

# Formal Specification and Analysis of Performance Variation for Sensor Network Diffusion Protocols

Sule Nair and Rachel Cardell-Oliver  
School of Computer Science & Software Engineering  
The University of Western Australia  
Crawley WA 6009 Australia  
email: {sule,rachel}@csse.uwa.edu.au

June 17, 2004

## Abstract

Discovering a routing tree for gathering or disseminating streams of data is an important operation in many sensor network applications. However, previous work has shown that protocols for tree discovery may have significant performance problems for certain configurations of a network and its application task. A novel formal model for the behaviour of push and pull diffusion is introduced in order to analyse such problems. The executable protocol specification is written in the Temporal Logic of Actions (TLA). Designers can experiment with the performance of protocols and discover their strengths and weaknesses in different environments. In this paper special emphasis is placed on the routing trees discovered by diffusion and the effect of their shape and size on protocol performance, which is shown to be highly variable. The cost of pulling data up a routing tree increases with the number of data producers even if the tree has many shared branches. The cost of pushing data down the tree, however, grows at a much slower rate as the number of data consumers is increased. The size of routing trees discovered by diffusion varies significantly; small trees are observed to give better performance for push diffusion, but they do not improve the performance of pull diffusion. The modular structure of our TLA protocol model allows for independent specification of the network topology, physical layer, MAC layer protocols and network layer protocols. Thus it is straightforward to adapt the model for analysing many other classes of sensor network protocols.

**Keywords:** Sensor network, routing trees, directed diffusion, application performance, TLA, formal specification

# 1 Introduction

The new technology of wireless sensor networks promises fine-grained, real-time data collection over a landscape, allowing scientists to discover properties of those landscapes that have not previously been observable. However, in order to realise this promise, new protocols are needed which are both suitable for the given application, and which make good use of scarce resources such as the limited energy available to each node.

An important operation in many sensor network applications is the discovery of a routing tree for gathering or disseminating data streams in the network. The routing tree discovery problem has a seemingly simple solution: the root initiates a search for the tree's leaves and then a reverse wave of messages reinforce the tree's branches. However, previous work has shown that proposed algorithms for tree discovery may have significant performance problems for given configurations of a sensor network and its application [5, 9].

One method that has been used to detect and remove errors in protocols such as tree discovery, is to perform simulation on the source code using the TOSSIM simulation environment [9]. A disadvantage of this approach is the high cost of implementing a protocol in order to discover that it may not, in fact, be suitable for a given application. Ideally, we would like to verify the *algorithm* for this protocol first.

Another approach is to investigate protocol behaviour by simulations averaged over a range of randomised topologies and routing trees [5, 6]. A disadvantage of this method is that it does not help a designer who has a specific application in mind, to answer the question of how the algorithm will perform in that setting. In addition, significant variations in protocol performance (for example between two different random placements of nodes) may be masked by such analysis.

In this paper we present a formal specification for a family of routing tree discovery algorithms based on Heidmann et al's one-phase directed diffusion [5]. These specifications are also executable for a given network configuration, providing a means to debug proposed protocols and analyse their performance, before the expense of their implementation.

The contributions of this paper are

- a formal specification in the Temporal Logic of Actions [8] of push and pull diffusion; and
- an analysis of the performance variation inherent in local state tree discovery algorithms, and the causes of this variation.

Section 2 presents a formal specification in TLA [8] of the behaviour of the diffusion family of protocols. Metrics for the performance of these protocols are defined in Section 3. Section 4 details performance results for the protocols and explores the causes of performance variations. Related work and conclusions are described in Sections 5 and 6.

## 2 Assumptions and Analysis Model

The behaviour of a wireless network protocol is characterised by 1) the protocol rules followed by each node in the network and 2) the message delivery service provided by the underlying MAC and physical layers of the network. In previous work we have presented an *extensional* specification of these properties for a flooding protocol using a CSMA MAC protocol, broadcast messages, and noisy radio signals [1]. In this paper, we present an *intentional* specification of ISI's S-MAC and physical layer protocols [14] and the protocol rules and message delivery service for push and pull diffusion[5].

### 2.1 Two Diffusion Protocols

Data diffusion protocols discover and construct a tree of data paths for routing streams of data from a set of producers to a set of consumers. In the *push* diffusion protocol a producer initiates the discovery of a routing tree with itself as root to one or more consumer leaves. It then reinforces the tree using messages from the leaves to the root, and thereafter pushes data down the tree from the producer to the leaf consumers. In the *pull* diffusion protocol a consumer advertises its interest in data to a set of producers, and so initiates the discovery of a routing tree from the consumer root to one or more producer leaves. Thereafter it pulls data up the tree from the producers to the consumer. Since each node discovers its parent during the exploration round, there is no need for a reinforcement phase in pull diffusion.

Examples of routing trees discovered by the diffusion protocol are shown in Figure 1. In Section 4 we analyse the wide variety of tree size and shape that can be discovered, and explain why this happens. In this paper we focus on diffusion with one to many trees, but the extension of our model to the more general case of many to many diffusion is straightforward.

Diffusion protocols use three types of message exchange: explore (also called interest), reinforce and data. Explore messages are broadcast through the network creating for each node a parent node. When a node receives the first explore message, it assigns the sender of this message as its parent.

