocols in WSNs

2.1.1 Hierarchical Routing

Hierarchical routing, also called cluster-based routing, is a well know technique for network routing with special advantages related to scalability, efficient

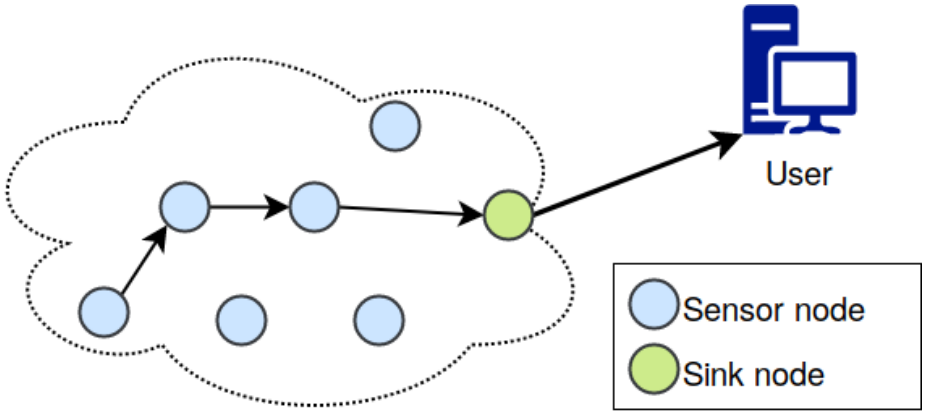


Figure 2.1: Figure shows an example of a WSN architecture.

communication and lower energy consumption in WSNs. Each cluster in the hierarchical routing protocol have a special node which is responsible for coordinating data transmission between all nodes in the cluster [1, 2, 4].

Two approaches based on hierarchical routing are Low-Energy Adaptive Clustering Hierarchy (LEACH)[1] and Power-efficient gathering in sensor information systems (PEGASIS)[5] which are described in Chapter 3.

2.1.2 Location-based Routing

In location-based routing, are nodes addressed based on their location. The distance between a node's and its neighbour can be estimated by incoming signal strengths, by GPS or via coordination among nodes [4]. Two approaches are Geographic Adaptive Fidelity (GAF)[17] and Geographic and Energy-Aware Routing (GEAR)[18].

2.2 Flood And Gossiping Protocol

2.2.1 Flooding Protocol

The simplest forwarding rule is to flood the network. In flooding, a node sends a packet received to all its neighbours expect the neighbours which sent the packet until the packet arrives at the destination or maximum number of hops for the packet is reached. Its biggest drawback is that protocol is not an energy efficient protocol and is not designed for sensor networks [16].

2.2.2 Gossiping Protocol

Gossiping is a modified version of flooding attempting to correct some of its drawback. In gossiping, a node with updated data will contact an arbitrary node. However, it is possible that the contacted node was updated by another node and in that case may not spreading the update any further [19].

3

Related Work

To improve the overall energy dissipaton of WSNs, LEACH [1] introduce a hierarchical clustering algorithm for sensor networks. It is self-organized and use randomization to distribute the energy load evenly among the sensors in the network. The sensor nodes organize themselves into local clusters where one node is the local BS or CH. The CH are not fixed to avoid nodes to drain their battery and to spread the energy usage over multiple nodes. The nodes self-elect a new CH depending on the amount of energy left at the nodes at different time-intervals. LEACH is divided into different rounds where each round include a setup phase and a steady-state phase [3]. In the setup phase will each node decide whether to become a CH or not. When a CH is chosen, each node will select its own CH based on the distance between the node and the CH and join the cluster. In the steady-state phase will the CH fuse the received data from the node members in the cluster and send it to BS.

In diversity, will nodes in our approach first connect to a cluster and then start a CH election rather than elect a CH first and then nodes joining the cluster. A resemblance between the two approaches is that neither of them consider a nodes energy level when calculating the CH.

LEACH do not consider a node's energy level when calculating the CH. It has therefor been a benchmark for improving algorithms such as the centralized clustering algorithm LEACH-C [9] and distributed clustering algorithm such as LEACH-E [11] and LEACH-B [12]. They concentrate on energy consumption reducing a nodes residual energy and more relevant criterions [8].

Fuzzy-LEACH (F-LEACH) [6, 7] have three different fuzzy descriptors such as energy, concentration and centrality used to complement the cluster head selection process. The BS performs the CH election in each round by computing the chances of a node becoming a CH by calculating the three fuzzy descriptors. F-LEACH also assumes that the BS elects the appropriate CH because it has a complete information about the whole network.

