A Real Deployment to Investigate PAMPA and Smart Gossip

Christopher Winstanley

Lancaster University

c.winstanley@lancs.ac.uk

Abstract

A broadcasting protocol for a wireless sensor network needs to be simple, lightweight and robust. The main goal of a broadcasting protocol is to gain the best propagation of a message for the lowest energy consumption. This is hard to achieve as typically sensor networks have low energy reserves and turbulent wireless transmission, making them extremely prone to failure. PAMPA and Smart Gossip are the two broadcast protocols compared in this paper, their performances in previous simulations are amongst the best in the current research area. This is determined by how successfully they deliver packets to each node on the network and more importantly how little energy they consume while doing this. It is also important to evaluate how they perform when they are subject to real network problems such as node, message and sending failure. Before either protocol can be recommended for deployment, it is important to find how well they perform when deployed on a real network. By comparing the two protocols on a real network we can find the most lightweight, efficient and robust protocol when used in real life. This paper discusses the benefits and drawbacks of the two protocols, presents an in-depth evaluation and establishes which is the most capable. The results of this paper conclude that PAMPA was by far the most effective broadcast protocol; it managed to deliver a high percentage of messages to each node on the network and managed to keep its message retransmissions and therefore its energy consumption very low.

Categories and Subject Descriptors C.2.1 [*Network architecture and design*]: Wireless communication

Keywords Wireless Sensor Networks, Broadcast Protocols, PAMPA, Smart Gossip

1. Introduction

A WSN (Wireless Sensor Network) is a self-configuring network of mobile devices connected by wireless links. It relies heavily on broadcast messages to implement higher-level services, as it has no centralised infrastructure.

A WSN needs to send broadcast messages to communicate data across the network. Each networked device can frequently change its link properties compared to the other devices on the network. A broadcast needs to be used so that a message can reach all nodes with one transmission from the sender. The propagation path of a message is defined as the path of nodes between the initial sender and receiver of that message. To broadcast a message across the network, each message must follow a particular propagation path.

Many academic papers have suggested protocols that try to find the most efficient and reliable propagation path on a WSN to broadcast a message. The problem that needs to be solved is how to get the most efficiency out of a broadcast protocol without sacrificing the reliability of it. The main goal of a broadcast protocol is to deliver a single copy of the message to each node, while minimising the number of transmissions.

When implemented on a network, a delayed flooding protocol will receive a message, make a record that the message has been received and then retransmits it. The protocol will not retransmit the message if it has been seen previously. In a lightweight network this has been proven to be a very wasteful solution to implement, as it will retransmit a new message every time. Networks are sometimes in long term deployment where the nodes have limited power they can harvest or store. This means that the inefficiencies of the delayed flooding protocol are not acceptable, as the nodes need to conserve battery power to run for a prolonged period of time. Another concern is that each node does not have a lot of available bandwidth and needs to restrict how much data it propagates.

Smart Gossip [1] and PAMPA (Power-Aware Message Propagation Algorithm) [2] are the two broadcast protocols compared in this paper. They have been chosen for comparison as they are two of the most prominent protocols in the field of energy efficient broadcasting.

PAMPA improves the efficiency of flooding proposals by removing some of the randomness associated with the retransmission decision on the nodes. The protocol uses the received signal strength of a message to estimate the distance to the messages' source, it then uses this information to decide whether or not to retransmit. PAMPA has shown in both a simulated [3] and a real-life environment [4] that it can provide message coverage close to that of a delayed flooding protocol without the large retransmission overhead.

The Smart Gossip protocol is one of PAMPA's largest competitors in terms of protocol efficiency and robustness. It has been proven in simulation to reduce redundant retransmissions greatly and be able to perform on an adverse network. Gossip protocols have been constantly used for comparison and improvement in the literature. Smart Gossip whether or not to retransmit a received message based on a dynamically assigned probability. It uses information about underlying network to decide on a probability of transmission thus making it dynamic and effective against node failures and the disparity of the network. In the simulations both protocols have been shown to be able to deliver a message to the majority of nodes on a network without having to retransmit redundant messages. They are efficient and will ultimately prolong the battery life of a node. The delayed flooding protocol is included in the experiments as a baseline to compare with the other two protocols.

With both PAMPA and Smart Gossip it is assumed there is no hardware support available (no GPS for relative positioning, no hardware implementation to efficiently broadcast to each node, no hardware support for determining the angle of arrival or the time of arrival). This means that the protocol running on each wireless sensor node will have to base its retransmission decision on the data it receives in each message.

This paper looks at how the protocols perform in a real heterogeneous network topology. This is where low performance of the protocol is more likely to occur, as the wireless nodes are not uniformly distributed and have to deal with real life environment variables that would not have affected the performance in a simulation. The protocols have to adapt to node failures properly, making sure that a broadcasted message gets to its destination even when its choice of propagation paths is limited.

The protocols are compared by implementing and running each of them on a low-power wireless sensor network, using TelosB motes. This paper comments on the results from this implementation and how each protocol performed on the network. To properly evaluate each protocol this paper discusses the protocols and an in-depth evaluation of the results obtained.

This report starts by looking into the background of broadcast protocols and the related work in Section 2. PAMPA and Smart Gossip are explained in Section 3. The experimental design is highlighted in Section 4. The implementation of each protocol is explained in detail in Section 5, followed by the challenges and issues encountered in Section 6. Section 7 analyses and discusses the results obtained. Section 8 contains concluding remarks and explains intended work for the future.

2. Background and Related Work

In this section, the main protocols that are related to and lead up to the development of PAMPA and Smart Gossip are explained. The research into broadcast protocols on wireless sensor networks is both extensive and deep-rooted. The search for a protocol that is both efficient and reliable is the main goal of the majority of WSN studies. Success in this area will allow for more application of small and low powered hardware in a larger range of environments. The recent emergence of popular wireless devices, such as the iPad and the Kindle, mean that these devices are no longer constrained to the research community. The mainstream application of wireless devices leads to a greater need for protocols that can perform sufficiently in a heterogeneous network. With constant node failures that make the topology dynamic and with no way of configuring a protocol at deployment time, there is a greater need for an efficient and reliable broadcast protocol that can cope with this distinct criterion.

2.1 Delayed Flooding

Most broadcasting protocols have their basis in flooding. Proposed methods in the literature mostly specify ways to optimise this basic functionality. Delayed flooding protocols retransmit a message if it has not been seen before. It will discard a message if it has been seen previously. This allows maximum use of the broadcast coverage, making it a very effective when propagating data. However as it retransmits every new message it receives, using delayed flooding creates a large number of redundant messages, all the messages it retransmits may have already reached the surrounding nodes thus making them unnecessary. This wastes the majority of its available bandwidth and consequently uses too much battery power in the process. The amount of traffic on the network from redundant retransmissions results in serious contention and collision on nodes, this is referred to as the broadcast storm problem [5].

2.2 Probability Based Protocols

Early gossip implementations such as GOSSIP(p) [6] & [7], use static gossip protocols. These protocols use a static probability to decide whether or not to retransmit a message. The probability is chosen when the network is deployed. This is unsuitable for wireless sensor networks, as the network topology is not always known before deployment. This means that any probability chosen would not be representative of the underlying network. If the probability is not set sufficiently high, then a node runs the risk of not being able to reach all nodes on the network with a broadcasted message. To compensate for this the probability of retransmission has to be set unnecessarily high, this could waste energy retransmitting when it is not needed, create too much network traffic or cause a broadcast storm. Denser regions of the network could achieve message dissemination with a very low probability. It is very hard to judge an appropriate probability for the whole network.

In adaptive Gossip Strategies such as GOSSIP(p,k) [6] where the retransmission probability is adapted on a per-originator or per-packet basis. The gossip probability is chosen in inverse proportion to the number of neighbours a node has. Trickle [8] adapts the retransmission probability on a per-packet basis by using the inverse proportion of the number of duplicate messages that have been overheard for each message it receives.

Both adaptive gossip strategies find it hard to cope with the heterogeneity of real network topologies. In a heterogeneous network there could be a large number of neighbour nodes leading to a crucial node. This crucial node could be a major link in a propagation path on the network. When using both protocols, the crucial node will not retransmit as it will set a very low retransmission probability. In Gossip(p,k), it will overhear from its neighbours and decide it does not need to retransmit all the time. In Trickle it will overhear a large amount of duplicate messages and decide not to retransmit. If the crucial node drops the message, this will lead to the rest of the network not receiving a large amount of messages.