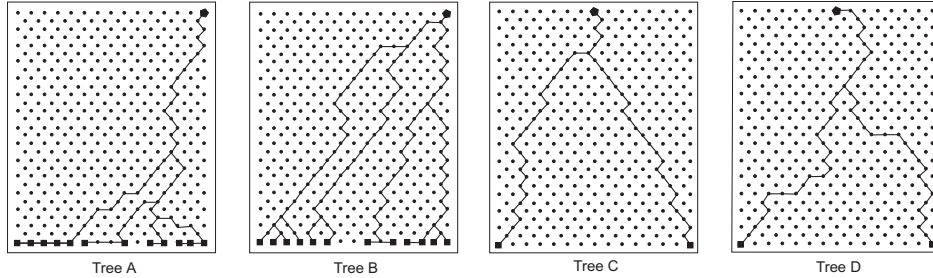


Figure 1: Examples of Discovered Routing Trees

Explore messages thus create paths between the root of the tree and its leaves. Reinforce messages are used only in push diffusion, where they are unicast from a child node to its parent in order to construct the links used to push data from the root to its leaves. A node's *children* are the nodes that form its next hop towards the routing tree's leaves. A node's *parent* is the node that provides its next hop towards the routing tree's root. Data messages, also unicast, are used to stream data from producers to consumers by pushing or pulling data messages through the constructed routing tree.

For lack of space, we can only include parts of the full TLA specification of diffusion in this paper. We show only the *push diffusion* definition here. A full specification can be found at [10].

## 2.2 Specification of the Diffusion Protocol

Initially, the Producer node in push diffusion is ready to forward an exploration message (*FwdExp*) and all other nodes (Routers and Consumers) await such a message (*WaitExp*).

We use a simple energy model in which the energy used by each node in the protocol is assumed to be proportional to the number of bytes transmitted [5]. Each rule for explore, reinforce, and data transmission includes constraints of the form,

$$\begin{array}{l}
 \wedge \text{node}[n].\text{energy} \geq \text{ExpDataEnergy} \\
 \wedge \text{node}[n].\text{energy}' = @ - \text{ExpDataEnergy}
 \end{array}$$

In TLA a primed variable denotes the value of a state variable after

the operation and the unprimed form its value before the operation. For a comprehensive introduction to TLA see [8].

### Explore Rule

A node in state *FwdExp* broadcasts the exploration message to all its neighbours. The definition of the *transmitters* set of nodes will be detailed in Section 3.2. Each node receiving its first explore message makes the sender its parent. It then either waits for a reinforce message, or if it is a Consumer, it generates a reinforce message. The receiving nodes forward the explore message.

$$\begin{aligned}
 \text{SendExplore}(n) &\triangleq \\
 &\wedge n \in \text{transmitters} \\
 &\wedge \text{node}[n].\text{state} = \text{FwdExp} \\
 &\wedge \text{node}[n]' = [ \text{node}[n] \text{ EXCEPT } \text{!.state} = \\
 &\quad \text{IF } (n \in \text{Consumer}) \\
 &\quad \quad \text{THEN } \text{FwdRef} \\
 &\quad \quad \text{ELSE } \text{WaitRef} ] \\
 &\wedge \forall r \in \text{broadcast\_receivers}(\{n\}) : \\
 &\quad \wedge \text{node}[r].\text{state} = \text{WaitExp} \\
 &\quad \wedge \text{node}[r]' = [ \text{node}[r] \text{ EXCEPT} \\
 &\quad \quad \text{!.parent} = n, \\
 &\quad \quad \text{!.state} = \text{FwdExp} ]
 \end{aligned}$$

### Reinforce Rule

Once the explore message reaches a leaf node then reinforce messages are forwarded to parent nodes and so on until the root of the tree is reached. Since several nodes may share the same parent, nodes must be prepared to accept reinforce messages as well as data in future rounds, and to recognise all the senders of reinforce messages as its children.

$$\begin{aligned}
 \text{SendReinforce}(n) &\triangleq \\
 &\wedge n \in \text{transmitters} \\
 &\wedge \text{node}[n].\text{state} = \text{FwdRef}
 \end{aligned}$$

$$\begin{aligned}
& \wedge \text{node}[n]' = [ \text{node}[n] \text{ EXCEPT } !.state = Data ] \\
& \wedge \text{LET } p = \text{node}[n].parent \text{ IN} \\
& \quad \wedge \text{node}[p].state \in \{ WaitRef, Data \} \\
& \quad \wedge \text{node}[p]' = [ \text{node}[p] \text{ EXCEPT} \\
& \quad \quad \quad !.state = FwdRef, \\
& \quad \quad \quad !.children = @ \circ \langle n \rangle ]
\end{aligned}$$


---

### Data Rule

Once a routing tree has been discovered, data is pushed through the tree from the Producer root until it reaches Consumer leaf nodes. A queue is used to hold the addresses of data messages awaiting transmission.