In contrast to F-LEACH will our approach elect a CH by the nodes in the cluster and not in a BS. The CH election will not consider "variables" such as battery level or number of nodes in the cluster.

PEGASIS is a chain-based protocol with the idea to form a chain among the sensor nodes so each node will receive and transmit data to a close neighbor. The sensor nodes will also take turns on being the leader for transmitting data to the BS and therefor distribute the energy load evenly among the sensor nodes. The chain can be organized by the nodes themselves using a greedy algorithm starting from some node or the BS can compute the chain and broadcast it to all the nodes in the network [5].

To increase the robustness of devices and lower power consumptions, ZebraNet [13] provides a low-power wireless system for position tracking of wildlife by using peer-to-peer network techniques. This reduces the researchers effort to manage the sensors and collecting logged data for their research. The radios on their devices also operates on different frequencies and have different bandwidth, range and other characteristics.

A diversity to our approach is that ZebraNet stores multiple copies of the same data across multiple nodes while our approach forwards the data to the next node in the path to leader.

Gossip-based protocols, or epidemic protocols, are popular protocols due to their ability to reliably pass information among a large set on interconnected nodes. Jelasity et al. [14] provide a Gossip-based communication protocol (GBCP) where each node have peers to gossip with in a large-scale distributed system [10]. These nodes can quickly join and leave the network at any given point of time. The general principle of their framework is that every node (1) maintains a relatively small local membership table that provides a partial view of all nodes and (2) periodically refreshes the table using a gossiping procedure.

The difference between GBCP and our approach is that our approach does not use a gossip protocol to update its table of nodes, but instead relies on the communication with other nodes to know about the election of a new CH, which of the node is the CH and when a node should accumulate data and

send it to the CH.

4

Idea

Technical idea

5

Architecture

This chapter describes the architecture of the system. The main functionality can be divided into 3 sub-sections: a nodes lookup service, discovery of other nodes and incoming network requests. The architecture of the system is presented in Figure 5.1.

5.1 Node Lookup Service

The lookup service is responsible for storing a list of all previous discovered nodes and their address.

5.2 Discovery Of Other Nodes

Each node can only discover other nodes within a simulated radio range. Figure 6.2 shows how a simulated radio range of a node may be discovered. When a new node in the network has been discovered, meta-data about the node such as address be stored locally on the device in the cluster.

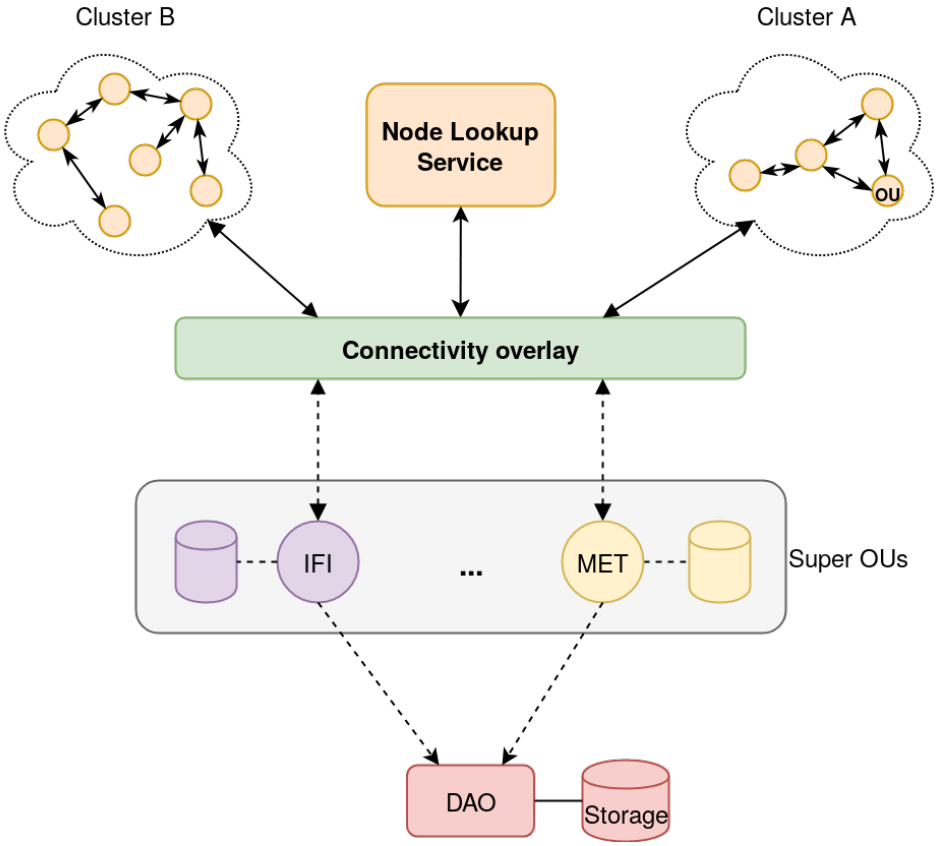


Figure 5.1: Figure shows the architecture of the system.