2.3 Counter Based Protocols

In a counter based protocol [5], a node will wait while the new message it has just received is queued for retransmission. During this time it will count the number of retransmissions of that message it has received. The message is only retransmitted if the number of retransmissions is smaller than a pre-determined threshold. A lot of further WSN protocols are based on counting protocols, including PAMPA, as it turns out to concede little in the way of reliability for highly improved efficiency when compared to flooding. Again the counter threshold is chosen when the network is deployed leading to the same inefficiencies as the static probability based methods.

2.4 Distance Based Protocols

Distance based broadcast protocols [5] are related to counting protocols, as they will wait a bounded amount of time before deciding to retransmit. However the decision on whether to retransmit is related to the signal strength and not the amount of messages a node receives. The protocol will find whether the received signal strength of the messages received, after the original, pass a predefined threshold. If they exceed the threshold then the protocol will determine that there are enough nodes nearby and will decide not to retransmit. This method relies on the assumption that the signal strength received by a node is in relation to the distance of the sender. The distance is estimated using formula found in [9]. More recent work such as [10] uses RSSI to locate sensor nodes in a real ad-hoc environment.

2.5 Latest Research

The latest research in the area of counter and distance based protocols has been seen in work PAMPA variants [3]. Ellis et al suggest methods to extend the PAMPA protocol to improve its coverage and efficiency. Common parenting is one of the most effective variants proposed it is a way to determine whether two versions of the same message have come from the same direction and uses this information to determine whether it is worth retransmitting the message. The protocol will find if the parents (the node before a messages last hop) of two versions of a message are the same. If they are the same then this would decrease the likelihood of retransmitting the message. Other variants to PAMPA are proposed such as dynamic thresholding using directional look ahead or angular look ahead, which improve the coverage and efficiency of the original PAMPA protocol.

The most recent research into probabilistic protocols has been looking into coping with node failures. Sensor networks succumb to node failure as nodes often have low power supplies and are prone to errors. In striving to take further account of the underlying network, protocols such as RAPID [11] have been proposed to cope with node failures and varying network densities. RAPID uses corrective deterministic measures to ensure a constant delivery of messages, regardless of the underlying topology. Nodes that miss messages can request them from their neighbours, thus keeping the delivery ratio high. However this could have an adverse effect on the efficiency of the protocol. For a node to find out which messages their neighbours have seen, determine which messages they have missed and obtain them from the neighbour, it adds considerable overhead to the protocols performance.

3. Protocols

This section includes an explanation of how PAMPA and Smart Gossip deliver a broadcasted message to every node on the network and try to cut retransmissions to be more efficient.

3.1 PAMPA

The PAMPA protocol is represented in Figure 1. It achieves its efficiency by removing some of the randomness associated with the retransmission decision on nodes. Battery power is saved by preventing devices broadcasting where the message coverage gained would be unnecessary or minimal compared to the battery power used.

Each message broadcast on the network contains a unique identifier (uid) in its header. A uid is the combination of the message's source id and the ordered number of the message. A node uses this uid to tell whether it has seen a message before. If it has not seen the message previously, the protocol will flag that the message has been seen. If it has seen the message before, PAMPA will increase the number of retransmissions it has seen.

It uses the Received Signal Strength Indicator (RSSI) to set a time to wait before it retransmits a message. Within this time it counts the amount of retransmissions of that message it receives. If this is higher than the maximum threshold permitted then the node will drop the message. The maximum number of retransmissions that a PAMPA node will hear before dropping the packet is set to n. In the experiments conducted in this paper $n = 2$. PAMPA will wait the specified time based on the signal strength. If a node hears n retransmissions of the same message or more it will drop the message rather than retransmitting it.

The consequences of this protocols actions is that devices far away from the original source, broadcast themselves first, preventing devices closer to the source from broadcasting. This allows messages to propagate but also stops redundant low coverage messages.

3.2 Smart Gossip

Smart Gossip is a probabilistic protocol, it adapts its retransmission probability based on the underlying network topology. The protocol

```
/*Message is received by mobile device*/
onReceive(Msg)
{
        /*set retransmit to true*/
        send = true;

        /*Calculate timeout from RSSI*/
        timeout = TIMEOUT_CONSTANT * Msg.getRSSI();

        /*set pre-defined threshold*/
        threshold = 2;

        *Check if timeout has occurred*/
        while(timeout>0)
        {
                /*Check if same message has been received
                if(receivedSameMessage(Msg) )
                {
                        /*increment retransmissions*/
                        msgsRcvd++;


                        if(msgsRcvd > threshold)
                        {
                                /*Drop the message*/
                                send = false;
                                break;
                        }
                }

                timeout--;
                sleep();
        }

        /*If ok to send, retransmit*/
        if(send)
        {
                broadcast(Msg);
        }
}
```

Figure 1. Pseudo code that represents PAMPAs' actions on reception of a message.

is represented in Figure 3. The protocol quantifies the importance of each node within the network. The importance of a node increases when other nodes heavily depend on it to receive a message. If a node is depended on, it will retransmit with a proportionally higher probability.

Smart Gossip takes note of each message it receives so that it does not process a message with the same uid twice. The protocol finds the importance of a node 'X' within a network by constructing datasets to highlight the dependencies for that node. Each node will have a neighbour set which contains nodes that have a link with node X. Node X will have a parent dataset which contains nodes it depends on to receive messages from. It will have a child set which contains other nodes that depend on node X for messages. It also has a sibling set which contains nodes that are neighbours but they do not depend on node X or node X does not rely on them for messages. Figure 2 shows the hierarchy this constructs.

The goal is to identify these data sets correctly so that they can adjust the probability of retransmission appropriately. Nodes will extract information from overheard gossip messages and using a set of rules will determine which dataset a node falls into. If a node has more than one parent then it is possible for each of those parents to retransmit with a probability less than 1, as long as it is sufficiently high that at least one node retransmits. Over time as

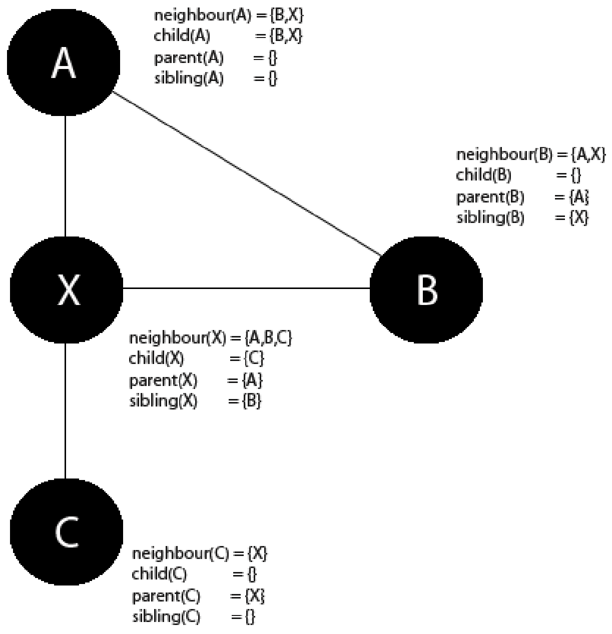

Figure 2. Dataset hierarchy that Smart Gossip will construct on the example topology. Node A will receive the message from the rest of the network and retransmit it to B and X. For C to receive the message B must decide to retransmit.

Smart Gossip learns about the underlying network it will refine the original retransmission probability of 1.

The application using Smart Gossip can also specify a required average reception percentage. For instance an application can specify that it wants each node to receive at least 90% of all messages sent out from the source. Even the furthest node from the source will need to achieve 90% delivery ratio therefore it is important to calculate the probability per-hop.

Probability per-hop is calculated where $Prequired$ is the probability defined by the children dataset and is used in:

$$Probability = (1 - Prequired)^{\delta} < (1 - Trel)$$

where $(Trel)^{\delta} = Tarp$. $Tarp$ is the specified average reception percentage for the whole network. δ represents the estimated diameter of network. From this probability Smart Gossip will determine whether or not to retransmit.