---

$$\begin{aligned}
\text{SendData}(n) & \triangleq \\
& \wedge n \in transmitters \\
& \wedge \text{node}[n].state = Data \\
& \wedge \text{Len}(\text{node}[n].data\_queue) > 0 \\
& \wedge \text{node}[n]' = [ \text{node}[n] \text{ EXCEPT } !.data\_queue = Tail(@) ] \\
& \wedge \text{LET } next = \text{Head}(\text{node}[n].data\_queue) \text{ IN} \\
& \quad \wedge \text{node}[next]' = [ \text{node}[next] \text{ EXCEPT} \\
& \quad \quad \quad !.data\_queue = \text{IF } (next \in Consumer) \\
& \quad \quad \quad \quad \text{THEN } \langle \rangle \\
& \quad \quad \quad \quad \text{ELSE } queue \circ \text{node}[next].children ]
\end{aligned}$$


---

### 2.3 Message Delivery

The diffusion protocol rules of Section 2.1 use the variable *transmitters*: the set of nodes able to transmit a message in a given state. Transmitters are determined by 1) the current state of a node in the Protocol and 2) the underlying activity of the MAC and physical layers. In a wireless network nodes must compete for the landscape in order to transmit their messages by radio broadcast. Thus the set of *transmitters* is a subset of the *ready-to-send* nodes determined by the protocol. Similarly, the set of nodes *receivers* in a given round is a subset of *ready-to-recv* nodes. To avoid deadlocks, all nodes are always ready to receive.

---


$$\begin{aligned}
&\wedge \text{ready\_to\_send} = \{n : \text{Nodes} \mid \\
&\quad (\text{node}[n].\text{state} = \text{Data} \wedge \text{Len}(\text{node}[n].\text{data\_queue}) > 0) \vee \\
&\quad (\text{node}[n].\text{state} \in \{\text{FwdExp}, \text{FwdRef}\})\} \\
&\wedge \text{ready\_to\_recv} = \text{Nodes} \\
&\wedge \text{transmitters} \subseteq \text{ready\_to\_send} \\
&\wedge \text{receivers} \subseteq \text{ready\_to\_recv}
\end{aligned}$$


---

When a node transmits it creates a *footprint* of radio signals. For broadcast messages the footprint consists of all neighbouring nodes. For S-MAC unicast messages a RTS-CTS-DATA-ACK handshake is used [14]. Thus a unicast footprint includes both the sender's and the receiver's neighbours.

The function *txtype* identifies the type of message (broadcast or unicast) being transmitted and *txdst* the destination node for a unicast message. The sending node, and also the receiving node in unicast, are not members of their own footprint sets.

---


$$\begin{aligned}
\text{footprint}(t : \text{Node}) &\triangleq \\
&\text{CASE} \\
&\quad \text{txtype}(t) = \text{unicast} \\
&\quad \rightarrow \text{node}[t].\text{neighbours} \setminus \{\text{txdst}(t)\} \cup \text{node}[\text{txdst}(t)].\text{neighbours} \setminus \{t\} \\
&\quad \square \\
&\quad \text{txtype}(t) = \text{broadcast} \\
&\quad \rightarrow \text{node}[t].\text{neighbours}
\end{aligned}$$


---

In the SMAC protocol, nodes share a schedule to ensure that all nodes co-ordinate a cycle of sleeping (to save energy) and wake up for an interval of time during which any node wishing to transmit or receive may try to do so. When a *set* of nodes transmit during an SMAC interval, the landscape is filled with signals from all the transmitting nodes. Whenever signals from a set of transmitting nodes overlap in the landscape it causes signal corruption at the nodes in these regions.

---


$$\wedge \text{signalled} = \bigcup_{t \in \text{transmitters}} \text{footprint}(t)$$


---

$$\wedge \text{corrupted} = \{s \mid \exists t_1, t_2 : \text{transmitters.} \\ s \in \text{footprint}(t_1) \cap \text{footprint}(t_2)\}$$

---

To reduce the risk of signal corruption, wireless networks use *carrier sense*: each node listens to the landscape before it attempts to transmit. If a node hears that a transmission is in progress then it does not transmit. In terms of our specification, no node can be a good sender and also a signalled node. For unicast transmission, both the sender and receiver must not hear any other transmission. Also, a unicast receiving node can not itself be a transmitter. On the other hand, every node that is both ready and able to send will do so.

---


$$\begin{aligned} \text{carriersense}(t : \text{Node}) &\triangleq \\ \text{CASE} \\ \text{txtype}(t) = \text{unicast} &\rightarrow \{t, \text{txdst}(t)\} \\ \square \\ \text{txtype}(t) = \text{broadcast} &\rightarrow \{t\} \\ \wedge \forall t : \text{transmitters.} &\text{carriersense}(t) \cap \text{signalled} = \{\} \\ \wedge \text{transmitters} \cap &\text{unicast\_targets}(\text{transmitters}) = \{\} \\ \wedge \forall n : \text{Nodes. } n \in &\text{ready\_to\_send} \setminus \text{transmitters} \\ &\Rightarrow \text{carriersense}(n) \cap \text{signalled} \neq \{\} \end{aligned}$$


---

The transmission behaviour of a wireless network is non-deterministic because each autonomous node waits a random time before attempting to transmit. So we can not predict in advance which nodes will actually send in a given SMAC transmission interval. Our intentional specification of *transmitters* captures this non-determinism. It defines the constraints on the set of nodes chosen, but not which possible set of nodes is selected. When executing this specification, we construct the transmitter set by selecting at random one ready to send node at a time, and adding its signal to the landscape if the carrier is free. When all signals have been added to the landscape we can identify areas of clear signal and of corruption, and thus calculate which nodes wishing to receive a message are able to do so in the current state. This simulation strategy, based on local signal information, is very efficient and scales well for large networks [11].

