EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH **Open Access**

Comparing wireless flooding protocols using trace-based simulations

Martin Jacobsson* and Christian Rohner

Abstract

Most wireless multi-hop networks, such as *ad hoc* networks and wireless sensor networks, need network-wide broadcasting, which is best done with a flooding protocol. In this article, we use packet trace information from a real test-bed network to define a simulator for flooding protocol performance studies. Five protocols are compared using the simulator. Trace-based simulations promise to have the benefits of the simulator, such as reducing required work effort and repeatability but still produce results close to the real test-bed or deployment. We propose and evaluate different approaches on how to use collected trace data and how to tune the parameters to achieve the best possible accuracy in comparison with actual test-bed measurements. We study the resulting accuracy of the model so that performance studies know with what confidence a certain conclusion can be made. Using the new trace-based model and knowing its accuracy, we compare the five flooding protocols to gain additional insights into their performance. Finally, by modifying the trace data, we study how real-world effects, such as links with in-between qualities and asymmetric links, influence the different flooding protocols.

1 Introduction

In some wireless networks, a node may not be able to directly transmit a packet to every node in the network due to transmission range limitations. Instead, it needs help from other nodes to relay the packet to the destination. Such wireless networks are called multi-hop networks and require special networking protocols. Examples are mobile *ad hoc* networks (MANETs), wireless sensor networks (WSNs), and mesh networks. Most wireless multi-hop networks also need network-wide broadcasting. Just like unicast routing, this is only possible if some of the nodes relay the broadcasting packet so that all nodes can be reached. This process is called *flooding* and is used by multiple other protocols and applications, including unicast routing protocols.

This article builds on our earlier work on flooding protocols [1,2], where we proposed a flooding protocol called prioritized flooding with self-pruning (PFS) and compared it to other common flooding protocols. In [1], we proposed the first version of this protocol and simulated it in the standard configuration of ns2 [3], which is a very popular simulation package for wireless networking. In [2], we implemented the same protocol in a real wireless network test-bed and again measured its performance and compared it to blind flooding and counter-based broadcasting (CBB) [4]. We found that PFS had problems in real networks, and we had to modify the protocol. We also noticed discrepancies in the performance results obtained in the simulator in comparison to what we observed in the real network. Hence, we wanted to better understand this and see if it is possible to find a model that more closely simulates a real network. With such a simulator, it becomes possible to study more parameters and more protocols compared to a test-bed but without loosing important real-world aspects.

That the results from simulator studies of wireless networks have problems is not new and has been known for many years. For instance, [5-8] compared common MANET or WSN simulators and showed radically different results due to different radio propagation models, frame reception models, and other assumptions. Kotz et al. [9] looked at the impact of common simplifications in *ad hoc* network simulations and showed drastic errors in comparison with more accurate assumptions. Hence, to use simulators, we must first validate the used models.

The aim of this work was to find simulator methods that will be able to tell how flooding protocols perform in real networks with a higher degree of confidence.

*Correspondence: martin.jacobsson@it.uu.se
Department of Information Technology, Uppsala University, PO Box 337,Uppsala 751 05,Sweden

To know whether we are successful, we will benchmark the simulator results with the real network observations that we have and try to find simulator models whose results are closer to the real network. A model that is closer to the real network is simply said to be more accurate. More accurate simulations allow us to use simulators to study the performance of various protocols and be more confident that the results also hold in the real network. On the negative side, there is a question on how general these results will be, given that we only study a handful of network deployments. To properly answer this question, the work in this paper should be extended to more network deployments. It is then possible to answer if the same model performs just as good in other scenarios and hence can be considered generic.