3.3 Smart Gossip Extensions

Previous studies [1] discuss protocol extensions to Smart Gossip to handle node failures. When nodes have failed later in the networks' lifetime, it is important to use higher gossip probabilities to meet the reliability requirements. If nodes are failing in the network, datasets used by a node have to be periodically updated to remove entries corresponding to failed nodes. If a node has not heard from one of its neighbours after a specified period of time, it will remove that failed node from its datasets. This action will increase the probability of retransmission. With this extension Smart Gossip should be able to adapt to failures within the network and construct new propagation paths with the dependency information gained.

```
Initialization at node i:
NeighborList = i;
SiblingList = i;
ParentList = null;
ChildList = null;
ParentAmountList = null;
TREL = 0.99; //probability required by application

/*Message is received by mobile device*/
onReceive(Msg)
{
    /*set retransmit to true*/
    send = true;

    /*obtain last-hop from message with amount of
     parents the node has*/
    parents = parent(j);

    AddToParentAmountList(parents);

    /*Find the lowest amount of parents from the nodes children*/
    lowestParentAmount = lowest(ParentAmountList);

    /*Calculate the probability of retransmitting*/
    probability = InversePower(1-lowestParentAmount, 1 - TREL);

    if(randomNumber > probability)
    {
        send = false;
    }

    /*If ok to send, retransmit*/
    if(send)
    {
        broadcast(Msg);
    }
}
```

Figure 3. Pseudo code that represents Smart Gossips' actions on reception of a message.

4. Experimental Design

Simulation results can be misleading. The simulations on PAMPA in papers by Miranda et al [2] and Ellis et al [3] both used ns-2 network simulator (v. 2.28 and v. 2.32 respectively) to simulate the broadcast of a message over a network. The Smart Gossip paper [1] used Qualnet(v3.7) to the same effect. A problem with network simulators is that they have poor failure models for message propagation [12]. In most cases they will drop a message at random. Failure criteria on a real network are a lot more complex. Message propagation suffers from such failures as: message convergence on a particular node, message error on a node, packet error from other transmissions on the network and packet error from obstructions in the wireless path. This is one of the main reasons to employ this set of experiments on a real network, to find out whether the protocols perform comparatively with their simulations. It would be unacceptable to recommend a protocol for real network use only knowing how it performs in a simulated environment.

4.1 Metrics

In each experiment several variables are measured to determine the performance of each protocol on the network. The variables obtained in each experiment are: the amount of messages broadcast, how many messages are received, how many messages each node retransmits and how many messages each node drops. These variables can be used to calculate the Retransmission Ratio and the Delivery Ratio.

Retransmission Ratio is the main criterion for WSN broadcast protocols. It represents the communication cost paid for each successful delivery by a broadcast. The highest possible retransmission ratio is 1 and corresponds to a run where all nodes that received the message retransmitted it. It is calculated by looking at the amount of broadcast deliveries and retransmissions on each node. The lower the retransmission ratio the more efficient a protocol is however if a protocol does not have a sufficient delivery ratio then it is ultimately useless. The retransmission ratio of a run is defined as the average ratio between broadcast deliveries and broadcast retransmissions:

$$Retransmission\ Ratio = \frac{1}{n_{\text{bcasts}}} \sum_{b=1}^{n_{\text{bcasts}}} \frac{\#\text{retransmission}_b}{\#\text{delivery}_b}$$

where n_{bcasts} is the number of broadcasts in the run (one per node), and $\frac{\#\text{retransmission}_b}{\#\text{delivery}_b}$ is the retransmission ratio of broadcast b, i.e. the proportion of nodes that retransmitted b.

Delivery Ratio is the average proportion of nodes reached by a broadcast on the network. The higher the delivery ratio is the more effective the protocol is and the larger the coverage of each broadcast. A delivery ratio of 1 means that every message that was broadcast across the network, has been received by every node. The delivery ratio can be gained from the results by calculating:

$$Delivery\ Ratio = \frac{1}{n_{\text{bcasts}}} \sum_{b=1}^{n_{\text{bcasts}}} \frac{\#\text{delivery}_b}{n_{\text{nodes}}}$$

where n_{bcasts} is the number of broadcasts in the run (one per node), and $\frac{\#\text{delivery}_b}{n_{\text{nodes}}}$ is the delivery ratio of broadcast b, i.e. the proportion of nodes that receive b.

The higher the delivery ratio the better the protocol was at propagating messages across the network, however its overall performance on a network is determined by the delivery ratio combined with the retransmission ratio

4.2 Experiment Reasoning

There are two sets of experiments conducted in this study. The Broadcast Node Switching set of experiments are used to find which protocol performs the best on a real network, how the location of the broadcast node affects the propagation of message and how obstructions in the networks' environment affects the performance of each protocol. The Node Failure set of experiments are conducted to find the most robust protocol that adapts best to nodes failing on the network.

4.3 Broadcast Node Switching

In the Broadcast Node Switching set of experiments the PAMPA and Smart Gossip protocols are compared. The delayed flooding protocol is used as a baseline for the comparison. One node in the network acts as a broadcast node, the other nodes implement a particular protocol by receiving messages from the broadcast node and retransmitting it if the protocols' logic concludes it should. A broadcast node periodically broadcasts messages over the network and does not receive any messages itself. The node that broadcasts on the network will be changed between experiments. This effectively changes the initial propagation path of a message and which direction it disseminates through the network. The different directions and paths a message can propagate through changes the weighting of traffic on the network.

Changing the broadcasting node between experiments gives an average of each protocol's performance on a heterogeneous network. This highlights the protocol with the lowest overhead and highest network coverage. It is interesting to see whether the protocols tested respond better to the broadcast node being in a clustered area of other nodes, mostly in the middle of the network, or whether they prefer the broadcasting node to be an outlying node on the edge of a network. This knowledge is important for future studies to be able to design better network topologies and find the best to way to implement the protocol on real networks.

This set of experiments is also an insight into how certain obstructions effect the retransmission of messages. The propagation of a message can be affected by many things such as: different thickness or material of obstructions in the environment, the volume of obstructions, other wireless signals or electrical wires. The testbed running the experiments, has a wide variety of obstructions for each protocol to contend with. It is interesting to see how the position of a node changes the amount and the type of obstructions, and how this effects the propagation of messages.

Hypothesis: In the Broadcast Node Switching set of experiments the delayed flooding protocol is expected to have a large overhead. As nodes always retransmit new messages it causes a lot of energy consumption. However despite its inefficiencies, it is expected have the best coverage on the network, because of the constant retransmissions each message always gets the maximum coverage it possibly can.

PAMPA and Smart Gossip are expected to be able to emulate the coverage of the delayed flooding protocol and have a far better efficiency as their retransmission logic should stop redundant messages being sent. PAMPA could do slightly better than Smart Gossip, its retransmission strategy based on signal strength is expected to be more adept at detecting and discarding redundant messages. This could potentially increase the protocols coverage as there will be less traffic on the network and it will be easier for messages to disseminate properly and reduce error rates on the network. PAMPA obtains better results when compared to the simulation results of Smart Gossip [1] & [3] but as the papers use different simulators and different methodologies this is of small weighting in the hypothesis.

4.4 Node Failure

In the Node Failure set of experiments, the two protocols are compared on a dynamic network where the topology is changed throughout each run. Delayed flooding is used as a baseline for the comparison again. A common cause of message disruption in WSNs is the failure of nodes. This affects the amount of propagation paths a message can be sent down and therefore the coverage of the message suffers. If a node fails then it would be harder for the network to propagate messages to the portion of the network the node serviced.

The main purpose of this set of experiments is to see how each protocol will cope when a node fails on the network and whether it will be able to maintain a high delivery ratio by rerouting messages efficiently. This is very important in WSNs as the battery powers of nodes often run low and the node can fail at any time. If a message is crucial to an application using the protocol, it still needs to get to every running node on the network regardless of other failures.

The Smart Gossip node failure extension termed Smart Gossip Advanced is included in this set of experiments. It is interesting to see if the periodic removal of failed nodes from a nodes' datasets improves on Smart Gossips' performance.

Hypothesis: In the Node Failure set of experiments delayed flooding is still expected to have very good network coverage but relatively poor efficiency. PAMPA is expected to perform better than Smart Gossip as it should be better at coping with node failure. Smart Gossip relies on constructing datasets of parent, child and neighbour nodes it has received messages from. If a node drops out, Smart Gossip will not adapt to the node failure and will still hold that node within its dataset, affecting the retransmission probability. PAMPA never creates a hierarchy and relies on the current message it has received to make a decision, it is a lot more dynamic and should react faster to the sudden absence of a propagation path.