---


$$\begin{aligned}
unicast\_targets(T) &\triangleq \\
&\{d : Node \mid \exists s : Node \in T. txtype(s) = unicast \wedge txdst(s) = d\} \\
broadcast\_targets(T) &\triangleq \\
&\{d : Node \mid \exists s : Node \in T. \\
&\quad txtype(s) = broadcast \wedge d \in node[s].neighbours\} \\
broadcast\_receivers(T) &\triangleq broadcast\_targets(T) \setminus corrupted \\
\wedge receivers &= unicast\_targets(transmitters) \cup \\
&\quad broadcast\_receivers(transmitters)
\end{aligned}$$


---

### 3 Performance Metrics for Tree Discovery

The main performance measure in this study is the cost of data dissemination or gathering during a data interval. The cost involves successful transmission of the data message from the producer to all the consumers in push and from all the producers to the consumer in pull. This cost is measured in bytes as it was done in [5]. The experiments were repeated with varying numbers of leaf nodes (from 1 to 15), placed randomly on the lowest row of the landscape. The root node was also placed randomly on the highest row of the landscape. The landscape is a regular hexagonal grid network of 421 nodes as shown in Figure 1. In order to make comparisons easier with earlier work [5], we measure the energy used in bytes transmitted and use the message sizes of [5]: 68B for explore and interest, 104B for reinforce and 84B and 92B for data messages in push and pull (respectively). In other applications we would expect control messages to be smaller. For example, in our environmental monitoring soil moisture application data messages are approximately 100B and control messages 20B. Also, as in [5], we do not use aggregation or duplicate suppression of messages. Simulations are repeated 100 times for each number of producers and consumers.

For each phase of the two protocols we measure:

- the amount of energy used;
- the number of packets transmitted;
- the number of cycles spent;

For each protocol run:

- the size (number of edges) of the routing tree;
- the shape of the tree

## 4 Performance Results

It has been reported that pull diffusion displays optimised and consistent performance for a few producers, and push displays optimised and consistent performance for a few consumers [5]. Our tests explore the relationship between the varying number of producer and consumer nodes and the performance of the algorithms. We also investigate the relationship between the tree size and shape and the performance of the algorithms. We provide baseline representation of the data we collected, to show that there is growth in the cost per data interval, as the number of producers and consumers increase for the respective algorithms.

We test a number of hypotheses concerning the relationships between the variables: cost per data interval, number of producers and consumers, tree size and tree shape.

**Hypothesis A.** The cost of data flow up the routing tree grows linearly as the number of producers grows in pull diffusion; and the cost of data flow down the routing tree grows slowly as the number of consumers grows in push diffusion.

**Hypothesis B.** Tree size and shape varies widely in both push and pull diffusion.

**Hypothesis C.** Smaller tree size gives better performance in push. However, smaller tree size does not give better performance in pull, except for the one consumer and one producer configuration.

We examine each of these hypotheses and present the results of the simulator tests.

### 4.1 Cost of Data Delivery through Routing Trees

Pull diffusion (called one-phase pull in [5]) is a routing algorithm optimised for configurations with multiple producers. Each producer sends a data message to the consumer, every data interval. Figure 2 illustrates the cost of sending data messages up the routing tree per data interval for a varying number of producers. The interest messages are initiated by the consumer and flooded through the network. The cost of flooding is proportional to