5.3 Incoming Network Requests

A node may receive incoming requests from other nodes in the network. The request handler will handle the request based on the type of request. The types of requests a node may receive are listed below.

5.3.1 Connect To Neighbours

When a node receive a list of neighbours in range from the lookup service, it will try to connect to the neighbours that are within the nodes range. It will only connect to the neighbour node if it receives a OK-message.

Receive OK From Neighbours

When a node receives a OK-message it will connect itself to the neighbour. The neighbour will also then be connected to the new node.

5.3.2 Cluster Head Election Request

A node may receive a CH election request when a node has joined the cluster.

Cluster Head Election Request

When a node receives a CH election request it will perform a leader election and forward the result to it's neigbours which will do a leader election as well.

Cluster Head Election Calculation Request

If there is a leader in the cluster already and a new election should be proposed, a CH election calculation request is sent from the leader. Nodes receiving this request will calculate their CH election number.

5.3.3 Data Transmission

Notify Neighbours About Sending Data To Leader

A node may receive a request that it should send its data to the leader. This request is forwarded to the nodes neighbour and so on.

Send Data To Leader

This request forwards a nodes data to the next node in the path to the leader. If the nodes data receiving this requests isn't sent to the leader of the cluster, will the data be accumulated with the received data and then forwarded to the next node.

5.4 Data Accumulation/Gathering?

6

Design

In this chapter we will look at the design of the system and present the design of each component of the architecture. Figure 6.1 shows how the cluster network may appear (in the system). Nodes are connected to other nearby nodes represented by arrows and together they form a cluster network.

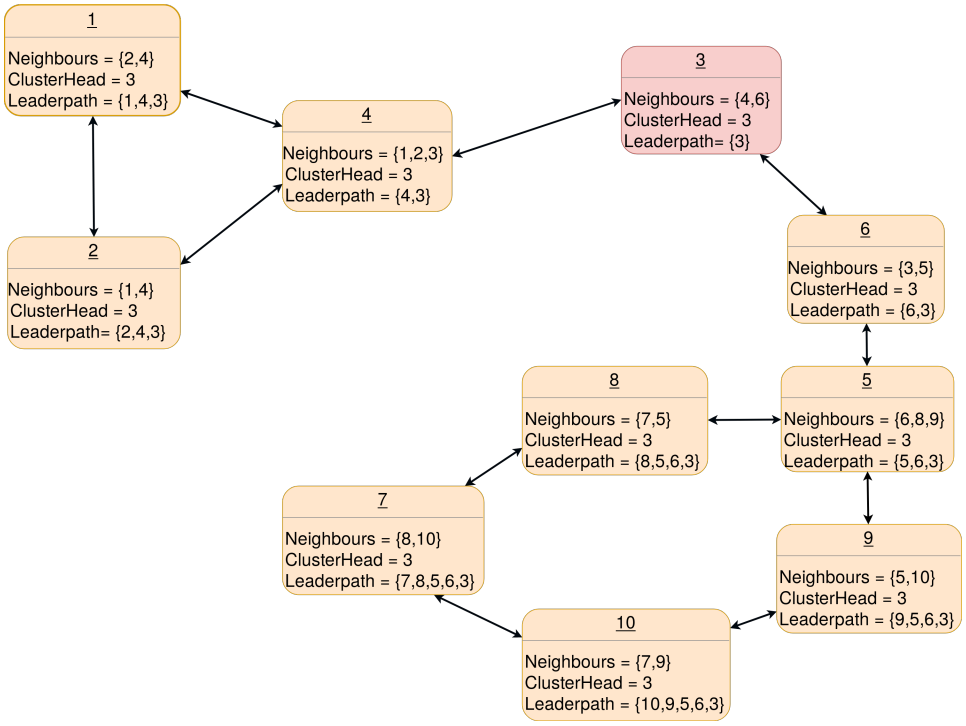


Figure 6.1: Figure show design of the system.

