2016

Analysis of Structured and Un-Structured Network Protocols for Data Aggregation Over Distributed Wireless Sensor Networks

Priyashraba Misra

Analysis of Structured and Un-Structured Network Protocols for Data Aggregation Over Distributed Wireless Sensor Networks

Priyashraba Misra

Thesis submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Electrical Engineering

Vinod K. Kulathumani, Ph.D., Chair
Yanfang Ye, Ph.D.
Yaser P. Fallah, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2016

Keywords: Wireless Sensor Networks, Data Aggregation, Structured and Unstructured Protocols, Aggregation Tree Protocol, Census

Abstract

Analysis of Structured and Un-Structured Network Protocols for Data Aggregation Over Distributed Wireless Sensor Networks

Priyashraba Misra

The focus of this thesis is on design and evaluation of one-shot data aggregation protocols for static and mobile wireless sensor networks (WSNs). The goal in one-shot data aggregation is to compute a statistical summary of sensor data such as max, average, sum, count and min, when initiated by a special node such as the base station. WSNs have wide range of applications in both static and mobile/dynamic systems. Static sensor networks are especially useful when monitoring is required in harsh, inaccessible environments and when the region to be monitored is really large. Examples of static sensor network applications include environmental monitoring systems, monitoring of industrial control systems, monitoring of degradation in slagging gasifiers, distributed object detection and tracking. Example of mobile applications include vehicular ad-hoc networks and networks of personal radios used in emergency dispatch and battlefields.

For data aggregation in static networks with stable links, structured approaches such as spanning trees are generally preferred. This is because, once a data aggregation structure has been established, link topologies remain fixed and there is minimal need to actively maintain and change the routing structures. In this thesis, one such tree based data aggregation protocol has been designed and evaluated using simulations in networks ranging from 100-1000 nodes. The protocol has also been implemented at a smaller scale in the context of a smart refractory environment, where slag penetration in gasifiers is remotely monitored using smart bricks that are embedded with sensors.

In mobile networks and networks with frequent link changes, topology driven structures are likely to be unstable and to incur a high communication overhead. Therefore, self-repelling random walks have been recently proposed as an attractive alternative for data aggregation in mobile systems. In this thesis, a brief overview of random walk based data aggregation has been presented and systematic evaluation of tree based and random walk based data aggregation protocols in networks ranging from 100-1000 nodes under varying degrees of node mobility has been done. The conditions under which unstructured protocols become more attractive in terms of convergence time and messaging efficiency as compared to tree based structured approaches have been quantified.

Acknowledgements

First, I want to thank my committee chair and advisor, Dr. Vinod K. Kulathumani, for guiding me in my research and providing me the opportunity to work with him and his other graduate students. This thesis work has been made possible with his constant support and guidance.

I also want to thank Dr. YangFang Ye and Dr. Yaser P. Fallah for being a part of the my committee. I have had discussions with them which were important in my understanding of identifying and solving certain research oriented problems in my Thesis.

I extend my gratitude towards the entire team of the Smart Refractory Systems project. Work in this thesis was partially supported by Department of Energy under Award Number DE-FE0012383.

I would like to thank my current co-workers for helping me out in collecting data for this work. They have been extremely helpful, supportive in building my understanding of the subject. I've learnt loads from them in discussions with them.

Last but not the least, I want to express my gratitude to my family. My parents have been very encouraging on my decision to go to grad school. Their support has been relentless and a constant motivation to my desire of pursuing Electrical Engineering in school.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview and objectives

Fueled by recent advances in MEMS technology, embedded systems and wireless networking, wireless sensor networks have emerged as an active area of research in the past two decades [1][2][3] with applications in different areas such as environmental monitoring, object tracking, smart agriculture, industrial process control, vehicular networks and surveillance. In many of these applications, the nodes are static. Examples include environmental monitoring and perimeter surveillance applications. Such systems are especially useful when the regions to be monitored are inaccessible and remote, and the events that need to be monitored have to be reported to a special node such as a base station with minimal delay. Wireless sensor networks which self-configure themselves into a multi-hop network are well suited for such applications. A key characteristic of static sensor networks is that the network topology does not change much over time. It is possible that link quality can change over time and occasionally some links may get worse or better, but the amount of link dynamics is pretty low. There are also applications in which the nodes are mobile. Examples include vehicular ad-hoc networks (VANETs) in which data from sensors embedded inside vehicles are exchanged with other vehicles. Another example consists of mobile radios used in military scenarios for communicating between personnel. In such networks, the network topology is constantly evolving. If the network is modeled as a graph, then the set of edges of the graph is constantly changing. This makes it harder to design protocols for aggregating

information.

A basic mode of operation for sensor networks is data collection from a large number of sensors that self-configure themselves into a multi-hop network and transmit data to a base station. Numerous protocols have been developed for various tasks such as data collection, parameter broadcasts, wireless reprogramming, data-driven routing, querying, unicast etc. The key focus in this research has been on maximizing data throughput, i.e., maximizing information communicated with minimal control overhead, and in improving the energy efficiency of the system. Energy efficiency is critical because nodes are mostly battery powered and radio communication consumes significant battery life. Hence the goal is to reduce the amount of radio communication involved.

In this thesis, my focus is on data aggregation. This is slightly different from data exfiltration where one is interested in individual data from all sensors. On the other hand data aggregation focuses on metrics such as max, average, sum, count that is computed over data from all sensors over a given interval of time. These metrics are duplicate sensitive and therefore data from each sensor can be aggregated only once.

A common approach to data aggregation in static networks is by using structures such as trees. Examples of such protocols include directed diffusion, TAG and CTP (described in greater detail in related work Section). The fundamental idea is to create a tree rooted at the base-station which is used to send data back from all sensors. Note that although we are only interested in aggregates, to ensure no duplicates one needs to include each sensor's information individually to the base station. Structured protocols such as this either need to periodically refresh the tree or actively maintain the tree structure. When the intervals between successive data aggregation is large, one prefers periodic refreshing of the tree structure, which is what we pursue in this thesis. When data requirements are very sparse, maintaining trees can have a high message overhead. The nodes need to continuously transmit control beacons in order to keep the tree stable and fix link failures. On the other hand, in periodic refreshing the network forms the tree when the base station requests for data. Data requests from base stations are floods which contain essential information for tree formation. As soon as a node has a parent it schedules its data transfer. After data transfer is complete the node does not have to maintain the tree till the next data

request comes in, unless the network is so dynamic that the topology has evolved before data communication is finished (this is possible in mobile systems). In this thesis, We have designed and implemented one such tree based protocol for data aggregation and we have evaluated under simulations in ns-3.

As an example application of the static data aggregation, in this thesis we have considered a smart refractory system. Refractory is a temperature furnace in which temperatures can reach upto 1400 degree Celsius. It is important to monitor in real-time if slag has penetrated such furnaces so that timely maintenance can be carried out. Traditional methods use thermo-couples to measure the temperature inside the furnace. Access ports on the wall of the furnace are used to put the thermocouple in. This methods has drawbacks as high temperature, harsh working conditions and also slag can cause the sensors to degrade very fast. As part of a interdisciplinary project with the Mechanical Engineering Department and Chemical Engineering Department at West Virginia University, our research group is working on designing smart bricks with embedded temperature sensors to address the remote monitoring of such furnaces (details are described in Chapter 3). In this thesis we have used this as my focus application and have utilized the tree based protocol for data aggregation in this project. Results from this case study are presented in Chapter 5.

But tree based protocols for data aggregation have problems in mobile networks or dynamic networks in which the links keep changing. The tree structure is the most important aspect of such protocols. Link changes being frequent in mobile networks result in unstable topology and high communication overheads. With increasing network size, the probability of a path to fail is likely to increases. These problems are further aggravated by even small node speeds, resulting in path failure and instability. When broken paths are detected they have to be fixed, more importantly at a rate faster than the rate of breaking. This involves high message overheads. So structure free approaches are desired for mobile systems.