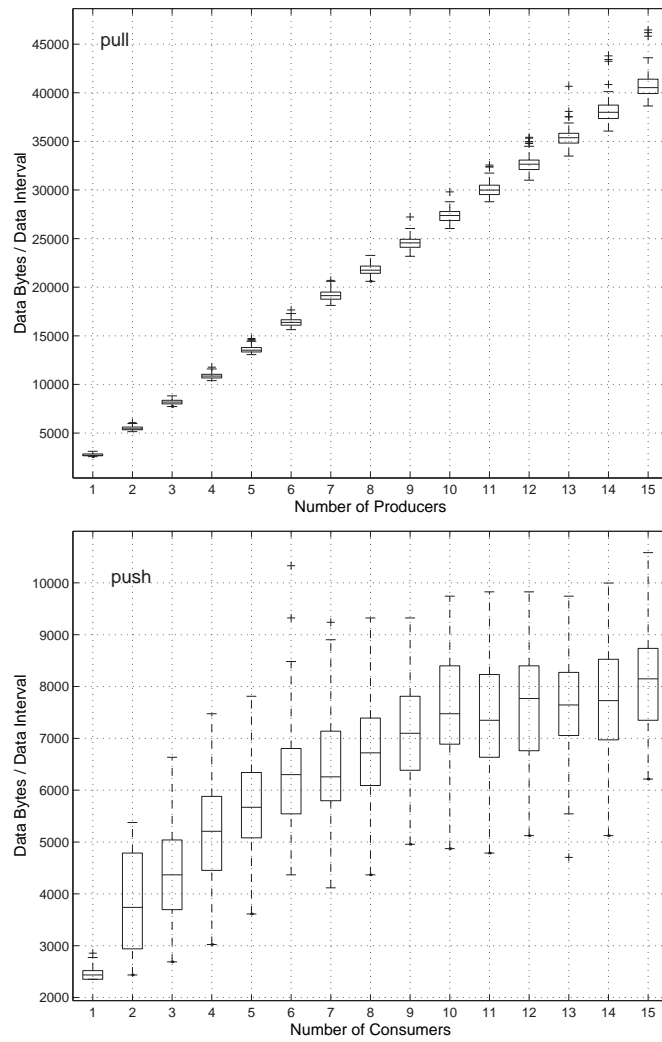


Figure 2: Boxplots of cost per data event for varying numbers of producers in pull and consumers in push

the size of the network, which we held constant for all the experiments. An identical linear growth was obtained, including the cost of control messages which is not included here. The linearly increasing cost per data interval shown here is also in line with earlier work, where the control messages were included in the calculation of cost per data event [5].

We have looked at the distribution of the cost data in order to get a better feel about the variations between different runs of the protocol, for the same number of producers. Figure 2 reveals the variability of the performance between multiple runs of the protocol, which is more emphasized as the number of producers grows. This is to be expected, as the cost depends on the number of hops, which is elicited by a non-deterministic tree discovery method, using flooding and local information only.

Our results, together with results from [5], provides support for the first half of the hypothesis Hypothesis A, which deals with (one-phase) pull.

Push is suited for applications with one producer and multiple consumers. The producer sends a data message down the routing tree, which is delivered to potentially many consumers. [5] (see Figure 4b) found that the cost per data event was nearly constant as the number of consumers grew. Our results contradict [5] since in our test configuration the cost per data event actually increases near linearly as the number of consumers grows as shown in Figure 2.

Push diffusion performance benefits from trees with shared branches, since data for many consumers need only be transmitted once down such branches. The network configurations tested in [5] are narrower than ours, since both producers and consumers are randomly selected from local areas at the top right and bottom left corners of a random node topology. Thus in the 5 protocol runs tested for each configuration in [5], we would expect to generate many trees with shared branches, and so the increase in cost as number of consumers grows would be near constant. For our wider trees the cost increase is near linear, although at a much slower rate than in pull diffusion (note the y axes scale in Figure 2).

Figure 2 also reveals the wide variability of cost between different runs of the protocol with the same number of consumers. Again this is a result of the non-deterministic nature of the route discovery method. The degree of variability in cost is similar to that of pull diffusion. The effect of the increase in the tree size is less in push, leading to a slower increase in the cost per data interval. This is due to the shared branches that exist in the discovered trees.

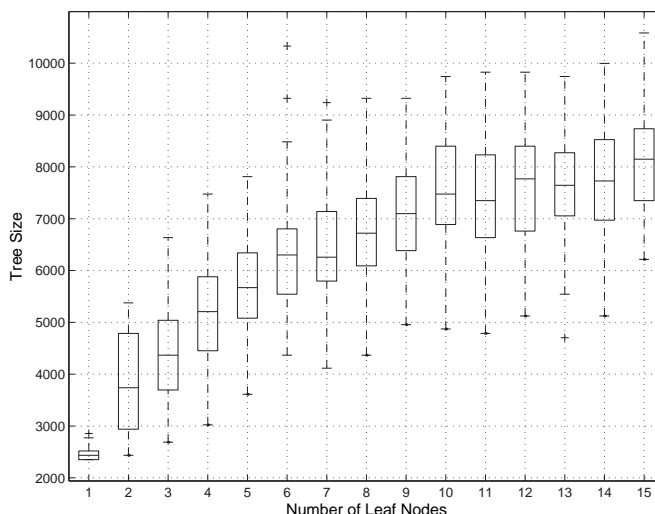


Figure 3: Boxplots of tree size for varying numbers of node leaves

## 4.2 Variation in Tree Size

The data collected from the protocol runs using pull or push diffusion give the same tree results since the flooding method used to discover trees in both is the same. Figure 3 shows the size of the trees discovered during the pull and push simulation runs. We see an overall increase in the number of edges included in the tree as the number of leaf nodes grows. This result is indicative of the irregular shape of trees created, which depart from the ideal as they connect more leaf nodes to the root node. Figure 3 shows that there is an upward trend in the tree sizes possible as the number of the leaf nodes grow. However, the variation in the individual tree sizes between multiple runs of the protocol is wide. These results demonstrate the unpredictability of the tree size based on the consumers and provides evidence for hypothesis Hypothesis B.

The results summarised in this section, together with hypothesis A, demonstrate that using an average hop size to cover the network does not lead to a realistic assumption of an average tree size. As the tree size is highly unpredictable for a particular protocol run, so is the cost of transmitting data messages.