Smart Gossip Advanced is expected to be comparable with PAMPA as the extension is designed with node failures in mind and should be able to adapt to adverse conditions. However it may take longer than PAMPA to react to node failures as it will take time for a node to omit a failed node from its datasets.

Figure 4. TelosB mote used to run each protocol during the experiments.

5. Implementation

The main task in implementing the protocols on a real network is to be able to compare them with each other fairly in a real environment. This section explains the hardware and software used to do this, how each experiment is conducted and how each protocol is implemented on the network.

5.1 Hardware and Software

The system is built, run and tested on TelosB motes (See Figure 4). TelosB motes are low-power wireless sensor modules designed for the research community. They have a 250kbps, high data rate radio and inbuilt antennae, they run on a 16-bit RISC processor with 10kB of RAM, they are powered by two AA batteries, and as their power consumption is rather low it gives them a long battery life. Their likely transmission range indoors is around 20m to 30m making them suitable to be used as a node in a WSN. Even though they are designed for the research community they will be very similar to anything used in a real network deployment.

Mobile hosts in the WSN used are assumed to share a single common signal channel with no link layer protocol to implement carrier sense multiple access (CSMA) or collision detection (CD). It is the responsibility of the protocol to try and avoid as many message collisions on the nodes as possible.

The Lorien operating system [13] runs on the TelosB nodes providing a lightweight control and interface to access some of the hardware's main functionality. The interface with the motes hardware properties is crucial when implementing some of PAMPAs and Smart Gossip's attributes.

Each experiment on the system is set up using a testbed web interface called TARWIS [14], (Testbed Architecture for Wireless Sensor Networks). TARWIS allows an Intel HEX file (i.hex) image of each protocol to be stored on the server and then experiments to be scheduled and set-up using the pre-defined images. An i.hex file is an ASCII text file with lines of text that follow the Intel HEX file format. Each line in an Intel HEX file contains one HEX record. These records are made up of hexadecimal numbers that represent machine language code or constant data. When an experiment has started, the testbed will remotely boot each node and then flash the selected image containing the 'i.hex file to the participating nodes. Each protocol will run as defined in the 'i.hex file which has been generated earlier.

TARWIS is used to reserve the network, schedule the experiments and store the results. TARWIS is provided by WISEBED [15] which is a joint effort of nine academic and research institutes across Europe to provide a multi-level infrastructure of integrated testbeds of large scale wireless sensor networks for research purposes. This is pursued using an interdisciplinary approach that integrates the aspects of hardware, software, protocols, and data.

5.2 Lorien

The Lorien operating system [13] is used to compile and run the code on the hardware used. Lorien is a dynamic component-oriented OS aimed at permitting component-based changes to it throughout every aspect of the system at runtime including its kernel. Its main intent is to support rich middleware. This is an open-source component programming model developed by the Computer Science department at Lancaster University, UK.

5.3 Wireless Sensor Network

It is important to maintain a stable environment throughout all runs of the experiment so the performance of each protocol could be obtained fairly. The protocols are tested in an office environment on the University of BERNs UBERN testbed. The main obstacles the nodes have to propagate messages through are doors and walls, potentially placing strain on the delivery of each message. The network is placed around the Computer Networks and Distributed Systems Department at BERN so there is also interference from other electrical devices. The network has 21 nodes placed randomly in different rooms around the office environment and sometimes different floors (see Figure 5 for a layout of the topology). Each node acts according to the chosen protocol on reception of a message and decides whether or not to retransmit, based on that protocol's rules.

5.4 Broadcast Node Switching

A total of seven experiments are executed for each protocol. After each experiment the broadcasting node is changed. Broadcast nodes are chosen that would be considered to be clustered in the middle of the network (nodes id: 1, 2, 3 and 14) and outliers on the network (nodes id: 18, 20 and 21). These nodes are chosen to be broadcast nodes as they give a good indication of whether a broadcasting nodes' position within a network affects the performance of the protocol implemented on it.

Each protocol experiment consisted of 20 runs so to get a suitable average for that protocols performance using that particular broadcast node. An experiment run is defined as the broadcast of 60 messages from the broadcast node, with one protocol being run on every node over a period of 18 minutes. From the experiments inception a message is broadcast every 10 seconds from the broadcasting node. This gives each message enough time to disseminate through the network before the next message is broadcast. Each run lasts for 18 minutes to give time for each node to set up correctly

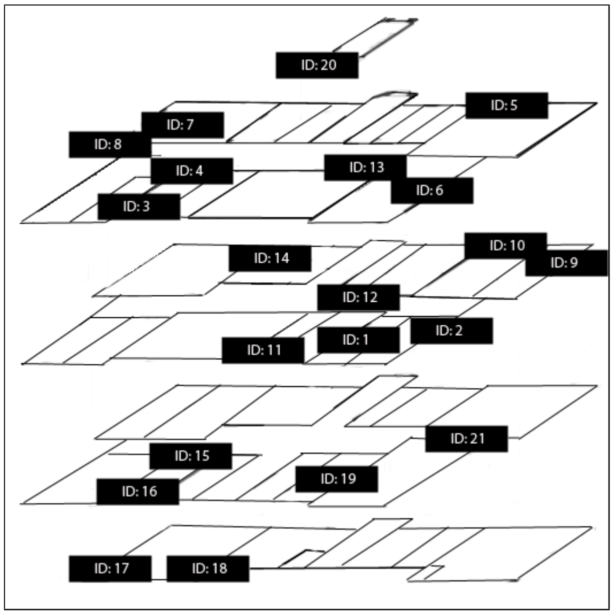

Figure 5. Map of the testbed used at University of BERN. Nodes placed over the four floors at random.

(around 7-9 minutes), give each of the 60 messages a chance to be broadcast and propagate through the network (takes around 6-8 minutes) and then a cooling off period (about 1-5 minutes) to make sure that there are no excess messages on the network when the next run starts .

There are some further runs in this set of experiments. The Smart Gossip protocol is tested with a slightly longer time between broadcasts. Instead of the usual 10 seconds between broadcasts, this is increased at first to 30 seconds for one run broadcasting from node 14 and then 60 seconds broadcasting from node 1. This is to test the dissemination of messages when using Smart Gossip. As there is no Carrier Sense Multiple Access with Collision Detection (CSMA-CD) in place on the network, messages may overlap. It is important to see whether the Smart Gossip protocol suffers from this as it dataset hierarchies may be corrupted.

5.5 Node Failure

A total of three experiments for each protocol are conducted. Three protocols in this set of experiments are tested: PAMPA, Smart Gossip and the Smart Gossip Advanced Extension, using the delayed flooding protocol as a baseline for comparison. Three randomly picked nodes (nodes id: 4, 10 and 15) are used as broadcasting nodes. They are spaced evenly around the network on different floors, this gives a fair average of how each protocol copes with node failure and not how they are responding to a particular broadcast node.

Each experiment consists of 20 runs using the same failure model for each. Nodes are dropped from the network using an exponential decay mechanism. Each node has a constant chance of failing per time unit. This corresponds to an adverse network environment, where all nodes are at risk of dying in parallel.

Exponential decay is calculated by using:

The number of surviving nodes at time t is:

$$N(t) = N(0)^{-\lambda t}$$

where λ is the 'decay constant', and corresponds to the probability for each node to fail per time unit. If the time of failure is larger than the duration of the experiment, the node will not fail during the experiment.

Each experiment has the same node failure model so it shows how well the protocol copes with node failure and not how badly the nodes failed on each particular experiment. The broadcast node never fails as this would completely cease any message propagation on the network. On a real deployment of the network the broadcasting node would constantly be changed, it could fail and messages would still be broadcast from other nodes.

5.6 Delayed Flooding Implementation

A basic delayed flooding protocol is implemented on the real network as it is a useful to see how good the performances of PAMPA and Smart Gossip are in comparison to this baseline. It is a simple protocol to implement and is a good stepping stone to use for implementing the other protocols over it. It gives an indication of how the network copes with large amounts of message traffic as it uses the full capacity of the network and is very inefficient. It also shows how each node copes with message convergence and the error rates on the network.