For the sake of this study, we say that there are two types of simulation models for radio propagation and channel characteristics: synthetic models and trace-based models. *Synthetic models* are purely based on mathematical models derived from a few observations. With them, you can create any scenario you like. However, they need to make a lot of assumptions and simplifications. The most common model, the two-ray ground reflection model, popular in ns2 and many other simulators, is known to be overly simplistic. Nevertheless, even more advanced synthetic models can give erroneous performance results (e.g., see [10]). It is simply not possible to take into account all the different aspects of wireless communication. In particular, many wireless simulations based on synthetic models make unrealistic assumptions, such as

- Omni-directional antennas;
- Symmetric links;
- Links without in-between quality;
- Too simplistic radio propagation (no or limited shadowing and multi-path);
- Too simplistic frame-reception model (e.g., w.r.t. interference);
- Frame length-independent reception.

The alternative to synthetic models is to collect data traces from real networks and use that in the simulator (e.g., [11,12]), a technique known as *trace-based simulation*. Trace-based simulators limit you to the scenarios of the networks where the traces came from. However, they still have all the benefits of a simulator, such as easier to use, repeatable, and speed. They also promise to produce more accurate simulation results, at least as long as you simulate the network from where the traces were collected.

During our earlier test-bed experiments [2], we collected general data about the network and its topology. These data include the packet delivery ratio between all node pairs and should be enough for a trace-based simulation. In this article, we will use that data to see whether we can mimic the test-bed behavior in the simulator and, if so, continue to simulate additional protocols that we did not have the time and resources to do in the test-bed. Further, we can also modify the traces to simulate what happens in special cases, such as stress-testing the protocols or remove certain real-world effects to see what impact such effects have on the protocols. Finally, better simulation models may also be used in other areas besides performance simulations, such as deployment optimizations [13], where we collect traces from an actual deployment, run simulations based on that to determine the best protocols and parameters for that particular deployment.

Hence, the contributions of this article is twofold: First, we will show how our collected data can be used in a trace-based simulation and how this leads to results closer to the real network than synthetic models. Secondly, we will use this model to do a performance comparison of five different flooding protocols.

This article is organized as follows. Section 2 introduces a simulation study based on the commonly used two-ray ground reflection model of ns2, investigates the performance of two radically different flooding protocols using this model, and compares these results with test-bed measurements. In Section 3, we introduce a new simulation model based on collected packet traces from the test-bed, and in Section 4, we tune the model parameters and study its accuracy in comparison with the test-bed measurements. In Section 5, we use the new model to simulate and compare five different flooding protocols. We also test how PFS behaves under different situations by modifying the traces. Section 6 contains related work, and we conclude in Section 7.

2 Synthetic simulation results

In this section, we simulate the two flooding protocols CBB and PFS using a standard synthetic simulation model and compare the results with the experimental results obtained in our previous work [2]. The code is based on the same code as we used in [1] but with the protocol enhancements that was found in [2] also implemented in the simulator. Hence, we use the ns2 simulator [3], which is a frequently used tool for simulations of wireless protocols. However, we changed the wireless network layer to IEEE 802.15.4, which was used in our test-bed experiments. Hence, the application, the flooding protocol, and the MAC protocols are all the same in both the simulator and the test-bed. The simulator is set up to closely mimic the test-bed.

In the following two subsections, we first highlight the most important aspects of the simulators and the test-bed experiments used. In the third subsection, we show the results of the comparison and analyze the discrepancies.

2.1 The test-bed experiment

For our test-bed experiments, we used t-mote Sky (based on Telos Revision B) from Moteiv Corporation. The wireless technology used by t-mote Sky is 2.4 GHz IEEE 802.15.4, which uses direct sequence spread spectrum radio frequency modulation with a data rate of 250 kbps.

In our experiments, a packet consisted of a payload plus a total of 17 bytes of headers (the MPDU consisted of a 9-byte header and a 2-byte CRC field). When transmitting, a mote used the clear channel assessment (CCA) function defined in [14] to determine whether the channel is idle or not. If there was an ongoing transmission, the mote waited a random time (uniformly distributed between 61 μs and 2.0 ms in the first attempt and between 61 μs and 7.8 ms in the subsequent attempts) and tried again. After eight unsuccessful attempts, it gave up and dropped the packet. Since we used broadcasting, no acknowledgments were exchanged.