6.1 Discovery Of Other Nodes

6.1.1 Broadcasting

When a new node start, it will contact the node lookup service to discover other nodes in the network. The node will then initiate a broadcast. Broadcast is limited due to a radio range limitation where only nodes that are within this range will receive the broadcast, shown in Figure 6.2. *Node 1* will only reach *node 5* and *node 7*.

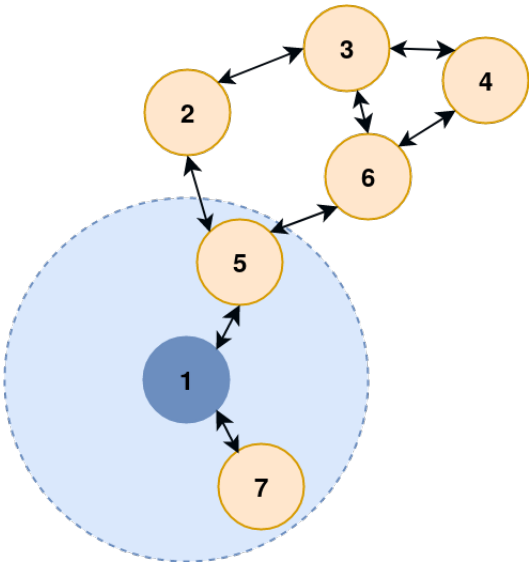


Figure 6.2: Figure show broadcast range of a OU.

6.2 Cluster Head Election

A CH election may occur in two scenarios listed below.

6.2.1 New Node In Cluster Starts Election

When a new node has joined the cluster, it will start a new CH election. It will calculate it's own CH-score and gossip the score to its neighbours. The neighbours will then start a CH election comparing the received CH-score against their own CH-score. The result of the CH election will then be gossiped to the node's neighbours with either the received CH-score or the nodes own CH-score. The gossiped message also contains a path to the leader. The node append its own address to the path if the received CH-score won the election, otherwise will the path to leader only contain the node itself. Eventually a new CH is elected by all the nodes and the CH-election result will be consistent in the whole cluster. An example of a CH election is shown in Figure 6.3.

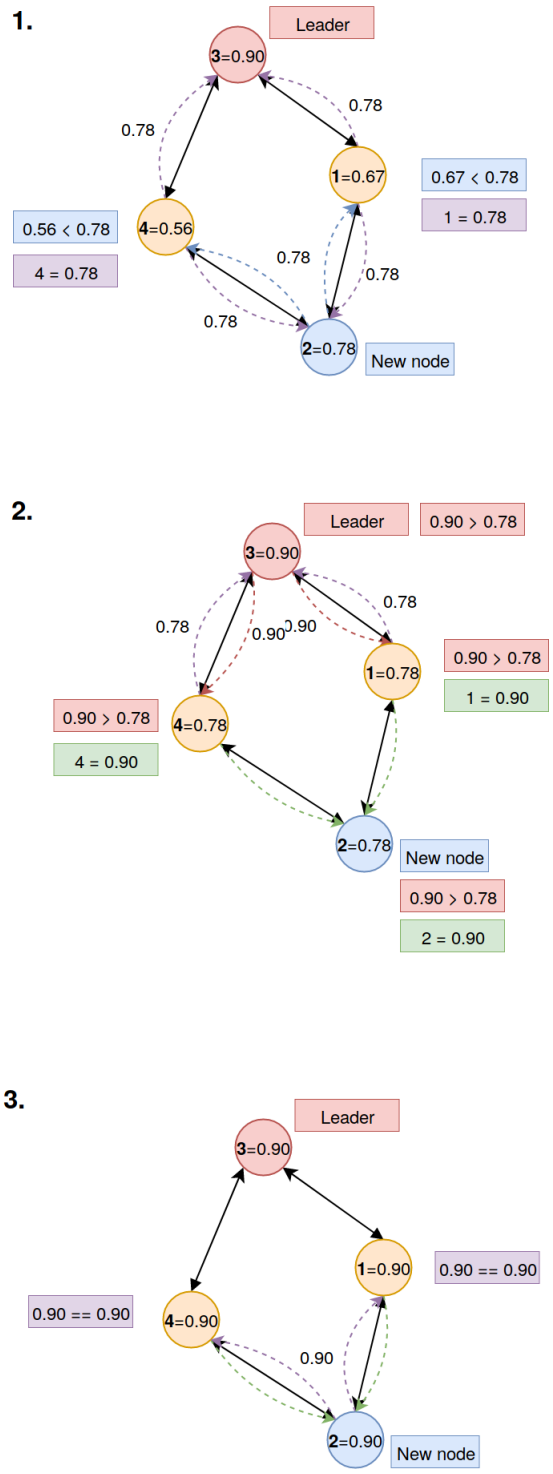


Figure 6.3: Figure show how a new node starts a leader election.