Structure-free approaches for data aggregation have not received as much attention as structured protocols. There are some popular ideas such as flooding and gossiping which are un-structured. However, they are very inefficient when used for data aggregation. This aspect is analyzed in more detail in Chapter 4. However, more recently, self-repelling random walks have been proposed as an attractive alternative for structure based approaches in

mobile networks [4]. The protocol is simple and efficient and requires very little overhead. In this thesis, we have used self-repelling random walks as the example for unstructured data aggregation protocol and used it to compare against tree based protocols.

> The objective of this thesis is to systematically compare structured an non-structured techniques for data aggregation in terms of latency, messages and reliability under static and mobile network scenarios

1.2 Summary of contributions

1. We have developed a tree based protocol for one-shot data aggregation. When the base station needs data from the nodes it initiates tree formation. When nodes have a parent, they schedule their data transmission. In mobile networks, the topology can constantly change. Hence periodically the trees are refreshed until the data aggregation is complete. We have evaluated this protocol using simulations in ns-3.

2. We have utilized the tree-based protocol for data aggregation in smart refractories and evaluated prototypes of this system in a refractory environment using Telos motes at a small scale.

3. As an example of structure-less protocol we have chosen Push-pull self-repelling random walk protocol. We have simulated it in ns-3 and used it as a benchmark for comparing with structured protocols.

4. We evaluated the tree based protocol and push-pull self-repelling random walks protocols systematically under varying mobility, density and network size. Parameters were carefully chosen for impartial comparison of the two protocols.

1.3 Organization of rest of the thesis

The remaining part of this document is arranged as follows. Chapter 2 covers the background and related work on data aggregation protocols. Design of the tree based protocol used in my thesis is discussed in Chapter 3. In chapter 4, the design of the structureless

protocol based on self-repelling random walks is discussed. The 5th chapter talks about results and analysis. The final chapter, summarizes the conclusion and possible future work.

Chapter 2

Background work and existing literature

In this chapter we talk about existing literature for both structured and unstructured protocols. In the first section we try to understand how structured protocols work. Some commonly used structured protocols like Directed diffusion[5], TAG[6] and CTP[7] are discussed here. We also consider one structured protocol, CTP and systematically analyse it to learn how structured protocols are designed. Section 2, briefly talks about existing research for structureless protocols.

2.1 Structured Protocols

In the last two decades extensive study has been done on designing protocols for data aggregation for networks. As a result of improvement in hardware technology, we now have low-cost sensor nodes which have high computing power, more memory, multiple sensing ability and improved radio communication systems. Sensing motes are independent devices and have limited source of power. So it is important to design protocols which can harness high computing capabilities of the nodes and at the same time make them energy efficient.

Most structure based protocols follow similar design patters. The nodes arrange themselves into structures, like trees and maintain the tree. Nodes may send their data to the base station periodically or when they receive data requests from base station. Now, let us

see how some of the common structure based protocols work.

Directed Diffusion is a data centric protocol in which nodes request data by sending *interests* for named data[5]. Data matching the interest are then routed towards the node. Attribute-value pairs are used to name tasks. It uses a one hop at a time mechanism to send the messages to sensor nodes. Datagrams, gradients, reinforcements and interests are important concepts of directed diffusion. The queries from the base station are flooded across the network as interest for named data. The process is called reinforcement. The flood helps set up gradients in the network. This gradient is used by the nodes to route the data back to the base station.

Interest caches are maintained by nodes across the network. They have information about the query and the one hop neighbour. They also maintains gradient fields for each of its neighbours. A node may use the cache to reject duplicate interests.

When the base station needs a data message, it reinforces one of its neighbours, and it in turn reinforces one of its neighbours in the next level. This continues till the reinforcement message reaches a desired node. Reinforcement refers to flooding the data request down the tree.

Madden et. al. in their paper on Tiny Aggregation Service for Ad-Hoc Network Service[6] (TAG), talk about developing aggregating services for motes. They implemented TAG, a declarative interface for data collection and aggregation. It tries to achieve power efficiency by distributing and executing queries. It also uses a tree routed to the base station for sending data back from the nodes. On its way up to the base station the data is aggregated according to the aggregation function.

To understand how structured protocols are designed for static network we choose one of the protocols, CTP (Collection Tree Protocol)[7] and do an in depth analysis of how the protocol works.

2.1.1 Collection Tree Protocol

The Collection Tree Protocol[7], is a best effort datagram communication to a base station. We dissect this protocol to understand how structured protocols are designed. It is

a tree based collection protocol designed for relatively low traffic. We define certain nodes as base stations and they advertise themselves. Rest of the nodes in the network from a path/tree rooted to the base station. Data can be sent to any of the base stations. A nodes selection of the next hop neighbour defines which base station is going to receive the node's data. Tree formation and tree maintenance are two main aspects of the protocol.

Expected Transmissions(ETX) is used as the routing gradient. Base stations or roots have ETX of 0. ETX values are real numbers with a precision on tenths. If the ETX of a node is 26, it is actually 2.6. The ETX of a node is defined as:

Node ETX = ETX of parent + ETX of its link to parent.

When nodes boot up all nodes have an ETX of infinite. Only roots have an ETX of zero. All nodes in the network periodically announce a control packet or beacon, with their ETX and node-ID. Nodes in the one hop range receive the beacon and update their ETX and neighbour table. If on hearing a beacon a node updates its ETX, it schedules its beacon in the near future. The ETX of nodes are represented on a scale of 10, i.e. a node can have ETX between 10-19, 20-29, etc. depending on its efficiency of forwarding data packets. Suppose a node X has ETX 20 and it forwards a packet without any retries then its ETX remains 20. If it forwards the packets in more than 32 retries it bumps its own ETX to 29. The calculations shown here do not exactly replicate the exact way of working but gives an understanding of how ETX values are updated.

Unstable routes or links are not desirable as they may cause routing loops or packets drops. They may be formed when a node leaves the network or when links fail. It is important to fix a broken link as soon as possible. Ideally the rate of fixing a broken link should be higher than the rate of link failure. A link can only be fixed when the node hears a beacon and updates its ETX and neighbour table. This means that for faster link fixing the beaconing rate has to be very fast. A smaller interval of becoming will help rectify flaky links very fast but a large interval will help save bandwidth in the network. So while deciding on the interval we have to make a trade-off.

The trade-off can be avoided to a certain extent by using *Adaptive Beaconing*. It consumes low bandwidth over a longer period on time and also helps in faster network recovery. Trickle[8] like beacon timings can be used for adaptive beaconing.

Trickle transmits the current version of the code. Along with this it uses suppression and adaptive timing. Suppression is the property by virtue of which a node suppresses transmissions if it hears the same version number as its own. Trickle also maintains a timer to maintain time intervals. It starts with a very low beaconing interval t_l and doubles it when the maintenance timer overflows till it reaches t_h, the highest beaconing interval. So if all nodes have same version of the code the beaconing rate stays at t_h. The beaconing interval is reset to t_l under the following conditions.

1. **When it receives a data packet from a node which has lower ETX**. This signifies that the node has got a data packet from it possible parent, which ideally should not have happened. The node lowers it beaconing rates to update its neighbours.

2. **Significantly reduced routing cost**. When it has a parent who announces its ETX as 1.5 times its own ETX, there is a possibility of finding a better parent. So the node again lowers it beaconing rate.

3. **A packet with P bit is received**. Generally a new node coming into the network advertises a P bit. In other words, the new nodes wants to listen from its candidate neighbours as soon as possible. When nodes listen to the pull bit they lower their beaconing interval.

Routing loops are a concern in tree bases protocols. It can cause multiple packet replicas in the network which may flow up the tree. These duplicate packets can quickly multiply with each hop. Some mechanisms used to avoid or rectify routing loops are:

1. Every control packet contains the gradient value of the transmitting node. If the gradient value is significantly lower than its own then it senses an inconsistency in the network and tries to resolve it. It broadcasts a beacon frame. The node listening to the beacon may adjust its ETX and gradient accordingly to fix the loop.