We placed 50 motes in a grid topology with ten motes lined up in five rows. We used six different deployment scenarios, where the distance between neighboring motes was the same but varied between 0.3 and 2.0 m for the different experiments. After 2.0 m, the network became disconnected with a very high probability. Each node was elevated about 0.2 m above the floor using blocks of polystyrene foam in order to avoid the worst kind of multi-path interference. To make the network multi-hop, we set the transmission power to almost the lowest level (−24 dBm).

We used the following definition as a measurement of the network density:

$$\text{Average node degree} = \frac{1}{N}\sum_{x=1}^{N}\sum_{y=1}^{N} R_{x,y}, \tag{1}$$

where N is the number of nodes, $R_{x,y}$ is the packet delivery ratio from node x to node y, and $R_{x,x} = 0$. It is basically the average number of direct neighbors but also considers the packet delivery ratio of the links. If a particular link experienced a packet delivery ratio of 40%, we counted that as a 0.4 link. We did this because a flooding protocol should be able to exploit the fact that packets can be sent on this link with a probability of 40%.

A higher value of Equation 1 means a denser network with 49 as the maximum in our 50-mote network. For each deployment scenario, we measured the average node degree by letting each mote transmit 100 packets with 2-byte payload. At the same time, each mote listened for packets from other motes so that the packet delivery ratio for all node pairs could be estimated. This was repeated five times before, during, and after each experiment, and an average was calculated.

To properly compare CBB and PFS, we first determined the parameters that worked best in all deployment scenarios. This was done for both of the flooding protocols. The resulting test-bed parameters are summarized in Table 1, while the details of this are given in [2].

To compare the flooding protocols, we use the following three measurements in the remaining of this article:

- *Reachability* This measurement evaluates a protocol's reliability. It is represented by the delivery ratio of a flooding message at all receiving nodes. For example, if there are 50 nodes in a network and a node floods a message using a certain flooding protocol resulting in 42 nodes receiving this message, then we say that the reachability is $42/49 = 85.71\%$.
- *Retransmission* This measures the number of retransmitting nodes for flooding a single message. Other messages, such as hello messages are ignored in this measurement. It measures the efficiency of the protocol.
- *Delay* The delay is the length of the time interval from the moment that the source node sends a flooding message until the moment when the last node in the network receives this message. If a message does not reach all nodes, the last node means the last node that received the message.

2.2 The simulator

For the simulations, we used ns2 version 2.34 and the standard two-ray ground reflection model, which is the default synthetic model used by ns2. The effect of this model is

Table 1 Parameters used in the comparisons

Protocol	Parameter	Value
Both	Number of nodes	50
Both	Total flooding packet size	42 bytes[a]
Both	Total hello packet size	19 bytes
Both	Number of floodings per scenario	50
CBB	$T_{\max}$	200 ms
CBB	Threshold	3
PFS	MAX_SLOTS	6
PFS	$tx\_\text{delay}(\|m\|) + D$	60 ms
PFS	EARLY_P	10.0
PFS	Hello protocol	Lenient
PFS	Hello packet interval	0.9 to 1.0 s

[a]The payload of the flooding packets was 23 bytes for CBB but only 15 bytes for PFS due to extra overhead.

that the network topology becomes a unit disc graph and that links are either perfect or non-existing among other things. There are simulators with better synthetic models, such as Castalia [15] and MiXiM [16]. However, starting with a very basic model, we will be able to investigate what real-world characteristics affect the performance of our flooding protocols.

To better model the MAC and PHY behavior in our test-bed, we configured the physical layer to mimic the 802.15.4 physical layer and implemented the relevant parts of the IEEE 802.15.4 MAC protocol. Since we only need to simulate broadcasting, we could ignore acknowledgements, retransmission, and most of the complexity of the full version of 802.15.4. Only the header formats and the CCA with backoff were implemented.