To implement the flag system used for the delayed flooding protocol, Smart Gossip and PAMPA, a lightweight byte array is used to store flags for each message. This keeps the need for data storage and capacity on each node to a minimum. Each message broadcast on the network contains a unique identifier (uid) in the header to differentiate each message, the uid is stored in the byte array to tell whether a message has been received before or not. To broadcast a message across the network a call to the underlying TelosB mote is made. This is called periodically using a timer defined when the node is instantiated and is changeable depending on the experiment being conducted.

5.7 PAMPA Implementation

PAMPA extensively uses the signal strength of a received message to determine whether or not to retransmit. To gain the RSSI from a message, a call to the underlying TelosB has to be made. This is done locally on each node by reading parameters from the incoming message, thus remaining context-free and is not reliant on meta-data about the network. A timer is then used to wait for retransmissions of the same message, if this exceeds the pre-defined threshold then the retransmit code is not called and the message is dropped.

Before the experiment is carried out the failure time of each node is calculated. An array is used to store the times at which each node should fail. A timer is called every second to increment a timeout variable, once the timeout variable reaches that nodes specific failure time, it will cease to receive or retransmit messages. This feature is only used for the Node Failure experiments.

5.8 Smart Gossip Implementation

This protocol implementation uses a flag system similar to the delayed flooding protocol. It uses this to determine whether it has seen a message before or not. The broadcast method is implemented in the same way as the delayed flooding protocol and PAMPA.

The datasets are stored on the nodes as linked list data structures. This makes them easy to traverse and update as necessary. Whenever the node receives a message from any node X with its parent field set to node Y, the rules to determine which dataset the nodes are placed are as follows:

1. Add X to Neighbour Dataset
2. If Y is not in Neighbour Dataset, add X to Parent Set

3. else if Y is in Parent Set, add X to Sibling Set
4. else if Y is in Sibling Set, add X to Child Set

To implement the Smart Gossip Advanced extension to Smart Gossip a timeout variable is applied to each of the nodes within the datasets and instantiated to 20. Every second a timer is called that decrements the timeout variable. If the timeout variable reaches 0, that particular node is removed from the dataset. If a message is received from a node within this timeout period then the timeout variable is reset to 20. This means that a dataset only has to wait a maximum of 20 seconds to get rid of a failed node from its datasets.

6. Challenges and Issues

In simulations a theoretical message failure model is never expected to produce exactly the same algorithmic response as real noise. However the results obtained are expected to be within a constant range of the simulated results and a protocol performing better in simulation should also perform better on the real network.

A reason why this may not be the case is the signal strength that PAMPA uses to calculate its retransmission probability can be unreliable. RSSI is not always a fair representation of the signal strength of a message. If the RSSI received is too high because of disruption, this could affect the signal strength of the message giving incorrect waiting times before a message is retransmitted. This could lead to more retransmissions of the message being received by the node; therefore the message is dropped when ordinarily it would be retransmitted. The opposite scenario, when the signal strength is received too low, produces a similar result.

As there is no link layer protocol in place, there is no carrier sense multiple access or collision detection. Therefore two nodes could try to use the same data channel simultaneously causing collision and corruption of the received messages on a node. This level of detail is not taken into account in the simulations and could therefore produce different results.

Another problem with a theoretical model is that the complexities of running experiments in real life are not taken into account. A simulation can be run at any time and gets the same performance results for that particular protocol every time. Running tests on a testbed around an office environment can cause certain anomalies to occur.

Nodes can fail even when not part of the node failing experiments. Nodes like any other hardware are susceptible to corruption and the breakdown of functionality. Every so often this means a node has to be repaired before being restored to the network. A full set of nodes is required to get the correct set of results needed, therefore any results obtained when a node had failed unexpectedly, the corresponding runs are executed again.

If a message is sent with such low signal strength then the data could be corrupt on reception and a node will try to interpret a corrupt message uid. This can then be propagated around the network. A retransmission of this corrupt message would never be heard and stopped by the protocol so the message would make its way to each node and skew the delivery ratio results. This also happens if a message is corrupted by other factors such as external broadcast signals.

A cyclic redundancy check (CRC) is placed in each packet header to prevent external broadcasts in the area being received and interpreted by each node. The CRC stops the messages that are valid on the network being corrupted by noise and then being interpreted. However, it does not stop noise corrupting the message entirely; this causes a reduction in delivery ratio when the message is dropped.

As the experiments take so long to set up and run individually they span over a large amount of time, this can lead to huge variants in the environment of the network. Factors such as people, other wireless activity, heating and other obstacles can all change dramatically during the period that the experiments are run in.

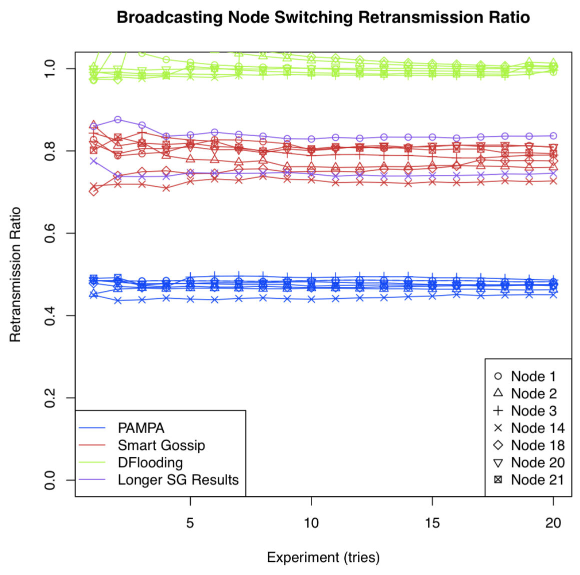

Figure 6. Average Retransmission Ratio results for PAMPA, Smart Gossip and the delayed flooding protocol with 20 experiment runs for each. Two more Smart Gossip experiments included with longer times between broadcasts. (One marker represents the average retransmission ratio from the first run to the current run).

7. Results

This section includes the results of the experiments on the real system and discusses the potential reasons behind them. The PAMPA and the delayed flooding protocol results are compared to that of the simulation results in Ellis et al [3], where the bridge used was set to 60m. The Smart Gossip results obtained are compared to the simulation results in Kyasanur et al [1] when the application reliability was 99%. The comparison points with both simulation results give a fair reflection of the obstructions faced in the implemented system.

7.1 Broadcast Node Switching

Figure 6 and Figure 7 show the cumulative average of delivery and retransmission ratios over 20 experiment runs. Over 20 runs the average is refined enough to give an appropriate representation of each protocols' performance on the network. The graphs also show the protocols where there was more erratic behaviour in message propagation than others. This can be seen by larger disturbances in the first few averages of the experiments. Some experiments with broadcasting nodes such as 18 and 21 are erratic throughout each of the runs therefore giving results that are harder to average.

This set of experiments showed that careful choice of a broadcasting node is important. For instance, in the runs where node 20 was the broadcasting node, the performance of each protocol suffers severely (this can be seen in Figure 7). Node 20 was a complete outlier to the network, it was positioned very high up in the office environment and struggled to get its initial broadcasts to the first set of nodes. Traces of this can be seen in Figure 12 where node 20's retransmissions are not received by any other nodes. It also shows how different volumes of obstructions can affect the performance of the protocol as the outliers that have to transmit their

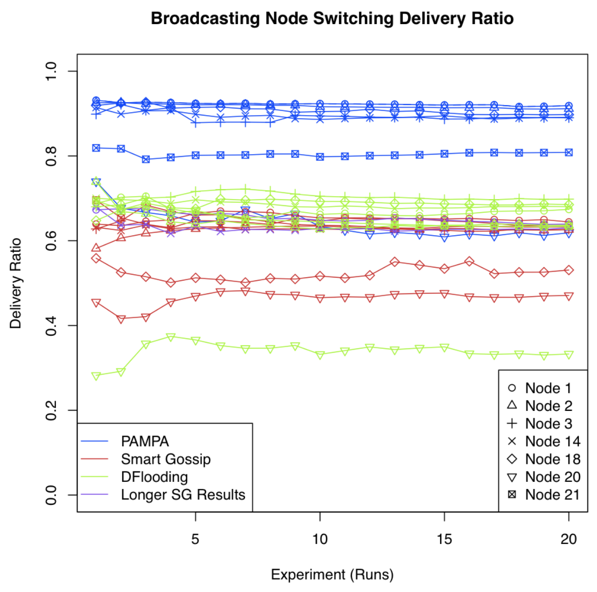

Figure 7. Average Delivery Ratio results for PAMPA, Smart Gossip and the delayed flooding protocol with 20 experiment runs for each. Two more Smart Gossip experiments included with longer times between broadcasts. (One marker represents the average delivery ratio from the first run to the current run).

initial broadcasts through more obstructions than those clustered in a single room. It seems all of the protocols like to broadcast in a cluster of nodes to obtain a higher delivery ratio. This can be seen in the results where clustered nodes such as node 1 and node 2 are used as broadcasting nodes, the performance for each protocol increases.