2. If ETX are higher than a particular constant, node just ignores them. This is a brutal way of managing loops but it works very well.

Next we shall see how datagrams generated at nodes are routed to the base station. If a node has multiple possible neighbours it selects the neighbour with the least ETX as its next hop parent. Routing and forwarding engine takes care of forwarding data packets to the next hop.

The routing engine has the following tasks:

1. To pick a time to transmit packets to next hop.
2. Retransmit packets to next hop when needed(No ACK situations).
3. If there are routing inconsistencies, initiate a fix.
4. Manage the transmit cache for local and forwarded packets.
5. Manage the receive cache and avoid duplicate packets.

The forwarding engine maintains a busy state when it is transmitting packets. When a transmission process is complete it looks in to the transmit cache for pending packets. In the process of transmitting a packet if it does not receive an ACK it follows a rigorous retransmission policy by trying to re-transmit up to 32 times before giving up and marking the packet as dropped. The size of the transmit and receive cache play a crucial role in determining the efficiency of a network.

Duplicate packets can be received at a node when the parent sends an ACK to the sender but the sender misses the ACK. Protocols implementing rigorous retransmission schemes pose more danger of duplication. When ACK is lost the sender re transmits the packet and the receiver has a duplicate packet to deal with. Packet duplication can be dangerous as it can aggregate over the network. On every hop the number of duplicated packets are multiplied by a factor of 2(approx.). So one duplicate packet multiplies to 2, then 4 and then 8 and 16 on subsequent hops.

To deal with this problem a cache of datagram signatures(Originating address, Seq No) is maintained. Whenever a node receives a new datagram it compares with all the signatures in the cache and drops the packet if it detects a duplicate. But sometimes, when routing loops are formed, duplicate suppression becomes a difficult task. In a loop originating address will

be consistent but seq. no will be incremental, exactly like in normal situation. To deal with this data frames can have a Time Has Lived(THL) field. It is incremented by 1 on each hop.

When a node receives a datagram it adds it to the receive cache. But before adding to the receive cache it checks weather the received packet is a unique packet or a duplicate. Duplicate packets are identified by comparing the packet instance, which can consist of the origin, seqno, collectid and THL of the received packet with all the packets in the receive cache. If duplicate packets are received they are not added to the cache and are dropped. This helps prevent duplicate packets from flooding the network and congesting it unnecessarily.

Similarly a transmission cache is used to store data to be transmitted. Transmissions are randomized to avoid collision. As only one packet is transmitted at a given instance, there can be more than one outstanding packets in the tx-cache. The size of the tx-cache is critical for maintaining high reliability. When a new datagram is generated by a node or when a node receives a datagram from its child to be forwarded, it checks for duplicates and puts it into the transmit cache. When the transmit timer overflows it picks up the top the cache and transmits it to its parent.

Nodes may have physical memory limitations. Choosing the cache sizes are a trade-off between available memory and reliability of the network.

2.2 Structureless Protocols

Relatively less work has been done for structureless protocols. Ideas based on random walk are seen as a viable method for designing such protocols. Structureless protocols reduce the overheads for maintaining tress. Such protocols are helpful in mobile scenarios where maintaining links are difficult. When nodes are mobile, even at small speeds, path failure and inconsistency in networks are expected to be significant.

Flooding and Gossiping[9] based protocols can be used for data accumulation. In gossiping based protocols nodes may forward data packets with certain probability. This helps in reducing the message overhead.

2.2.1 Flooding and Gossiping

Flooding data using structure free approach from all nodes to every other node has a messaging cost of $O(N^2)$. For calculating average consensus multiple rounds of local gossip can be used by a node to calculate the average of its neighbours. This can be repeated till convergence.

A diffusion like approach can be used as a variant for flooding and gossip. In this case the initiating node broadcasts a group of registers one for each of its neighbours. When a node gets a register it adds its own data if it has not been added. It then rebroadcasts the register if it knows about any new nodes. It continues till all nodes have copies of all the registers. Message size here is $O(N)$ and hence messaging cost is $\Omega(N^2)$. This technique assumes that node ids are known beforehand.

Flooding and Gossiping are not very efficient for aggregation related work. Recently, work has been done in WVU's wireless networks lab on push-pull based random walk protocols, with mobile networks as a motivation. We have used this protocol as a benchmark for evaluating against structured tree based protocols. Details about this protocol is discussed in chapter 4.

Chapter 3

Structured Protocol Design

In this chapter we discuss about the tree based protocol that we developed in WVUs's wireless networking lab. The protocol was developed for one-shot data aggregation. When the base station needs data, it initiates a flood. The nodes in the network use the flood to arrange themselves into a tree. The nodes transmit their data once they have a parent. Some features of the protocol are:

1. **Gradient-less ETX.** Child's ETX is 10 more than parent's ETX. Gradients do not exist.
2. **ACK Packets.** Packet level ACK for acknowledging datagram reception.
3. **Tx-spread.** Transmission spread for collision avoidance and latency control.
4. **No-Tree maintenance.** This protocol does not rigorously maintain trees. It uses periodic refreshing.
5. **Local Aggregation of data at nodes.** If multiple data packets are available in the tx-queue, the node aggregates all the packets into one and transmits them.
6. **Virtual Synchronization.** Virtually synchronize the entire network using the flood-ID. This is different from clock synchronization.
7. **Tx-Buffer and Rx-Buffer.** A transmission buffer which can handle larger number of packets and a receive buffer to check for duplicates.

8. **No Trickle Beaconing.** Beaconing is done at a constant rate. No trickle timers are involved.

Some aspects of the protocol which need more explanation are discussed in the following section.

3.1 Design Aspects

3.1.1 Beaconing

Creation of a tree is always initiated by one of the base stations. When the nodes boot up, all the nodes set their ETX and parent values to infinite. Only the base station sets its ETX to zero. The base station announces its ETX periodically to notify its one hop neighbours that it is a base station. Let us call this the control beacon. The beacon contains the node's ETX, nodeID and a beacon sequence number. The base station stops beaconing when it has at least one data packet from all the nodes in the network.

3.1.2 ETX and Parent Update

Nodes hearing the control beacon can update its ETX and parent if needed. They maintain a *lastHeardBeaconId*, which stores the *beaconID* of the last beacon the node heard. When it hears a beacon with an incoming $beaconID > lastHeardBeaconId$, it reacts to the control beacons the the following way:

- It updates its ETX if the beacon ETX is smaller than its own ETX by atleast 10.
- It also updates its parent and ETX if the incoming beaconID is greater than lastHeard-BeaconId.
- Updates its lastHeardBeaconId.
- Forwards the beacon ,i.e. it announces a control packet with its own ETX and *last-HeardBeaconId*.
- May schedule a data transfer with respect to the beacon-id.

3.1.3 Pushing Datagrams

Datagram originating from the node are pushed into transmitting buffer, called the tx-queue. The *pushData* thread keeps checking the queue for data to be transmitted. If a single packet is available in the queue, it simply transmits the packet. If multiple packets are available the nodes perform local aggregation. The aggreagation process is described later in this chapter.

The data packets are unicasts and are directed to the parent node. After transmitting data packets the node waits for the *ACKCheckTimer* to overflow. On overflow it checks if an ACK for the previous packet was received. In case an ACK was not received, the node retires up to 32 times before giving up and marking the packet as dropped.

It is important to notice that a dropped packet is not completely forgotten. The data packet is pushed back into the tx-queue. The node also sets its ETX to infinite and forgets its current parent. When the node hears a new control beacon, it updates its ETX and parent and tries to send the packet again. Dropped packets do not have any priority in the queue. When they are pushed back in the queue again, they will be processed on a FIFO basis. This process of pushing dropped data packets back to the queue continues till the packet is successfully delivered to the parent node.