An important parameter in flooding is the delay between when a packet arrives at the interface and is processed by the flooding algorithm. If this delay is long, it may degrade the flooding performance as we have shown in our earlier works [1,2]. This is due to timing issues between when a decision to retransmit is taken and when received packets are processed, packets that may make the node refrain from retransmitting. If the processing is delayed, unnecessary retransmissions may happen. To model this behavior in the simulator, we used the delay parameter in the link layer module (LL) of ns2, which introduces a delay on both incoming and outgoing packets between the MAC level and the higher levels. A fixed constant was used that we changed to find the best result. The value we used in the simulations, unless mentioned otherwise, was 5 ms. See Section 4.1 for more details on how we arrived at that value. Note that for the delay measurement, we used the packer arrival time at the interface, i.e., before the 5-ms LL delay, since this better mimics the test-bed implementation.

In the simulator, we placed the nodes in the same grid topology as in the test-bed experiments. Also here, the distance between neighboring nodes was altered to achieve different network densities. We used the same metric (Equation 1) to measure the network density in the simulator so that we can compare with the results from the measurements.

Due to the perfect simulation environment created by the two-ray ground reflection model (it becomes a unit disc graph), we experienced some strange border effects. To avoid this and also introduce some randomness into the simulations, we also tested random topologies. In the random topology, the nodes were uniformly distributed in a rectangular area corresponding to the grid topology in the test-bed. We used a rectangular area with various sizes but always with the same ratio of 4:9 between the sides. We refer to the former simulator topology as *Grid* and the latter as *Random*.

2.3 Comparison

Figure 1 shows the resulting graphs. In these, and the following graphs, each flooding measurement from the test-bed was repeated 50 times from random nodes. In

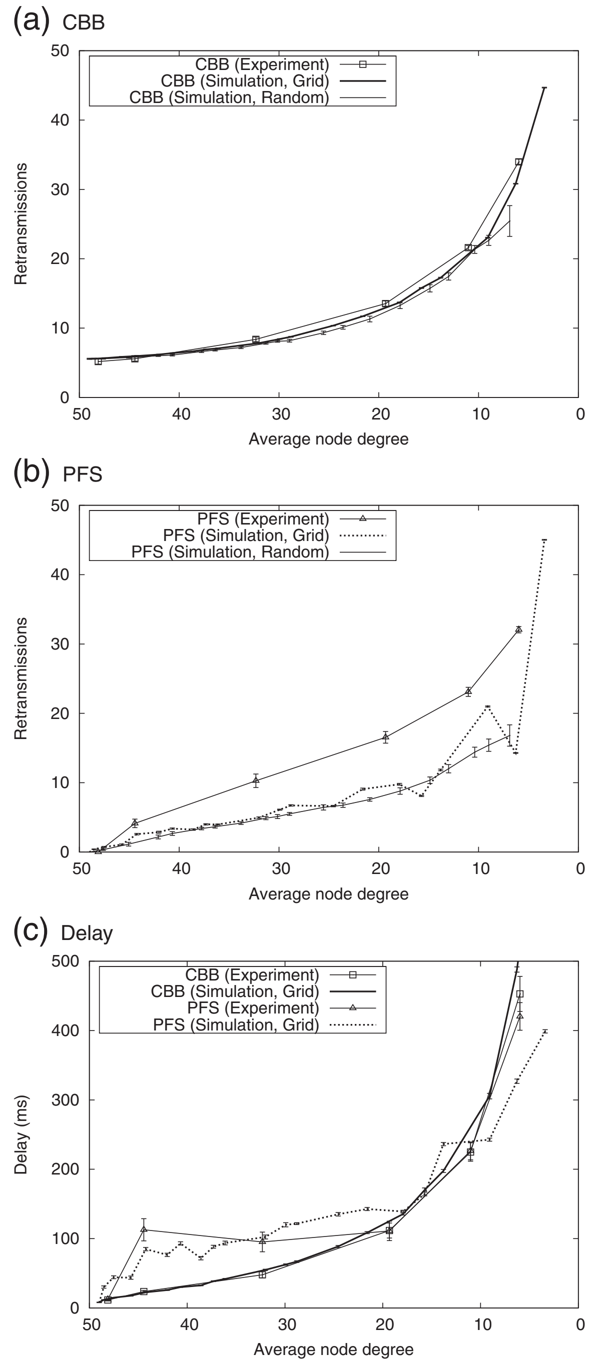

Figure 1 Experiment and ns2 simulation comparison. This figure shows a comparison between a standard ns2 simulation and the test-bed experiments. (**a**) CBB retransmissions, (**b**) PFS retransmissions, and (**c**) end-to-end delay for both CBB and PFS. Note how the results are similar for CBB but different for PFS.