The delayed flooding protocol, used as a baseline comparison, had surprisingly interesting results. The protocol is expected to have the best delivery ratio by far however the amount of packets it retransmitted congested the network with a lot of unnecessary traffic. This caused a lot of packet errors and message convergence on nodes. This leads to its very poor delivery ratio and shows that a smarter protocol is definitely needed on this type of network, a message's propagation cannot just be brute forced.

The retransmission ratio results in Figure 6 show that PAMPA is more efficient than Smart Gossip. Wherever the broadcast node resides throughout the network, PAMPA is always the most efficient protocol to use. Smart Gossips' results are disappointing, its retransmission ratio is a slight improvement over the inefficiencies of delayed flooding but is largely wasteful when compared to PAMPA.

PAMPA unexpectedly obtained a greater coverage of the network than the delayed flooding protocol. Smart Gossip was also the worst performing protocol when it came to delivery ratio. A confirmation of the greater coverage of PAMPA can be seen in the difference delivery matrix (Figure 12), it shows that PAMPA reaches far more nodes with each broadcast than Smart Gossip and that it provides a much greater delivery to each of the main nodes within the network.

Sending messages in quick succession, creates a lot of network traffic and therefore creates favourable conditions for the occurrence of the broadcast storm phenomena. This creates a large number of collisions on each node in the network. While nodes in one hop are retransmitting message n, the sender is already trying to transmit message n+1. As there is no Link Layer protocol present, each node ignores the remaining traffic on the network whilst it is already processing a message.

Smart Gossip tries to create paths for retransmission, a trait that would be good if the propagation paths were stable. Even if nodes do not fail, the collisions on nodes make it look like nodes are failing. Consider two messages, m1 and m2 sent in fast succession. As the sender is always the same and nodes do not move, the path to be followed by m1 and m2 should be always the same. However, a node that is processing m1 does not receive m2. This can impact the protocols' performance because it would expect this node to retransmit both. As a result, propagation paths may be corrupted and the corresponding probabilities wrongly determined. This leads to a mass of redundant retransmissions. The information passed to the parent nodes from their children node is therefore not affecting the retransmission decision enough. The protocol needs to be far more brutal to lower its retransmission probability. This severely affects Smart Gossips' results.

In contrast this phenomena does not affect PAMPA, if the more distant node ignores m2, it is simply like it never existed (it could be said that PAMPA is stateless). The second more distant node will retransmit, this should hopefully ensure message propagation of m2 and could even deliver m2 to the node that was busy processing m1. This can be seen in the delivery matrix in Figure 10 where node 3 will miss out on all the initial message transmissions even though it is close enough to node 1, it will gain the messages from another node later on such as node 4 or node 6. As PAMPA is able to maintain its retransmission ratio at such a low level . This causes reduced traffic on the network allowing important messages to get through and not be corrupted by other transmissionsthis leads to an increased delivery ratio.

This paper investigates whether the proposed problem with Smart Gossip could be potentially solved by increasing the time between message broadcasts (this would not be feasible on a real time system using a WSN). Each message potentially has more time to disseminate through the network and potentially be dropped by each node. The results for the two experiments conducted first with 30 seconds between broadcasts and then 60 seconds produced similar results to the original results with just 10 seconds. It seems that no matter how much time given to Smart Gossip it is unable to drop packets as thoroughly as needed to reduce the network traffic before the next broadcast is sent. The nodes keep retransmitting the message until a new message is broadcasted. This message is not propagated fully because of all the increased traffic still on the network.

The Broadcasting Node Results show that the simulations have no real bearing on the performance of a protocol on a real network. In the simulations [3] PAMPA presented with around a 50% retransmission ratio, the results obtained from this set of experiments show similar results of 48%. However PAMPA's delivery ratio in contrast delivers each message 91% of the time to each node on the network. This was a vast improvement on the simulation results which found it to have a 76% delivery ratio. Smart Gossip's retransmission ratio was around 80%. Compared to the simulation results [1] where it obtained an overhead of 68%, this is relatively high. The 63% delivery ratio shows that Smart Gossip performs much worse on a real network, in the simulations it achieved a delivery ratio of 95%. As already mentioned the simulations have poor message failure models and do not represent the complex failures experienced in a real network.

7.2 Node Failure

Figure 8 and Figure 9 shows the average of delivery and retransmission ratios per minute over the 20 experiment runs. Over the 20 runs the average is refined enough to give an appropriate representation of each of the protocols performance on the network per minute. As

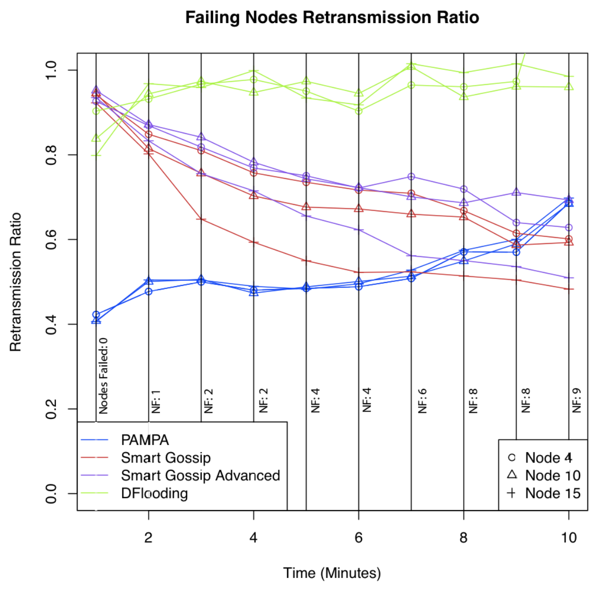

Figure 8. Average Retransmission Ratio results in the Node Failure Experiments for PAMPA, Smart Gossip and the delayed flooding protocol with 20 experiment runs for each. (One marker represents the average retransmission ratio from that minute in each of experiment runs).

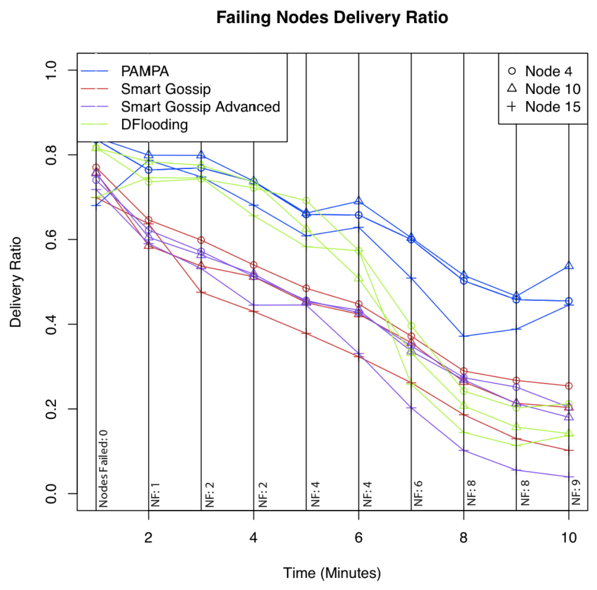

Figure 9. Average Delivery Ratio results in the Node Failure experiments for PAMPA, Smart Gossip and the delayed flooding protocol with 20 experiment runs for each. (One marker represents the average delivery ratio from that minute in each of experiment runs).

nodes fail on the network (shown on Figure 8 and Figure 9 by the vertical lines) the graphs display how each of the protocols adapt to the failure of nodes and attempt to keep propagating messages.