### 4.3 The Effect of Small Trees on Performance

In investigating pull diffusion, we not only discovered strong support for hypothesis A, but also found evidence to support the hypothesis: Smaller tree size does not imply better performance in one-phase pull. Our results show that the total number of edges that constitute the routing tree does not correlate with the total cost of sending data messages from the producers to the consumer. In Figure 4, the data are presented as scatter plots between the tree size and the data sending cost in one data interval. In the case of one producer and one consumer, all the 100 protocol runs fall on one of seven points. There is a clear linear correlation between the cost and the tree size in this case because there is only one data event for which a data message is created, and this message is transmitted through a tree which comprises of only one branch.

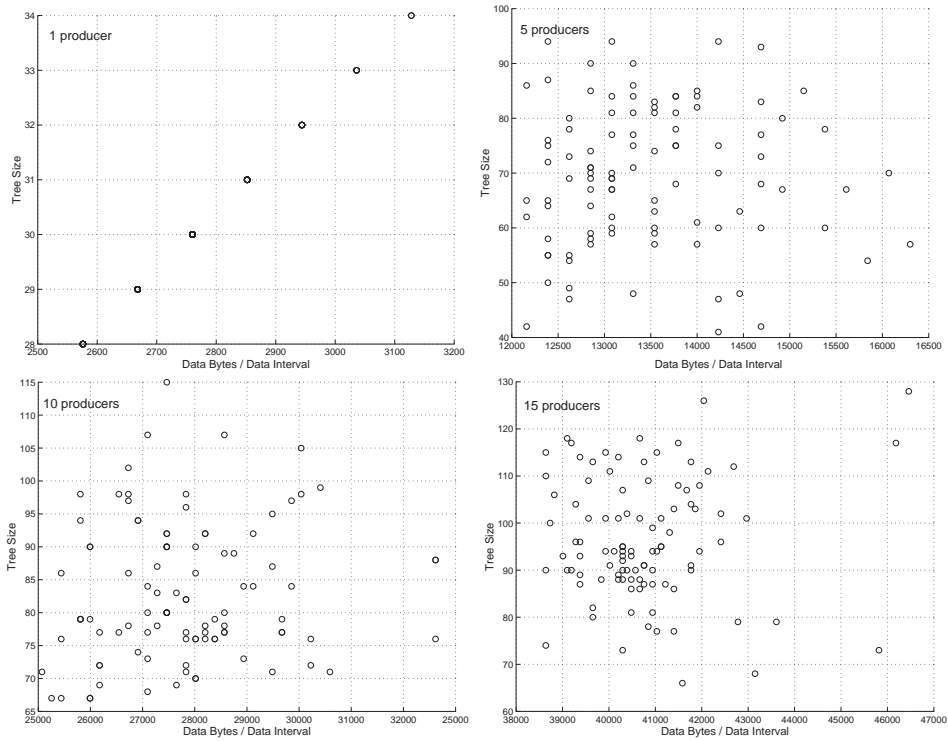


Figure 4: Scatter plot of tree size against the cost per data interval for varying numbers of pull producers. Each circle represents one run of the protocol

As the number of producers increase, the data points gradually fall into wider intervals and become less clustered. Plotting the protocol runs with varying numbers of producers from 1 through to 15 illustrates this gradual trend. We only show the plots for 1, 5, 10 and 15 producers here for brevity. The plots in 4 reveal that the same tree size can lead to varying cost in different runs of the protocol with the same number of producers. In pull diffusion the dominating factor affecting the cost is the length of the route from a specific producer to the consumer. Therefore, a large tree, with a large number of long branches linking producers to the consumer, can provide good performance.

The rationale behind the first part of hypothesis C (Smaller tree size implies better performance in push) is that a tree with minimum number of edges will connect the producer and multiple consumers with common paths, carrying the data message originated from the single producer to the closest possible node before branching out to reach the individual consumers. The data we have collected showed strong evidence to support this rationale and hence hypothesis C.

The scatter plots between tree size and cost per data interval for push diffusion in Figure 5 show clear linear correlation. All the data points for different protocol runs are located on a prominently straight line, indicating positive linear relation between the two variables. Although they are not shown here, we have plotted the data we collected for each number of consumers from one to fifteen and have not observed any discrepancies.

Figure 1 shows routing trees discovered during runs of the push and pull protocols. In runs A and B, and in C and D, the locations of the root and the leaf nodes are identical. Despite the identical node configuration, the algorithm discovers two trees displaying entirely different characteristics. Tree A has 62 edges and Tree B has 115 edges. The data dissemination cost in the push protocol is directly proportional to the size of the routing tree and so these two trees display very different performance during the data intervals they are used. For pull diffusion, the larger tree B actually gives better performance than tree A because the average length of paths from the producers to the consumer is shorter. The sizes of Tree C and Tree D are 52 and 53, respectively. Although, there is only one edge difference in the tree size, the difference in total cost of one data interval for push diffusion is 552B, which corresponds to 6 more hops in Tree D. The problem of finding an optimal routing tree is NP complete [2, 6]. Effective heuristics for doing so are a topic for future work.