Transmission Spread

This is a small randomness introduced while transmitting a datagram. A tx-spread(δ_t) of 3 seconds was selected for the protocol. The nodes select a random number in the range $[0, \delta_t]$ and adds it to the scheduled transmission time t. The data packets are transmitted at $t + \delta_t$ time. This helps in avoiding collisions, hence reducing number of retransmissions.

3.1.4 Local Aggregation

When nodes have multiple outstanding packets in the transmission queue, instead of transmitting them one after another, the nodes aggregate them in to a single datagram.

Let us consider n outstanding packets in the tx-queue, size of data packets as S_d bytes and size of data headers as S_h. If data packets were transmitted sequentially the total

amount of data transmitted would be $n * (S_d + S_h)$ bytes. Where as if we aggregate the data packets the amount of data transmitted would be $(n * S_d) + S_h$ bytes. The amount of bandwidth saved was $(n-1) * S_h$ bytes. Considering that headers contain significantly large amount of information, like originating address, parent's id, packet type etc, we save a lot of bandwidth.

Aggregation has its own trade-offs. The positives about aggregation being:

1. It reduces the number of data packet transmitted over the entire network.
2. Less bandwidth is used for routing similar amount of information at the base station. Most of the bandwidth is saved because of eliminated headers.

Some possible disadvantages of aggregation and how they can be handled are:

1. It may be difficult to trace the origin of the packet. This can be avoided by adding the origin field to the data packet.
2. Adding additional field to the data packet, increases the size of the datagram. But the entire bandwidth consumed is still less as the number of headers decrease significantly.
3. The biggest drawback is, when a datagram is lost, we lose data from multiple nodes.

3.1.5 Duplicate Packets

Data aggregation in a network, gives us an overall idea of the current state of the system. While performing mathematical operations like summation, average, count, etc it is essential to make sure that packets are not duplicated. In this protocol duplicate datagrams may be generated because of the following reasons

- Nodes retransmitting datagram due to lost ACK packets.
- Although nodes maintain a fixed size rx-buffer to check for duplicates with recently received packets, if duplicated packets are received at large time intervals the buffer may have been over written.

- The ETX or parent of a node or its parent has changed before the ACKCheck timer overflows.

If duplicate packets are not detected by intermediate nodes in the network, the base station processes all the incoming data and makes sure that it has only one copy form all the nodes. The base station needs data about the origin of the data packets to successfully eliminate duplicate packets.

3.2 Differences with CTP

The collection tree protocol can also be used for data aggregation. But it is not optimized for one shot aggregation. Some differences in the design of my protocol when compared to CTP are:

1. It uses periodic beaconing instead of Trickle.
2. Unlike CTP it does not use ETX gradients.
3. ACKs are packet level, whereas CTP uses link level ACKs.
4. Performs data aggregation at nodes, which CTP does not.
5. Nodes can by virtually synchronized up to a certain extend by using beacon-IDs.

3.3 Smart Refractory System

In this section we discuss about an application where the structured network protocol was implemented in real time. West Virginia University(WVU) in collaboration with Harbison Walker International Technology Center(HWI), is working on demonstrating the use high temperature sensor concepts for monitoring the health degradation of slagging gasifiers. The results of the research will help in development of smart refractories with features like embedded temperature and stress/strain sensors.

The refractory walls are made of high-chromia bricks. Current technology used thermocouples for measuring temperatures inside furnaces. Due to high temperatures of upto 1400

Celsius and corrosion limitations of sensors materials , the sensors degrade in a short time. Moreover these sensors are put into the furnace through access ports, which are prone to slag penetrations, hence affecting the performance of the sensors.

The idea of using smart brick with embedded sensors is being tested as a part of the research. Embedding the sensor into a brick will ensure maintaining the integrity of the sensor for a longer time under harsh operating conditions.

Interconnecting the sensors in the smart bricks to an external system, where the sensor signal will be processed by low power electronics and transmitted wirelessly to a centralized data processing center, can help in monitoring the health of the furnace. The data collected over the wireless network can be used for model based prediction of refractory health. An overview of such a system is shown in figure 3.1

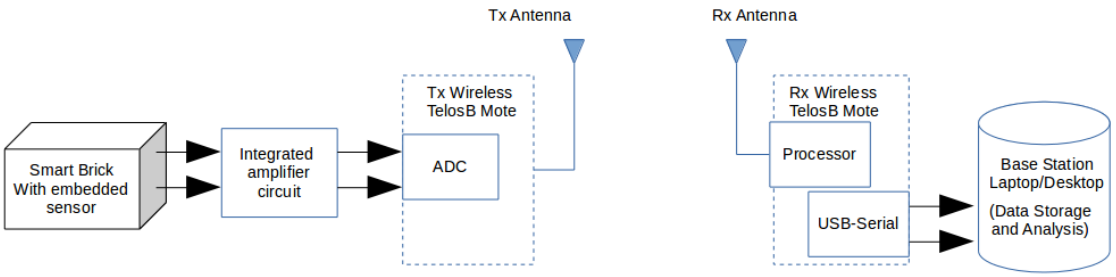

Figure 3.1: Wireless Network overview for data collection form smart bricks.

A tree-based protocol was used to collect data from the smart bricks. The performance and analysis of the implementation is discussed in chapter 5.

Chapter 4

Structureless Protocol Design

In this chapter, we discuss the design of data aggregation protocol that does not create or maintain a tree structure in the network. The protocol is based on biased random walks[4]. In this thesis, we have used this protocol as an example of unstructured protocol and used it for comparing against the tree based protocol described in Chapter 3. In this Chapter, we discuss the design and key ideas behind this protocol.

4.1 Self-repelling random walks

As described before, in static networks with stable links, data aggregation can be realized by traversing fixed routing structures such as trees or network backbones. However, in mobile networks and networks with frequent link changes, topology driven structures are likely to be unstable and to incur a high communication overhead for maintenance. Random walks are attractive alternative for MANETs because they are inherently stable in the presence of network dynamics, have no critical points of failure, avoid structure maintenance, and have very little state overhead. The resulting protocol is also quite simple. One or more tokens can be initiated in the network. A node that holds a token picks a random node in its neighborhood and transfers the token to that node. The first time that any token visits a node, it can be used by the node to add its state into the token for the aggregate being computed. This process is repeated until all nodes have been visited.

Unfortunately, the cover time for random walks (time to visit all nodes) is typically high

because of wasted exploration when a token repeatedly encounters already visited nodes. In order to expedite the cover time, an idea that we have previously explored is that of partially guiding random walks towards unvisited nodes [4] using a self-repelling strategy. In such a biased random walk, if there are one or more unvisited nodes in the direct one-hop neighborhood (i.e., within the communication range) of a token, the token is passed to one of the unvisited nodes chosen at random. If all the nodes are visited, the token is passed to a node that is visited least often (with ties broken randomly). It has been shown analytically that such a local bias self-repelling strategy, by itself, achieves significant speed up and a cover time of O(N.log(N)). It has also been shown that by just using local bias, a significant portion of the network can be covered without much wasted exploration at already visited nodes. However, when the fraction of already visited nodes in the network rises beyond a certain threshold, self-repelling random walks exhibit a slowdown. This is because when all the nodes within the communication range of a token holder are already visited, the scheme reduces to a canonical random walk until an unvisited node becomes a neighbor. While the order of convergence in relation to N remains O(Nlog(N)), the slowdown creates a long tail in the convergence and significantly increases cover time. To redress this shortcoming, a push-phase can be prepended before the random walk. We describe this in the next subsection.

4.2 Push-pull biased random walks

The key idea here is that each node advertises its own data to all its neighbors. This incurs an O(N) complexity. However, the push phase ensures that each node now carries information about all its one-hop neighbors. As a result, the random walk (pull phase) now does not have to traverse each node to complete data aggregation. By doing so, the expectation is that data aggregation using the random walk can finish before the slowdown starts (due to all neighbors already being visited). Thus even the pull phase can finish within O(N) time.