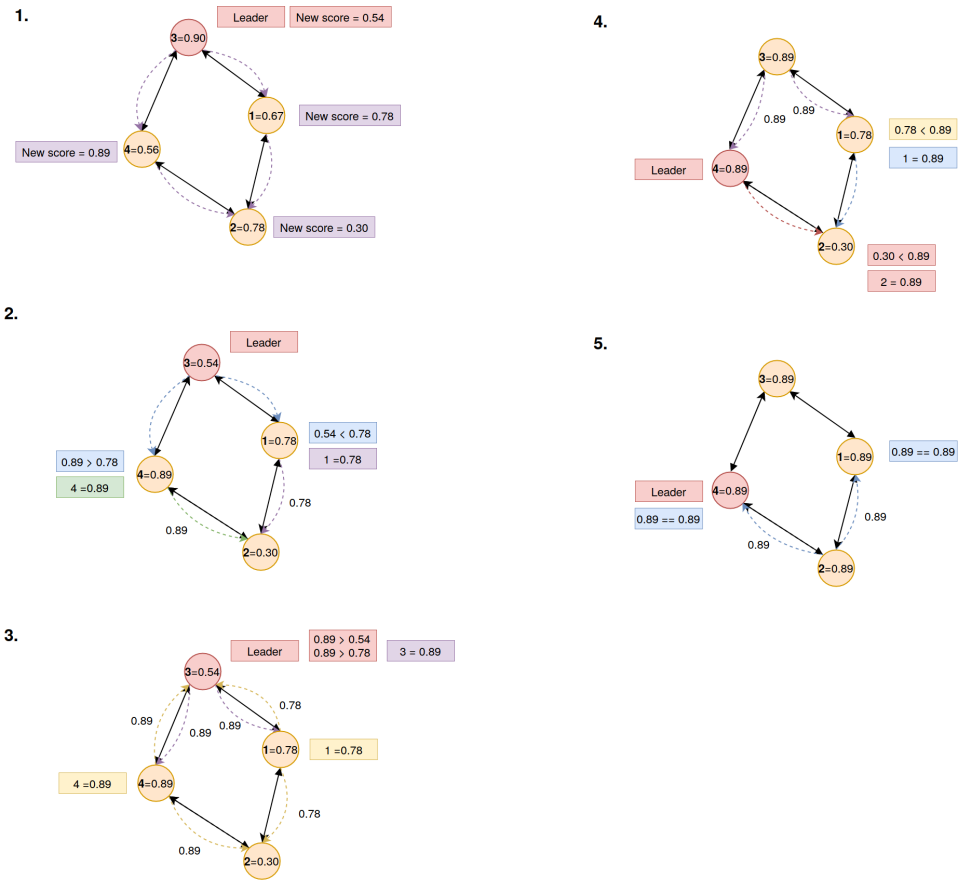


Figure 6.4: Figure show how a new node starts a leader election.

6.2.2 Cluster Head Starts Election

When a CH has accumulated data a certain times, it will start a new CH election. Initially, the CH will gossip a message to nodes the cluster to calculate a new CH-score. Then will the CH start a new election and gossip this election to the other nodes. The nodes receiving this gossip message will start their leader-election as described in the section above. The election is illustrated in Figure 6.4.