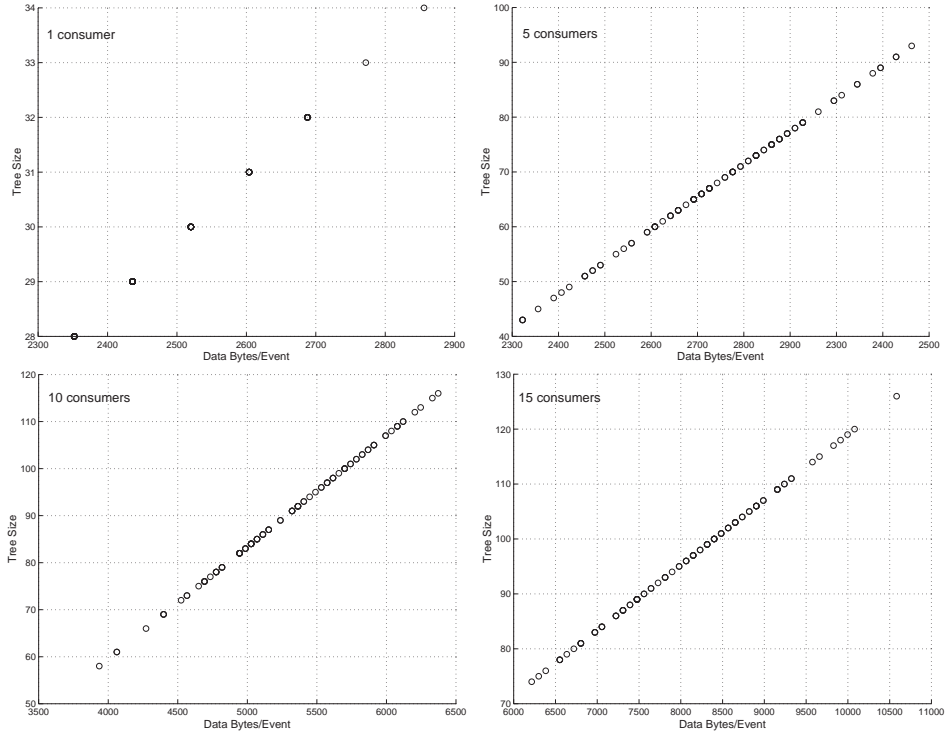


Figure 5: Scatter plot of tree size against the cost per data interval for varying numbers of push consumers. Each circle represents one run of the protocol

## 5 Related Work

Cardell-Oliver [1] defines the behaviour of wireless sensor networks using a formal model; analyses the likelihood of flooding failure under different physical network topologies; and documents the effects of network topology on the performance of reliability and efficiency, in particular. Both node density and placement are considered and analysis is based on formal verification and simulation experiments. A simulation based analysis based on this model is presented in [3]. Our work extends the formal model in [1] to include S-MAC broadcast and unicast, a richer protocol specification model and the energy levels of the nodes are taken into consideration. Our work also extends [1] by adding the specifications of the diffusion family of data dissemination and gathering protocols. Another state-oriented model of ad-

hoc network nodes is presented in [13]. Although, the node states proposed are useful for keeping track of the network state, they do not provide a formal specification of the overall behaviour of the network. The node model is used to resolve topology through dissemination of node states in the context of delivering QoS in mobile ad-hoc networks.

A line of work focuses on improving the effectiveness of data dissemination and gathering by optimising the tree-based routing topologies. [5], for instance, defines one-phase pull and push diffusion algorithms and their comparative analysis based on communication patterns of particular applications and the number of consumer and producer nodes in the network. Analysis is based on data collected by using the real application code in an emulation environment: a method much costlier than ours. [7] studies the performance of different data aggregation algorithms under different network topologies. They consider the number of producer and consumer nodes, the distances (in hops) between nodes, and the structure of the routing tree, which is predefined. They do not consider the effect of widely varying shapes and sizes of the routing trees discovered by the protocols under investigation and so disregard the highly non-deterministic nature of the tree discovery procedure used in these protocols.

Another technique for analysing protocol performance is to simulate protocol source code using, for example, the TOSSIM simulation environment [9]. This approach has been shown to be successful for detecting a flaw in the Surge routing protocol that caused a high end-to-end loss rate. Although queue overflow is not a problem for the configurations presented in this paper, our model has detected similar problems when simulating a real sensor network of 22 nodes monitoring soil moisture with a 1% SMAC duty cycle and noisy transmission. Lost packets, which may have been caused by this problem, were also observed in tests with the physical implementation of this network. The TinyOS code we developed for this implementation represents a much greater effort than encoding the protocol in our abstract model, and so it is desirable to remove such bugs from the protocol algorithm rather than its code wherever possible.

A similar study to ours in terms of analysing the relation between the cost of data gathering and the routing structure is presented in [4]. The aim of the study is to understand if there are easy-to-deploy configurations other than random placement of nodes on a uniform grid that can give important gains, in lowering power consumption. A wheel structure is given as an example of an efficient two dimensional placement and transmission structure. A wheel network is dense closer to the consumer where the data load is higher, and sparse further away from the sink. The study provides

heuristics for designing network configurations for data gathering (pull). Data dissemination (push) is not considered.