4.3 Protocol design

Each node fist advertises its data to its neighbors using a single hop broadcast. The pull phase (self-repelling random walk) then begins. For self-repelling random walks, a token holder announces that it has a token. Nodes that hear this message and have not been visited, start a timer to send a request at a random slot within a chosen interval [0, ..,Tr]. A timer Tr is started at the token holder to accept requests for the token. The token holder picks a random unvisited node if at least one unvisited node sends a request. Otherwise, the token holder picks the node that has been least visited. The token is transferred to the chosen node. The node that receives the token marks itself as visited if it was unvisited so far. If the token is used for data aggregation, an already visited node may not add its information again to a token. This concludes the procedure for token passing using self-repelling random walks. The token is continued to pass iteratively using this procedure. Once the token fails to visit new nodes for a threshold amount of time, the data aggregation is assumed to have terminated. The result is then flooded to all nodes which again incurs only O(N) message complexity. We have evaluated the above protocol in ns-3 and used it to compare against the tree based protocol.

Chapter 5

Results: Comparison of Structured and Unstructured Protocols

The discussions in this chapters are divided in to two parts. The first part focuses on the performance analysis and comparison of structured and unstructured protocols on networks with varying degrees of mobility. The second part shows the results of the implementation of static protocols for data collection in smart refractory environment.

5.1 Structured Versus Unstructured Protocol

Network Simulator 3(ns-3) was used to simulate the protocols. Ns-3 is a discrete event simulator. It is widely used by researchers working in wireless communication. It allows users to define network topology, use its in-build models(UDP, IPv4, point to point, etc.), define node and link configurations, execute discrete evens, log data and also supports graphical visualization.

Static Sensor Network Model

For the static sensor network, nodes were laid out in a rectangular grid at uniform distances. We considered two different network densities, i.e. number of nodes per given area. In the first one, the radio range was adjusted such that each node has exactly 4 neighbors on the grid. In the other, the radio range was adjusted such that each node has 8

neighbors on the grid. The network sizes used were 100,200,300,400,500,750 and 1000.

Mobility sensor network model

For the mobile network, the nodes are deployed uniformly over a square area. It has been shown that a node degree proportional to $log(N)$ is critical to ensure with high probability that the network remains connected. Hence, the radio range is adjusted such that in every unit communication range area (i.e. πR^2 where R is the communication range), the average number of nodes is $4log(N)$. A random walk mobility model is used. In this mobility model, at each interval a node picks a random direction uniformly in the range $[0, 2\pi]$ and moves with a constant speed that is randomly chosen in the range $[v_l, v_h]$. At the end of each interval, a new direction and speed are calculated. If the node hits a boundary, the direction is reversed. Motion of the nodes is independent of each other. An important characteristic of this mobility model is that it preserves the uniformity of node distribution: given that at time $t = 0$ the position and orientation of users are independent and uniform, they remain uniformly distributed for all times $t > 0$ provided the users move independently of each other. Average node speeds that were considered are from 3 m/s to 21 m/s in steps of 6. A speed of 3 m/s would mean that all nodes in the network have an average speed of 3 but the individual speeds of all the nodes are uniformly distributed between 2 m/s and 4 m/s. Both the protocols were run under static and mobility models at least 5 times with different random seeds. The average of the five runs was used as final data for analysis.

Nodes in both the models are simulated to have a link failure rate to 2% every 30 Seconds. Failing nodes recover within the next 5 seconds with a 95% chance. The exact recovery time is uniformly distributed over the 5 seconds.

Metrics

The following metrics were chosen to evaluate the two protocols.

- Aggregation time: In the case of the tree based protocol, this measures the time between the issuing of the aggregation request from the base station to the receipt of the data from the last node in the network. For the self-repelling random walk,

this is the time between the start of the push-phase and the time when the token has aggregated all the information.

- Messages: For the tree-based protocol, this is the sum of the flood packets and the data packets that are generated. Re-transmissions (when acks are not received)is included in this metric . Also, when packets are opportunistically aggregated at intermediate nodes, we count the individual node records that are transmitted. This is because, even though the messages are aggregated into one packet, the size of the message is proportional to the amount of data that is aggregated.

5.1.1 Results for Static Network

Total Packets Transmitted

For the structured tree, the number of packets is the sum of control packets and data packets. For the random walk total packets is the sum of token transfers and token requests. As the size of the network increases we expect the total number of packets to increase. Figure 5.1 shows the number of packets transmitted over the entire network. We see that the number of packets in a degree 8 network is less than the number of packets in a degree 4 network. That is because a node has more neighbours and lesser hops to the base station. The total packets transferred irrespective of the the degree is more for random walk as compared to the static tree.

Packets Per Node

Total packets in the network do not explain if the number of packets linearly increases with increasing number of nodes. To analyse this we consider the packets per node. Figure 5.2, shows the packets transmitted per node. A plot parallel to X axis would signify a linear increase in the number of packets. But we see that the trend is rather super linear. It can also be seen that for random walk load per node is $O(nlog(n)$. For the structured tree we expect the order be to $O(n^p), where 1 < p < 1.5$.

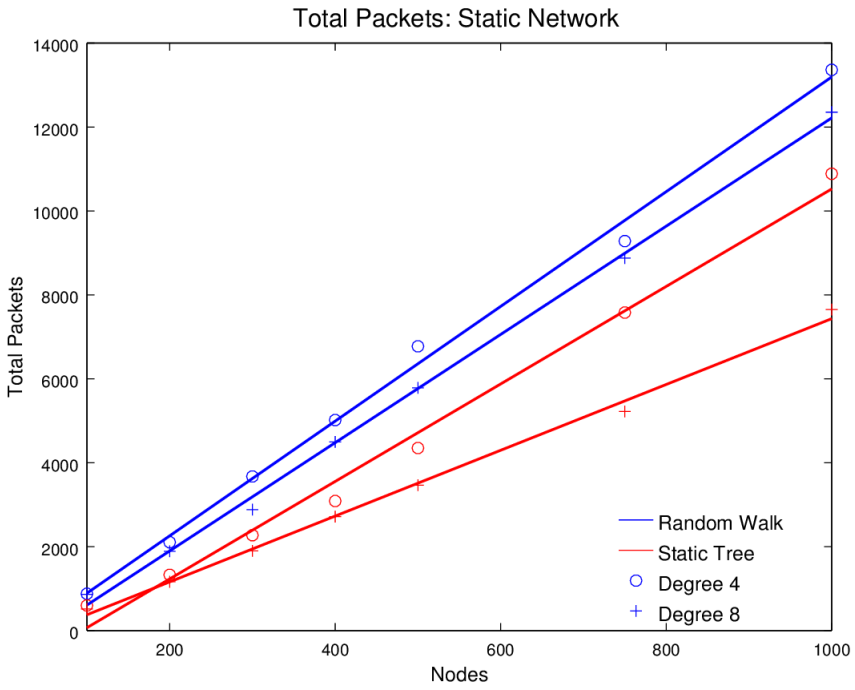

Figure 5.1: Total packets transmitted in a static network.