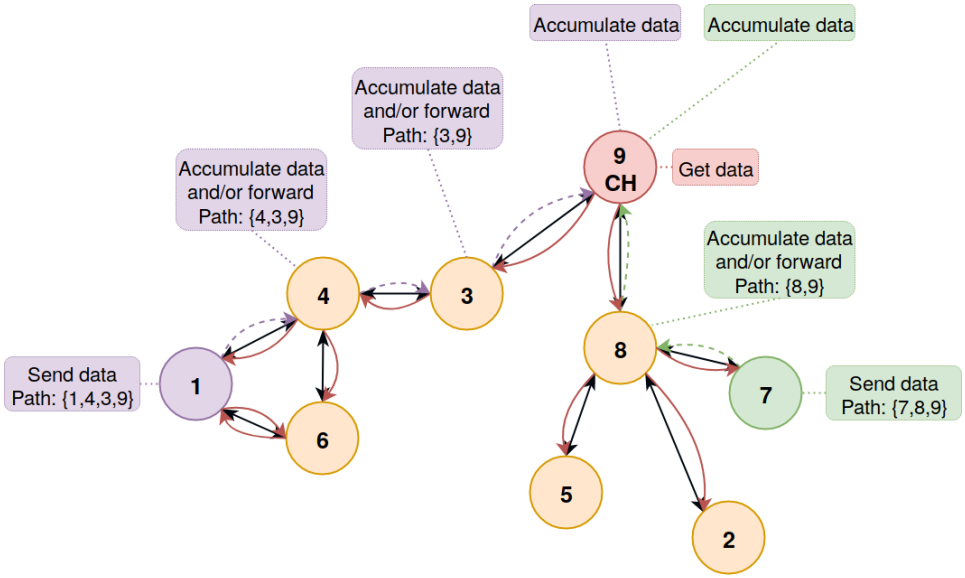


Figure 6.5: Figure show how CH gossips a GET-data request and how nodes sends their data to the leader through the nodes in the path.

6.3 Data Accumulation

A CH will start gathering data by gossiping a message to nodes in the cluster, illustrated as the red arrows shown in Figure 6.5. When a node receives this message it will send its data to the next node in the path to the CH as Figure 6.5 shows. When a node receive data from another node it will check if it's own data has been sent to the CH or not by a fingerprint and a status. If it hasn't been sent earlier, the node will accumulate the received data with its own data, and then send the data to the next node in the path to CH. The CH will accumulate all received data and store it.

7

Implementation

This chapter will elaborate on how we implemented the system, general implementation requirements, issues and choices.

The system is implemented in the open source programming language GO 1.9.3[1] and the nodes communicate with each other by Golangs HTTP Server which listens on the Transmission Control Protocol (TCP) network. This was a practical choice because we were familiar with the language from previous projects, its developed for making concurrent mechanisms easy and to get the most out of multicore and networked machines and offers multiple different libraries.

It is not suitable to have nodes deployed in the Arctic Tundra at such an early development. This implementation simulates a real-life environments where nodes can communicate with each other through the TCP network. Each node is assigned *x,y* coordinates within a network grid size and a broadcast width, as Table 7.1 show.

1. https://golang.org/

Parameter	Value
Number of nodes	100
Network grid size	500 x 500
Node broadcast width	100

Table 7.1: Parameters of simulation

7.1 Distance To Other Nodes In The Network

A new node will contact the lookup service to discover other nodes in the network by sending a POST-request containing meta-data about itself such as address and position (*x,y* coordinates).

The distance formula 7.1, also called Euclidean distance, is used to calculate the range between two points in a two-dimensional coordinate map and is used to see if a node is within a simulated radio range or not, as seen in Figure 6.2. The two points to be calculated is the node itself together with all the running nodes in the network.

$$d = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2} \tag{7.1}$$

Figure 7.1 shows how a node contact the lookup service where the node's position is calculated and nodes within the simulated radio range is returned to the node. At last, the node will try to connect to the reachable nodes.

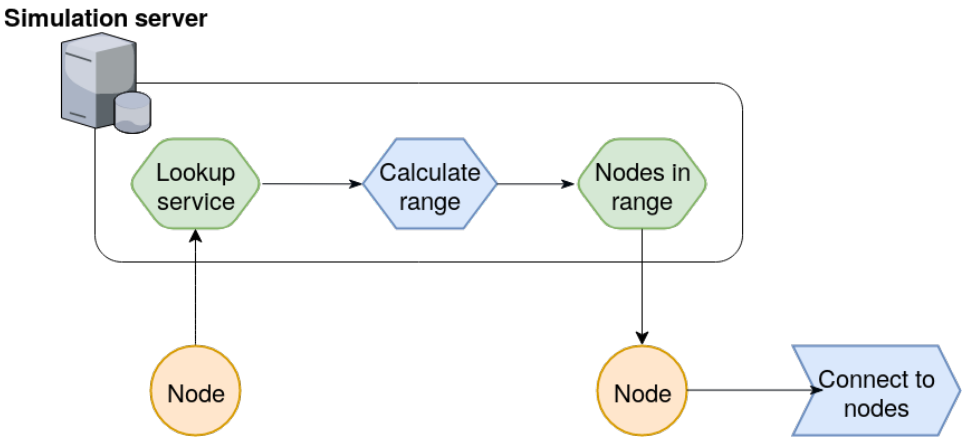


Figure 7.1: Broadcast simulation

7.2 Connect To Neighbours

A node will only be able to connect to another node in the network if the node accept the request to be neighbours. Right now, every node will be able to connect with their neighbours, as long as the node isn't unavailable by means of sleeping or if it's dead. There is no restriction in number of neighbours a node can have or that a node receiving a request from a new neighbour must forward the request to the CH and let the leader decide whether the new node can connect to the neighbour or not.