## 6 Conclusions and Future Work

The main result of this paper is that there are wide variations and inconsistencies in the performance of the diffusion family of routing protocols; and that this is the consequence of non-determinism in the tree building procedure, which entirely relies on nodes' local information. Pull diffusion displays linearly increasing cost as the number of producers increase. Push, on the other hand, demonstrates slowly increasing cost as the number of consumers grow.

We have demonstrated the importance of tree shape and size in the performance of the algorithms. In applications wthat use non-frequent route re-discovery intervals, the network can be stuck with a high-cost routing tree for a long period of time. Increasing the frequency of the tree re-discovery interval, on the other hand, will cause an increase in the control overhead, which may not be justifiable in applications, such as environmental monitoring, with non-frequent data events and requirements to maximise network lifetime. A complete guarantee of high performance may not be possible as the tree discovered each time is independent of previous runs of the protocol. We have also simulated diffusion with noisy radio transmission. The configuration tested in this paper was shown to be fairly robust to the noisiness of wireless transmissions, although as noise increases, so does the cost of data delivery in proportion to the message loss rate because of retransmissions [10].

Alternatives to flooding in route discovery do exist, but they are not applicable to the family of routing algorithms we have analysed. For instance, the rendezvous technique [6] requires global information. Geographic routing is another alternative. However, this approach is only beneficial if the leaf nodes are clustered together; in our configuration geographic routing would not reduce tree size.

Our results show the importance of research effort into heuristic methods, which will help discover efficient routing trees, where only local information is available. These results can then be used to optimise existing and new data dissemination and aggregation algorithms [12]. There will be a net increase in protocol performance and predictability.

We have demonstrated the comparability of our methodology to other more expensive methods in verification and performance measurement of

algorithms, such as running simulations with the real application code [9], simulation with a fined grained network model [11], or observing real applications [9, 5].

Our future work will include investigation of the impact of network congestion and aggregation of data messages using our methodology. Although congestion was not a problem with the configurations and assumptions we used in this study, it may cause loss of packets due to buffer overflow. We will study different configurations with different sets of assumptions, which reflect the problems that may occur in a real network, including radio noise models. We will also generalise the costs involved in the applications of the routing algorithms under investigation and provide proofs, for instance, characterising the cost of tree discovery and the cost per data interval.

## References

- [1] Rachel Cardell-Oliver. Why Flooding is Unreliable. Extended Version, Technical Report UWA-CSSE-04-001, Feb 2004. [Online] Available at [http://www.cs.uwa.edu.au/research/Technical\\_Reports/UWA-CSSE-04-001/Report-04-001.pdf](http://www.cs.uwa.edu.au/research/Technical_Reports/UWA-CSSE-04-001/Report-04-001.pdf). Submitted for Review.
- [2] Razvan Cristescu, Baltasar Beferull-Lozano, and Martin Vetterli. On network correlated data gathering. In *INFOCOM*, 2004. To appear.
- [3] Patrick Downey and Rachel Cardell-Oliver. Evaluating the impact of limited resource on the performance of flooding in wireless sensor networks. In *International Conference on Dependable Systems and Networks, Florence*, July 2004. To appear.
- [4] Deepak Ganesan, Razvan Cristescu, and Baltasar Beferull-Lozano. Power-efficient sensor placement and transmission structure for data gathering under distortion constraints. In *Proceedings of the Symposium on Information Processing in Sensor Networks 2004, Berkeley, California*, 2004.
- [5] John Heidemann, Fabio Silva, and Deborah Estrin. Matching data dissemination algorithms to application requirements. In *Proceedings of the first international conference on Embedded Networked Sensor Systems*, pages 218–229. ACM Press, 2003.
- [6] Bhaskar Krishnamachari, Deborah Estrin, and Stephen Wicker. Modelling data-centric routing in wireless sensor networks. In *Proceedings of IEEE Infocom 2002*, 2002.

- [7] Bhaskar Krishnamachari and John Heidemann. Application-specific modelling of information routing in wireless sensor networks, usc isi technical report isi-tr-676, August 2003.
- [8] L. Lamport. Specifying Concurrent Systems with TLA+. In M.Broy and R.Steinbruecken, editors, *Calculational System Design*, number 173 in Series F: Computer and Systems Sciences, pages 183–247, Amsterdam, 1999. IOS Press.
- [9] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proc. Second ACM International Workshop on Wireless Sensor Networks and Applications (SenSys 03)*, 2003.
- [10] Sule Nair and Rachel Cardell-Oliver. Analysis of diffusion in sensor networks. [Online] <http://www.csse.uwa.edu.au/~sule/diffusion> as of 1 July 2004.
- [11] Verleri Naoumov and Thomas Gross. Simulation of large ad hoc networks. In *Proc. 6th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 03)*, 2003.
- [12] Gyula Simon, Peter Volgyesi, Mikles Maroti, and Akos Ledeczi. Simulation-based optimization of communication protocols for large-scale wireless sensor networks. In *2003 IEEE Aerospace Conference*, March 2003.
- [13] John A. Stine and Gustavo de Veciana. A paradigm for quality of service in wireless ad-hoc networks using synchronous signaling and node states. *IEEE JSAC Special Issue on Quality of Service Delivery in Variable Topology Networks*, To appear. Accepted March 2004.
- [14] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE Transactions on Networking*. To appear, 2003.