PAMPA obtained the best results by far in this set of experiments. PAMPA's delivery ratio, stayed higher than the delayed flooding protocols' and Smart Gossips' throughout the experiment and for all three broadcasting nodes. There were a few exceptions to this where the delayed flooding protocol coverage was marginally higher but after around the fifth minute when more nodes had started to fail it is clear that PAMPA results are far better.

The delivery ratio will decreases as there are less propagation paths for messages to use to reach outlying nodes. Unlike the other protocols, PAMPA shows signs of adapting to the new propagation paths in the network. After every failure of a node the delivery ratio can be seen to improve slightly, however it does not get chance to improve greatly before another node drops out.

A further sign that PAMPA adapts to new propagation paths in the network is the retransmission ratio results. It can be seen that as nodes drop out, PAMPA slightly increases its retransmission probability to try and cope with the failure. The protocol gets slightly less efficient after every node failure, however it keeps the delivery ratio much higher than the other protocols.

Smart Gossip does not respond to the loss of propagation paths and blindly tries to retransmit packets without taking account of the current underlying network, this can be seen when more nodes drop out of the network the retransmission ratio decreases constantly. The protocol shows no intention of compensating for the loss of nodes and reacting to the new propagation paths on the network by increasing its retransmission ratio.

In addition to other reasons mentioned previously, Smart Gossips poor performance in this set of experiments is due to the rigid structure of the protocol. The protocol puts each node into datasets, this causes a very structured path for message dissemination. When a node fails it is not removed from the datasets and the probability is not adjusted accordingly, leading to redundant retransmissions.

PAMPA has been shown to adapt very well to a dynamic network and increasingly adverse conditions. PAMPA is a lot more dynamic because of its looser structure and lower data overhead. As PAMPA never creates a hierarchy of nodes and relies on the current message it has received to make a decision, it reacts quicker to the loss of former propagation paths.

The Smart Gossip Advanced extension to Smart Gossip where it refreshes its datasets periodically, was expected to vastly increase its performance in this experiment as it made the protocol more dynamic. However the results of Smart Gossip Advanced were more or less the same as the original protocol. The data structures are still too rigid to change in such a short amount of time. It takes around three message propagation before the protocol can change its probability, thus making it very hard to find new propagation path in a network that is constantly changing. In a rapidly decaying network the retransmission probabilities are not changed enough based on the new information obtained.

8. Conclusion

The main aim of this paper was to investigate whether PAMPA or Smart Gossip was the best solution to the efficient broadcast problem. The results conclusively show that PAMPA is the best performing protocol on a real system. The way it bases its retransmission decisions on the current signal strength of the message keeps it lightweight and dynamic, allowing it to adapt to new and better propagation paths within the network.

Smart Gossips' performance on the network was relatively poor as it retransmitted messages far too much. Smart Gossip tries to construct hierarchies of each node on the network to disseminate messages effectively. There is no link layer protocol in place on the network therefore messages can be completely ignored by nodes whilst they are processing other messages. This corrupts the propagation paths of messages meaning Smart Gossip cannot construct an effective hierarchy of nodes. This leads to incorrect retransmission probabilities and poor efficiency. PAMPA does not suffer from this as it is effectively stateless and bases its retransmission decision on each packet it receives.

This paper has shown that when a protocol drops redundant messages this will increase the delivery ratio. If a protocol can keep its retransmissions low, it causes less traffic on the network. A protocol can then propagate messages with less hindrance from collision on nodes or error in transmission, this leads to a far greater coverage of the network. On a small network, like the one implemented in this paper, the increased retransmissions from a ‘dumb’ protocol creates too much congestion on the network and leads to an overall poor delivery ratio. For research in the future this paper has shown that it is important to choose the position of broadcasting nodes within the network carefully. Taking care to place important nodes in clusters of other nodes will result in better coverage for each of the broadcasting nodes’ transmissions.

PAMPA coped more efficiently with nodes failing on the dynamic network, this is because of its stateless approach. When Smart Gossip constructs hierarchies of the surrounding nodes it does not remove failed nodes from its datasets. The failed nodes will still affect the retransmission decision leading to redundant retransmissions. Even with extensions to Smart Gossip to periodically remove failed nodes from datasets, it was still to slow to react to a rapidly decaying network.

The simulation results showed little bearing on the results in this paper apart from PAMPA performing better than Smart Gossip. A further point to be derived from the results is that it is very important to test protocols on a real network before they are recommended for real world deployment.

8.1 Future Work

In this paper PAMPA showed promising signs of recovering from node failure. It was not given long enough to recover properly before another node failed and changed the propagation paths once more. It would be interesting to see if given a longer period of time it could fully recover and obtain its previous delivery ratio.

A comparison between PAMPA and RAPID [11] on a real life network would be interesting to study. RAPID has been shown in simulations to cope well with network failures and varying network densities. Its results were also vastly improved upon Smart Gossip in dealing with node failures. It would be a good comparison to see how well PAMPA performs against it.

Xi et al, [10] have developed a wireless network and protocol in a forest covered environment. It would be very interesting to change the environment in which the network used in this paper resides and see if and how the protocols can cope with this change. The real network in Bern was implemented around an office environment, a difference in obstacles to overcome would be an interesting test of the hardware and the protocols’ capabilities. It could also show how the protocols would perform in real deployments for geographical studies that are often in natural environments.

References

[1] P. Kyasanur, R. Choudhury, and I. Gupta, “Smart gossip: An adaptive gossip-based broadcasting service for sensor networks,” *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, vol. 0, pp. 91–100, 2006.

[2] H. Miranda, S. Leggio, L. Rodrigues, and K. Raatikainen, “A power-aware broadcasting algorithm,” in *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium on*, pp. 1 –5, 2006.

[3] C. Ellis, H. Miranda, and F. Taïani, “Count on me: lightweight ad-hoc broadcasting in heterogeneous topologies,” in *Proceedings of the International Workshop on Middleware for Pervasive Mobile and Embedded Computing*, M-PAC ’09, (New York, NY, USA), pp. 1:1–1:6, ACM, 2009.

[4] C. Winstanley, “A real life evaluation of lightweight ad-hoc broadcasting topologies.” Paper looking into PAMPA and its variants on Real Networks, 2010.

[5] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, “The broadcast storm problem in a mobile ad hoc network,” in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom ’99, (New York, NY, USA), pp. 151–162, ACM, 1999.

[6] Z. J. Haas, J. Y. Halpern, and L. Li, “Gossip-based ad hoc routing,” *IEEE/ACM Trans. Netw.*, vol. 14, pp. 479–491, June 2006.

[7] M. J. Miller, C. Sengul, and I. Gupta, “Exploring the energy-latency trade-off for broadcasts in energy-saving sensor networks,” *Distributed Computing Systems, International Conference on*, vol. 0, pp. 17–26, 2005.

[8] P. Levis, N. Patel, D. Culler, and S. Shenker, “Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks,” in *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, (Berkeley, CA, USA), pp. 2–2, USENIX Association, 2004.

[9] E. K. Wesel and E. Khayata, *Wireless Multimedia Communications: Networking Video, Voice and Data*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed., 1997.

[10] W. Xi, Y. He, Y. Liu, J. Zhao, L. Mo, Z. Yang, J. Wang, and X. Li, “Locating sensors in the wild: pursuit of ranging quality,” in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys ’10, (New York, NY, USA), pp. 295–308, ACM, 2010.

[11] V. Drabkin, R. Friedman, G. Kliot, and M. Segal, “Rapid: Reliable probabilistic dissemination in wireless ad-hoc networks,” *Reliable Distributed Systems, IEEE Symposium on*, vol. 0, pp. 13–22, 2007.

[12] Y. Bresler and A. Macovski, “Exact maximum likelihood parameter estimation of superimposed exponential signals in noise,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 34, pp. 1081 – 1089, Oct. 1986.

[13] B. Porter and G. Coulson, “Lorien: a pure dynamic component-based operating system for wireless sensor networks,” in *Proceedings of the 4th International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks*, MidSens ’09, (New York, NY, USA), pp. 7–12, ACM, 2009.

[14] P. Hurni, G. Wagenknecht, M. Anwander, and T. Brau, “Draft paper on a testbed management architecture for wireless sensor network testbeds (tarwis).” http://rvs.unibe.ch/research/pub$_{f}iles/HWA10.pdf$, 032011.