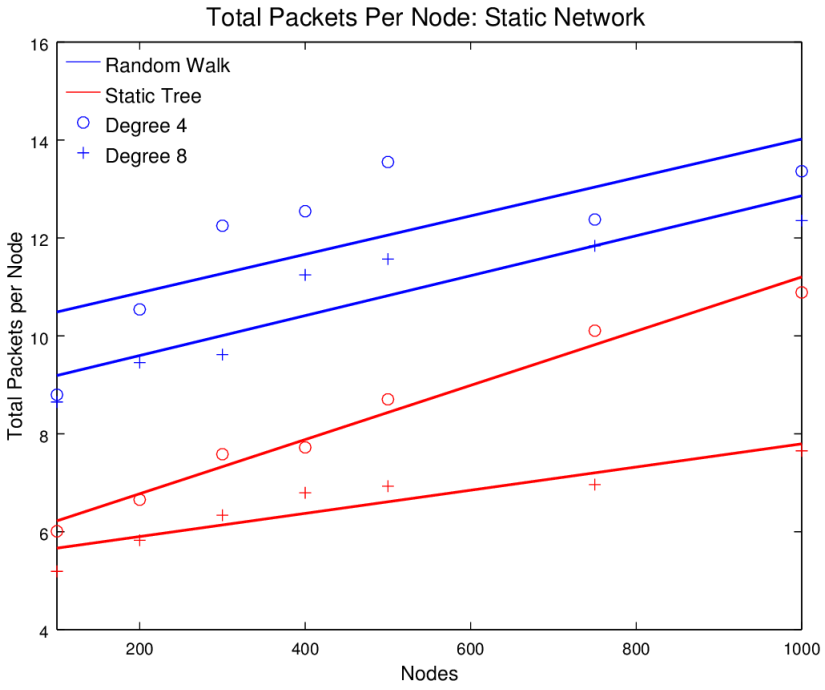

Figure 5.2: Packets per node in a static network.

Coverage Time

Coverage time for an aggregating network can be defined as the time taken to aggregate a single aggregation request from all the nodes. Figure 5.3 shows the coverage time for random walk is much higher as compared to tree based protocols. For 1000 nodes random walk takes more than twice the time.

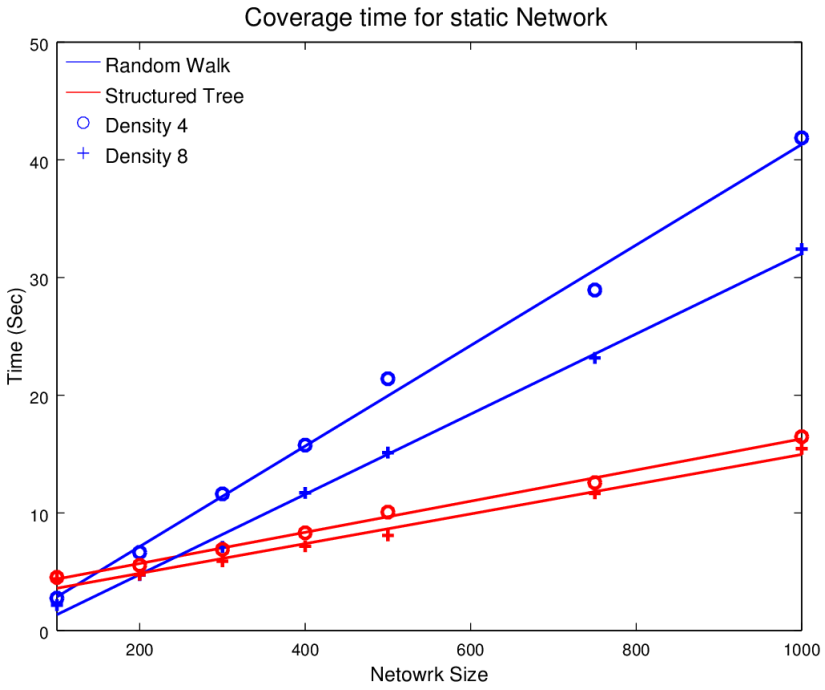

Figure 5.3: Coverage time for static network.

5.1.2 Results for Mobile Network

We discussed earlier that in mobile networks it becomes difficult to maintain tree like structures. This is due to moving nodes and radios which results in frequent node changes. The table below shows the link change rate per node per second in a network of density $4ln(n)$.

	3 m/s	9m/s	15 m/s	21 m/s
100	1	5	7	9
200	2	6	9	12
300	2	7	10	14
400	2	8	12	16
500	3	8	12	16
750	3	9	13	17
1000	3	9	14	18

Table 5.1: Link changes per node per second

Total Packets Transferred

We saw in static networks that total packets over the network was less for tree based protocol. But in a mobile network random walk performs very well. Although, for very small network sizes of less than 100 nodes structured protocol performs marginally better for all speeds but for higher network sizes random walk is a distant winner. Figure 5.4 shows the total packets transferred over the network.

Packets Per Node

We see in Figure 5.5 that the total packets transferred per node is almost linear for random walk, i.e. order is $O(n)$. This means even thought the network size and speed increases it has little impact on the load on each node. Structured protocol on the other hand has a slight advantage for very slow speeds and low density, but for the most part does not do really well. The increase of load per node is super-linear with increasing number of nodes. With increasing speed the packets per node also increases. We expect the order of packets per node to be $O(nlog(n))$.

Coverage Time

Figure 5.6 shows the coverage time for the network at different speeds and network size. With increase in speed it shows a very interesting trend. For a given network size with increasing speed the coverage time decreases significantly. This behaviour can be related to

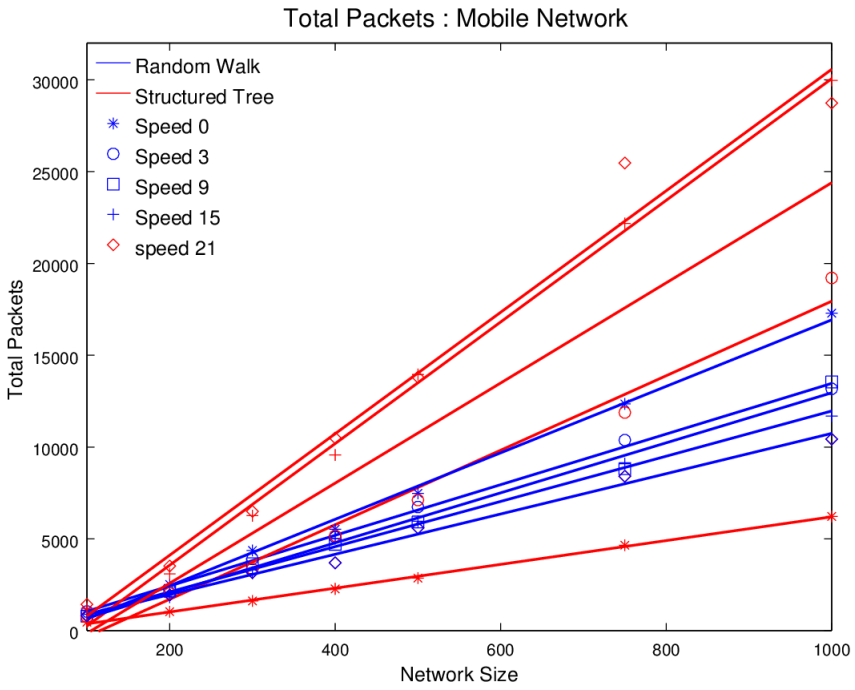

Figure 5.4: Total packets in mobile network.