7.3 Cluster Head Election

A CH election is proposed either when a new node joins the network or by a CH. The CH election algorithm is based on the Bully algorithm where the node with the highest ID number is selected to be the leader. Our approach do not use the node with highest ID number, but elect the node with highest score between 0 and 1 to be a CH. Each node makes its own decision about whether to become a CH or not depending on the received score from neighbour nodes. Our apporach have several improvements discussed further in Section 9.2.1.

7.4 Minimize Path To Leader

Since CH elections are gossiped from all nodes, can each node receive multiple gossip's per CH election and some paths will be longer than others. In order to minimize the path to the CH, will a node when receiving an election compare its existing path to the one received [20].

7.5 Data Accumulation

Since all nodes run concurrently and independently, can each node receive multiple requests from different neighbouring nodes. This especially arise in the CH when collecting data from the nodes in the cluster. The collected data is stored in a map and maps in Go are not safe for concurrent use. If a map is read from and wrote to from concurrent goroutines, the access must be compromised by a synchronization mechanism. One of the most common ways to protect maps is by using mutexes as illustrated in Listing 7.1.

Listing 7.1: Small Go program showing how mutexes are used when updating a map

```
/*SensorData is data from "sensors" on the OU*/
type SensorData struct {
        ID          uint32
        Fingerprint uint32
        Data        []byte
}

/*DBStation is a strucure that contains
  a map that store data at CH */
var DBStation struct {
        sync.Mutex
        BSdatamap map[uint32][]byte
}

func sendDataToLeaderHandler() {
        var rd SensorData

        DBStation.Lock()
        defer DBStation.Unlock()

        (...)

        err := json.Unmarshal(body, &sData);
        if err != nil {
                panic(err)
        }

        (...)

        /*Update map in key FingerPrint
          with received data */
        DBStation.BSdatamap[rd.Fingerprint] = rd.Data
}
```

7.6 Data Transmission?

8 Evaluation

This chapter describes the experimental setup and metrics used to evaluate the implemented system.

8.1 Experimental Setup

All experiments were done on a Lenovo ThinkCenter with the following specifications:

- Intel® CoreTM i5-6400T CPU @ 2.20GHz × 4
- Intel® HD Graphics 530 (Skylake GT2)
- 15,6 GiB memory and 503 GB disk
- Ubuntu 17.04 64-bit with gcc V6.3.0 compiler and GO 1.9.3

8.2 Experimental Design

8.2.1 CPU Measurements

8.2.2 Memory Measurements

8.2.3 Network (Socket) Usage

8.2.4 Network Lifetime?

8.3 Results

In this section we will discuss the results of the testing described in the sections above.

8.3.1 CPU Usage

8.3.2 Memory Usage

8.3.3 Network Usage

8.3.4 Network Lifetime?

9 Discussion

This chapter discusses our approach, experience, how we solved the problem and why we chose the solution we ended up with..

9.1 Availability of nodes in the system

9.2 Cluster Head Election

9.2.1 Cluster Head Calculation

At the moment, the CH election is simply based on which node has the highest score between 0 and 1. Despite the working approach, it does not take in account many other realistic aspects which would improve the CH election. To approve this approach and make it more realistic, we could have used several sub-factors listed below:

- Number of nodes between a node and CH
- Number of neighbours for the node
- Power left on node

- Bandwidth - WiFi, LoRaWan[1], cable/Ethernet?
- Prior history
- Traffic on node

9.3 Path To Leader

9.4 Data Transmission

9.4.1 Data Accumulation

9.5 Base Station Access

1. https://lora-alliance.org/about-lorawan

10

Conclusion

In this thesis, we have implemented a system/prototype...

Our experiments showed that the system ...

11

Future Work

12

Appendix

Bibliography

[1] W. R. Heinzelman and A. Chandrakasan and H. Balakrishnan, *Energy-efficient communication protocol for wireless microsensor networks*, 2000, in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 10 pp. vol.2-*.