[15] I. Chatzigiannakis, S. Fischer, C. Koninis, G. Mylonas, and D. Pfisterer, *WISEBED: An Open Large-Scale Wireless Sensor Network Testbed*, vol. 29 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 68–87. Springer; 1st Edition. edition (2 Mar 2010), 2010.

A. Delivery Matrices

Figure 10 and Figure 11 shows the delivery ratio between each of the links within the network. They show each nodes reception of messages from all of the other nodes in the network. Figure 12 shows the difference between the delivery ratios on each node for PAMPA and Smart Gossip. Through this we can see how PAMPA and Smart Gossip compared at establishing links around the network and where their propagation paths may have got stuck, or where a node was unreachable by any protocol. If the delivery ratio in a particular cell is 0 it means there was no link present between the two nodes. There are many occasions where messages are sent but not received with Smart Gossip, while they are normally received with PAMPA. A value of zero would indicate a link that does not exist (out of range) the much results show there is an obvious problem of link quality with Smart Gossip, probably due to collisions on crucial nodes.

PAMPA Node 1 Delivery Matrix

SEND

RECV

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0.45	0	1	0	0.02	0	0.64	0.67	0.44	0.33	0	0	0.03	0	0	0.2	0	0
4	0.78	0	0	0	0	0.13	0	0.02	0	0	0.17	0.13	0.07	0	0	0	0	0	0.27	0	0
5	0.95	0	0	0	0	0.13	0	0	0	0.09	0.06	0	0	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0.32	0.81	0	0	0	0.15	0	0	0	0	0.73	0.3	0	0	0	0	0	0	0
8	0	0	0.32	0.74	0.12	0.13	0	0	0.05	0.55	0	0	0.6	0.26	0	0	0	0	0	0	0
9	0.85	0.04	0	0	0	0.13	0	0	0	0.18	0.22	0	0	0	0	0	0	0	0.07	0	0
10	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0.98	0	0	0.03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0.81	0.04	0	0.03	0	0.25	0	0	0	0.18	0.17	0	0	0	0	0	0	0	0.07	0	0
13	0.69	0	0	0.06	0	0.38	0	0	0	0.27	0.28	0.06	0	0	0	0	0	0	0.27	0	0
14	0	0	0.13	0.52	0	0	0	0.12	0	0	0.5	0.56	0.47	0	0.04	0.24	0	0	0	0	0
15	0	0	0.1	0.68	0	0	0	0	0	0	0.33	0.75	0	0.13	0	0.24	0	0	0.4	0	0
16	0	0.08	0	0.45	0	0	0	0	0	0	0.67	0.63	0	0	0.02	0	0	0	0.73	0	0
17	0	0	0	0	0	0	0.56	0.73	0	0	0	0	0	0	0	0	0	0.16	0	0	0
18	0	0	0	0	0	0	0.85	0	0	0	0	0	0	0.87	0	0	0.18	0	0	0	0
19	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0.9	0	0	0	0	0	0	0	0	0	0.8	0	0	0	0	0	0	0	0
21	0.88	0	0	0	0	0.13	0	0	0	0.09	0.17	0.06	0	0	0	0	0	0	0.07	0	0

Figure 10. Delivery Ratio Matrix for PAMPA using node 1 as a broadcasting node. (Sending nodes are along the x axis, recipients are down the y axis, each cell represents the delivery ratio of the senders' messages from that recipients' node).

Smart Gossip Node 1 Delivery Matrix

SEND

RECV

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0.56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0.63	0	0	0.07	0.15	0.04	0	0	0	0	0	0.04	0.02	0	0	0.05	0	0	0	0	0
4	0.48	0	0	0	0.03	0	0	0	0	0	0	0	0	0	0	0.05	0	0	0.14	0	0
5	0.67	0	0	0	0	0.04	0	0	0.04	0.05	0	0.04	0.02	0	0	0	0	0	0	0	0
6	0.63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0.15	0.33	0	0	0	0	0	0	0	0	0.33	0	0	0	0	0	0	0	0
8	0	0	0.15	0.5	0.06	0	0.04	0	0	0.05	0	0	0.37	0.06	0	0	0	0	0	0	0
9	0.37	0	0	0	0.03	0	0	0	0	0.11	0	0	0	0	0	0	0	0	0	0	0
10	0.46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0.54	0	0	0.03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0.37	0	0.03	0.07	0.03	0	0	0	0	0	0	0	0.02	0	0	0	0	0	0.14	0	0
13	0.59	0	0.03	0.03	0.18	0	0	0.05	0.07	0.05	0.06	0.08	0	0	0	0.05	0	0	0	0	0
14	0	0	0.09	0.23	0	0	0	0.03	0	0	0	0.31	0.18	0	0.09	0.05	0	0	0	0	0
15	0	0	0.3	0.3	0	0	0	0	0	0	0	0.23	0	0.03	0	0	0	0	0.57	0	0
16	0	0	0	0.13	0	0	0	0	0	0	0	0	0.02	0	0	0	0	0	0.07	0	0
17	0	0	0	0	0	0	0	0.74	0	0	0	0	0	0.03	0	0	0	0.2	0	0	0
18	0	0	0	0	0	0	0.57	0	0	0	0	0	0	0.55	0	0.09	0.44	0	0	0	0
19	0.39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0.94	0	0	0	0	0	0	0	0
21	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11. Delivery Ratio Matrix for Smart Gossip using node 1 as a broadcasting node. (Sending nodes are along the x axis, recipients are down the y axis, each cell represents the delivery ratio of the senders' messages from that recipients' node).

Difference Matrix Between Smart Gossip and PAMPA

SEND

RECV

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0.44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	-0.63	0	0	0.38	-0.15	0.96	0	0.02	0	0.64	0.67	0.4	0.31	0	0	-0.01	0	0	0.2	0	0
4	0.3	0	0	0	-0.03	0.13	0	0.02	0	0	0.17	0.13	0.07	0	0	-0.05	0	0	0.12	0	0
5	0.28	0	0	0	0	0.08	0	0	-0.04	0.04	0.06	-0.04	-0.02	0	0	0	0	0	0	0	0
6	0.37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0.17	0.47	0	0	0	0.15	0	0	0	0	0.41	0.3	0	0	0	0	0	0	0
8	0	0	0.17	0.24	0.06	0.13	-0.04	0	0.05	0.49	0	0	0.23	0.2	0	0	0	0	0	0	0
9	0.48	0.04	0	0	-0.03	0.13	0	0	0	0.08	0.22	0	0	0	0	0	0	0	0.07	0	0
10	0.54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0.45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0.44	0.04	-0.03	-0.03	-0.03	0.25	0	0	0	0.18	0.17	0	-0.02	0	0	0	0	0	-0.08	0	0
13	0.1	0	-0.03	0.03	-0.18	0.38	0	-0.05	-0.07	0.22	0.22	-0.01	0	0	0	-0.05	0	0	0.27	0	0
14	0	0	0.04	0.28	0	0	0	0.1	0	0	0.5	0.25	0.28	0	-0.05	0.2	0	0	0	0	0
15	0	0	-0.21	0.38	0	0	0	0	0	0	0.33	0.52	0	0.1	0	0.24	0	0	-0.17	0	0
16	0	0.08	0	0.32	0	0	0	0	0	0	0.67	0.63	-0.02	0	0.02	0	0	0	0.66	0	0
17	0	0	0	0	0	0	0.56	-0.01	0	0	0	0	0	-0.03	0	0	0	-0.04	0	0	0
18	0	0	0	0	0	0	0.29	0	0	0	0	0	0	0.32	0	-0.09	-0.27	0	0	0	0
19	0.61	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0.9	0	0	0	0	0	0	0	0	0	-0.14	0	0	0	0	0	0	0	0
21	-0.12	0	0	0	0	0.13	0	0	0	0.09	0.17	0.06	0	0	0	0	0	0	0.07	0	0

Figure 12. Delivery Ratio Matrix representing the difference between the delivery ratios for PAMPA and Smart Gossip using node 1 as a broadcasting node. (Sending nodes are along the x axis, recipients are down the y axis, each cell represents the difference between PAMPA and Smart Gossips delivery results. A positive number means that PAMPA had a better Delivery Ratio (highlighted in green), whereas a negative number means Smart Gossip had a better Delivery Ratio (highlighted in red).