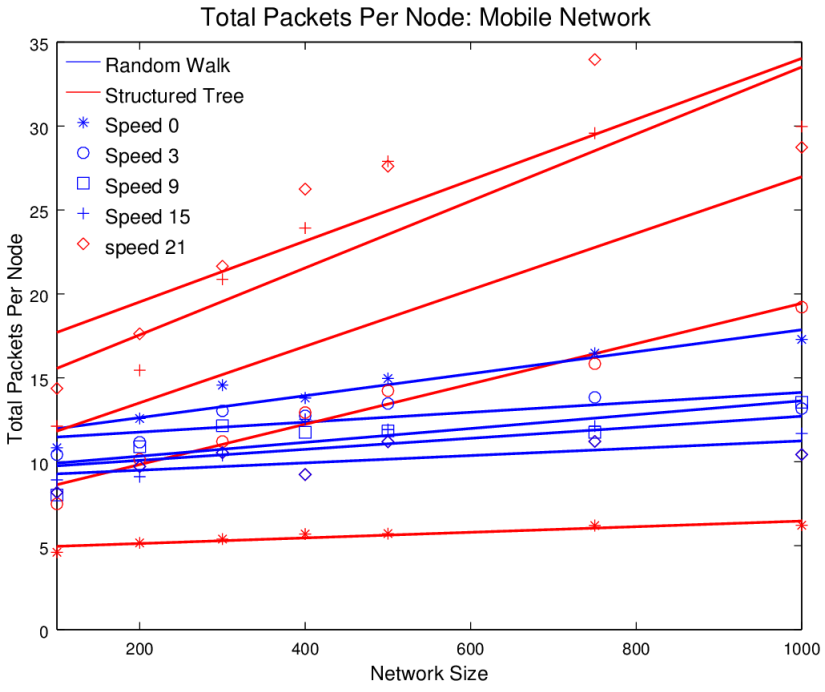

Figure 5.5: Packets per node in mobile network.

the fact that at higher speeds, randomness of more. Hence more nodes may come in to the one hop range of a given node. Figure 5.7 shows the coverage time for 750 nodes at various speeds. Structured protocol's coverage time is not very encouraging. With increase in speed its efficiency reduces drastically.

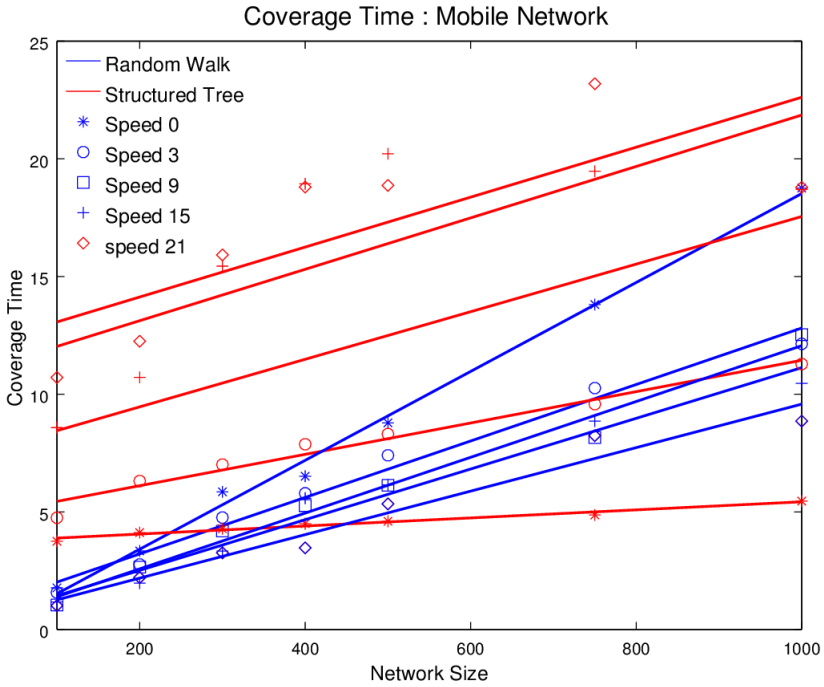

Figure 5.6: Coverage time in mobile netowrks.

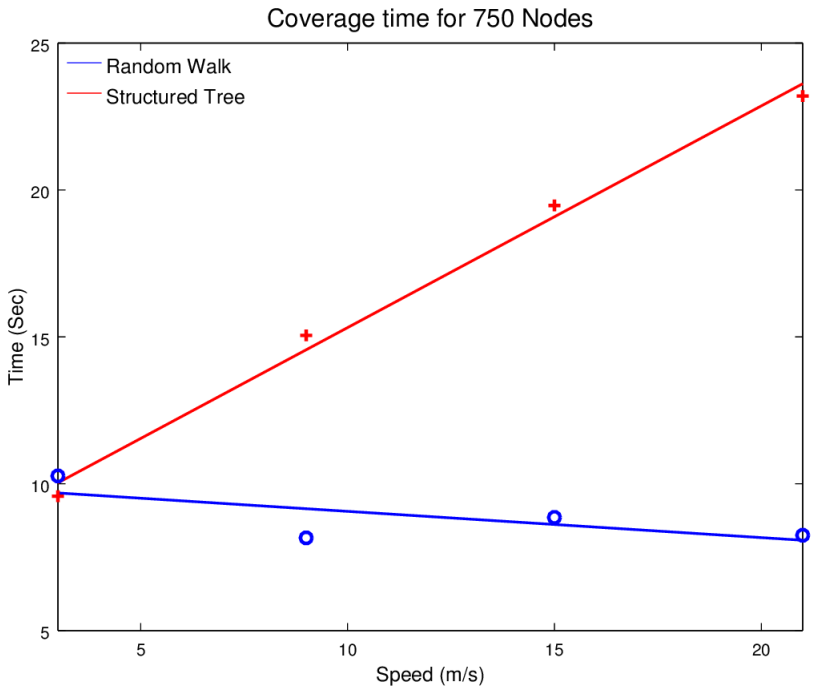

Figure 5.7: Coverage time for 750 nodes at different speeds.

5.2 Performance of Tree Based Protocol on Real Motes

The smart refractory system was used for performance evaluation on real motes. A team from WVU's Department of Mechanical and Aerospace Engineering has designed bricks with embedded thermistor based temperature sensors for use in refractory walls. Each brick has one sensor in it. It is connected to the ADC of a TelosB mote. An analog amplifier is used to amplify the output of the sensor to match the working range of the ADC. The motes were programmed using TinyOS.

A graphical user interface(GUI), developed in Python was used for easy visualization. It Plots the sensor data in real time and acts as a control unit at the base station. All the incoming data is logged into text files. Figure 5.8 shows how the GUI is organized.

The main challenge was to make sure that proper data was being transmitted over wireless. To verify this a self evaluation test was done. Instead of using the smart bricks, a dummy signal was generated. The signal was recorded both on LabView® and over the wireless network. The results of the self test can be seen in Figure 5.9. We see that the

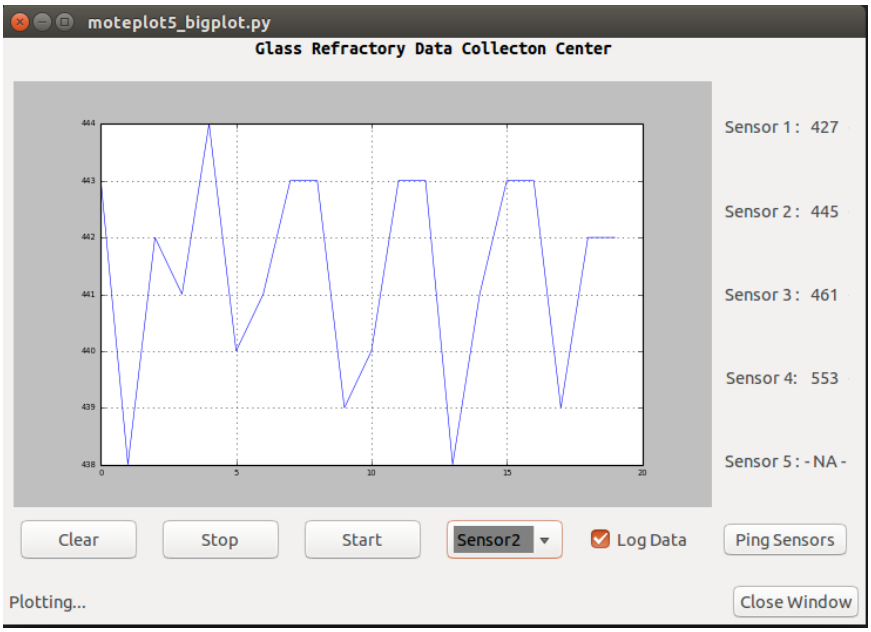

Figure 5.8: GUI for visualizing data and managing motes.