[2] K. Latif and M. Jaffar and N. Javaid and M. N. Saqib and U. Qasim and Z. A. Khan, *Performance Analysis of Hierarchical Routing Protocols in Wireless Sensor Networks*, 2012, in *2012 Seventh International Conference on Broadband, Wireless Computing, Communication and Applications, pp. 620-625*.

[3] Z. Han and J. Wu and J. Zhang and L. Liu and K. Tian, *A General Self-Organized Tree-Based Energy-Balance Routing Protocol for Wireless Sensor Network*, 2014, in *IEEE Transactions on Nuclear Science Vol.61, Nr.2, pp. 732-740*.

[4] J. N. Al-Karaki and A. E. Kamal, *Routing techniques in wireless sensor networks: a survey*, 2004, in *IEEE Wireless Communications Vol.11, Nr.6, pp. 6-28*.

[5] S. Lindsey and C. S. Raghavendra, *PEGASIS: Power-efficient gathering in sensor information systems*, 2002, in *Proceedings, IEEE Aerospace Conference Vol.3, pp. 3-1125-3-1130*.

[6] A. K. Mishra and R. Kumar and J. Singh, *A review on fuzzy logic based clustering algorithms for wireless sensor networks*, 2015, in *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), pp. 489-494*.

[7] Indranil Gupta and D. Riordan and Srinivas Sampalli, *Cluster-head election using fuzzy logic for wireless sensor networks*, 2005, in *3rd Annual Communication Networks and Services Research Conference (CNSR'05), pp. 255-260*.

[8] Maryam Sabet and Hamid Reza Naji, *A decentralized energy efficient hierarchical cluster-based routing algorithm for wireless sensor networks*, 2015, in *AEU - International Journal of Electronics and Communications Vol.69, Nr.5, pp. 790 - 799*.

[9] W. B. Heinzelman and A. P. Chandrakasan and H. Balakrishnan, *An application-specific protocol architecture for wireless microsensor networks*, in *IEEE Transactions on Wireless Communications Vol.1, No.4, October 2002, pp. 660-670*.

[10] Demers, Alan and Greene, Dan and Hauser, Carl and Irish, Wes and Larson, John and Shenker, Scott and Sturgis, Howard and Swinehart, Dan and Terry, Doug, *Epidemic algorithms for replicated database maintenance*, in *ACM: Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing, 1987, pp. 1–12*.

[11] Chen, Bai and Zhang, Yaxiao and Li, Yuxian and Hao, Xiao-Chen and Fang, Yan, *A Clustering Algorithm of Cluster-head Optimization for Wireless Sensor Networks Based on Energy*, in *Journal of Information and Computational Science, Vol.8, 2011*.

[12] M. Tong and M. Tang, *LEACH-B: An Improved LEACH Protocol for Wireless Sensor Network*, in *J2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM), 2010, pp. 1-4*.

[13] Juang, Philo and Oki, Hidekazu and Wang, Yong and Martonosi, Margaret and Peh, Li Shiuan and Rubenstein, Daniel, *Energy-efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet*, in *ACM: SIGARCH Comput. Archit. News, Vol.30, No.5, December 2002, pp. 96–107*.

[14] Jelasity, Márk and Voulgaris, Spyros and Guerraoui, Rachid and Kermarrec, Anne-Marie and van Steen, Maarten, *Gossip-based Peer Sampling*, in *ACM Trans. Comput. Syst., Vol.25, No.3, August 2007*.

[15] Draves, Richard and Padhye, Jitendra and Zill, Brian, *Routing in Multi-radio, Multi-hop Wireless Mesh Networks*, in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking, 2004, pp.114–128*.

[16] Holger Karl, Andreas Willig, *Protocols and Architectures for Wireless Sensor Networks*, John Wiley & Sons, Ltd, 2006.

[17] Xu, Ya and Heidemann, John and Estrin, Deborah, *Geography-informed Energy Conservation for Ad Hoc Routing*, in *MobiCom '01: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, 2001, pp.70–84*.

[18] Yu, Yan and Govindan, Ramesh and Estrin, Deborah, *Geographical and Energy Aware Routing: a recursive data dissemination protocol for wireless sensor networks*, in *UCLA Computer Science Department Technical Report, Vol.463, 2001*.

[19] Tanenbaum, Andrew S. and Steen, Maarten van, *Distributed Systems: Principles and Paradigms (2Nd Edition)*, Prentice-Hall, Inc., 2014.

[20] Dijkstra, E. W., *A note on two problems in connexion with graphs*, in *Numerische Mathematik, Vol.1, No.1, December 1959, pp.269–271*.