Wireless data is consistent with the LabView® data.

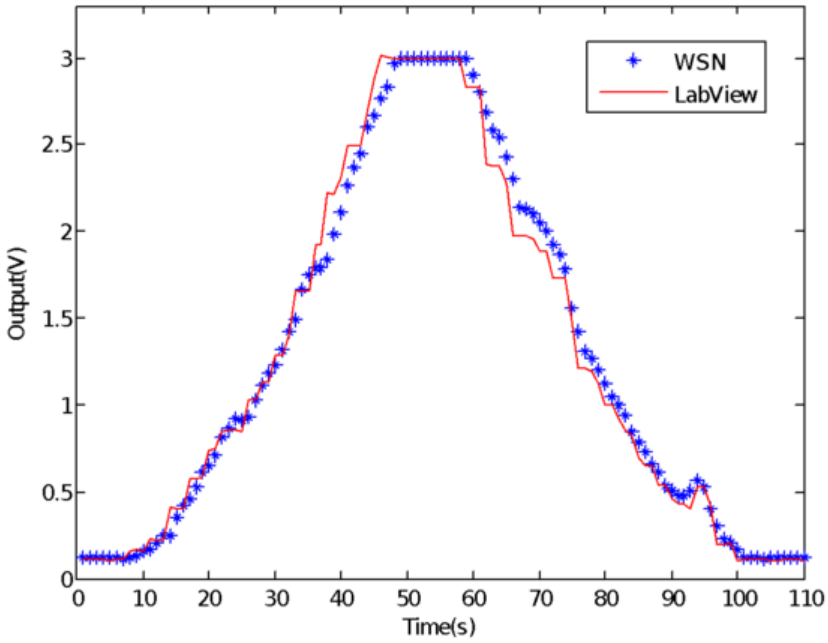

Figure 5.9: Self Test Results for WSN

Once the self test proved the reliability of wireless data collection, the nodes were connected to actual smart bricks to read sensor data. Figure 5.10 shows how the analog amplifier

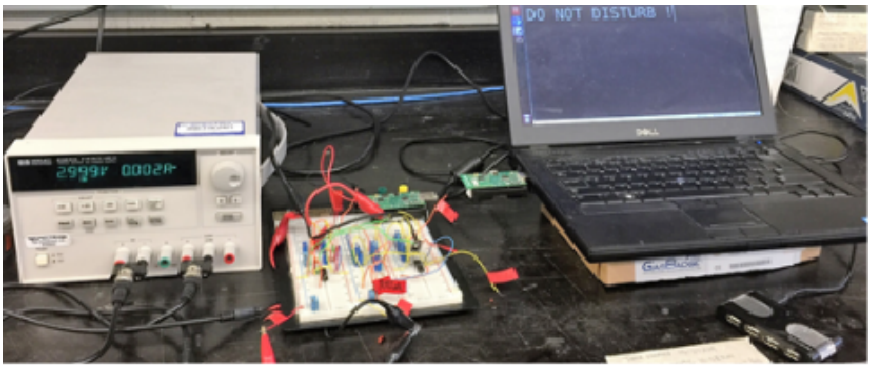

Figure 5.10: Wireless Mote and amplifier setup for data collection

and motes were connected in real-time.

Figure 5.11(a) shows the setup for high temperature furnace. The setup for the smart brick inside the furnace is shown in Figure 5.11(b).

(a) High temperature furncace

(b) Smart brick setup in furnace

Figure 5.11: Furnace and smart brick setup

The data was collected from the smart bricks. Again, to validate the data collected

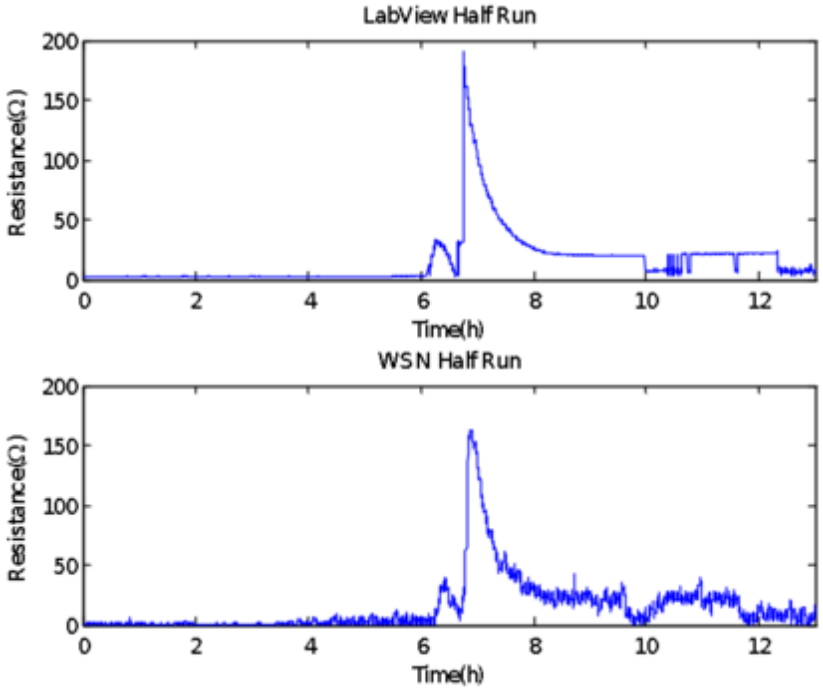

Figure 5.12: Comparison of smart brick data on wireless and Labview

over wireless, it was compared with the LabView® data. Figure 5.12 shows that data was accurately collected on wireless.

Chapter 6

Conclusion and Future Work

6.1 Summary

By comparing and analyzing the results we conclude that, in a static network, structured protocols perform better than random walks. They have lower number of packets transmitted, lower packets per node ratio and faster coverage time for bigger network size.

In mobile network, at very low speed and network size structured protocol performs reasonably well. With increasing speed and size its performance degrades rapidly. Scalability of structured protocols in mobile networks did now show encouraging results. Packets per nodes show a super linear trend with increasing network size and mobility. The coverage time also increases. Random Walk in mobile networks, maintained an almost linear packets per node ratio even with increasing mobility and size. Decreasing coverage time of random walk with increasing speed of the nodes reinstates its performance capabilities in mobile networks.

6.2 Future Work

We would like to evaluate performance of both the protocols at even higher scales of up to 5000 nodes and at higher speeds. This would help us understand the possibility of such protocols being used in real-time scenarios.

For the Smart Refractory System, we intend to test the system in a industrial refractory.

References

[1] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: A survey," *Wireless Commun.*, vol. 11, no. 6, pp. 6–28, Dec. 2004.

[2] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, New York, NY, USA, 2002, WSNA '02, pp. 88–97, ACM.

[3] Yinsheng Xu, Fengyuan Ren, Tao He, Chuang Lin, Canfeng Chen, and Sajal K. Das, "Real-time routing in wireless sensor networks: A potential field approach," *ACM Trans. Sen. Netw.*, vol. 9, no. 3, pp. 35:1–35:24, June 2013.

[4] "Census: Fast, scalable and robust data aggregation in manets," http://arxiv.org/abs/1409.7368.

[5] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 2–16, Feb. 2003.

[6] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 131–146, Dec. 2002.

[7] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, Maria Kazandjieva, David Moss, and Philip Levis, " CTP: An Efficient, Robust, and Reliable Collection Tree Protocol for Wireless Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, no. 3, 2013.

[8] Philip Levis, Neil Patel, Scott Shenker, and David Culler, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," Tech. Rep. UCB/CSD-03-1290, EECS Department, University of California, Berkeley, 2003.

[9] Zygmunt J. Haas, Joseph Y. Halpern, and Li Li, "Gossip-based ad hoc routing," *IEEE/ACM Trans. Netw.*, vol. 14, no. 3, pp. 479–491, June 2006.