the resulting graphs, we show both the average value and the 90% confidence interval. In the simulator experiments, 200 flooding messages were generated from random nodes in each simulation, and each simulation was repeated 20 times. In the graphs, we show the average and the 99% confidence interval. In both the simulations and the test-bed measurements, flooding and hello packets were separated in time. Also, the time between floodings was big enough to avoid one affecting the next.

Figure 1a shows the results for CBB. In terms of retransmissions, we can see that the grid topology simulation corresponds almost perfectly with the experimental results despite its very simple model. Further, switching from a grid topology to a random node topology does not significantly affect the results. The same holds for reachability (not shown), which is very high in all three cases. Concerning the delay, we can see in Figure 1c that a similar curve is obtained as in the experiments. Hence, our simple simulator model produces very accurate results for CBB.

In Figure 1b, the same comparison is shown for PFS. Unfortunately, we cannot see a good correlation between the simulation results and the experimental results as we did for CBB. It seems that the simulator is too optimistic concerning the retransmissions of PFS when the network becomes sparser. There can be a number of reasons for this, such as the unrealistic assumptions made by common synthetic models that we listed in Section 1. The two-ray ground reflection model of ns2 makes all of those assumptions. Exactly which assumptions cause these discrepancies is unclear from this experiment. However, in Section 5.1, we will return to this question.

The delay results for PFS are also shown in Figure 1c. In this case, the simulations are fairly close to the observations from the test-bed.

The conclusion of this work must be that we need to find a better simulation model that better reflects the real situations that we experience, especially for the number of retransmissions. In this case, the optimistic simulation results for PFS even lead to the wrong conclusion that PFS is better if we only rely on the simulation results. Hence, we have once again demonstrated the shortcomings of relying on too simple simulation models without looking at the accuracy. On the positive side, we can also note that our simulations were much quicker than the test-bed measurements. We could achieve over 20 floodings per second per processor core (Intel Core i7-2677M, Intel, Santa Clara, CA, USA) in the simulator for PFS. In the test-bed, one single flooding took 20 s to complete. Therefore, it is still worth to continue and try to improve simulator accuracy. To do this, we will first identify a better simulation model and then again verify the simulations against test-bed measurements.

3 A trace-based simulation model

In this section, we will explore the possibility of using our network density measurements from the test-bed in the simulator. Hopefully, that will give more realistic results.

The traces that was collected is solely on the packet level, which means that our trace-based model will be somewhat different from traditional models, such as log-distance and log-normal radio propagation models based on SINR and radio sensitivities. Instead, our model will solely be on what the MAC and higher layers experience, i.e., the reception or non-reception of packets.

3.1 Using traces

The available trace data consist of the total number of received packets per link and per direction, translated into a percentage (0% to 100%). Figure 2 demonstrates, at high level, this information for one experiment (from a 1.0-m deployment scenario). Each row represents a transmitting node, and each column represents a receiving node. When a cell is dark gray, that link has a 90% or higher delivery ratio. A white cell means less than 10%, and the gray cells are everything in between. It is interesting to notice that only about 10% of the links have a delivery ratio between 10% and 90%.

This information can be used in the radio propagation model. We load the topology information from a given test-bed experiment into the simulator. When node i sends a packet, we decide whether node j can receive the packet or not by drawing a random value uniformly between 0 and 99 and compare it to the delivery ratio in row i, column j of the matrix. If the random value is lower than the delivery ratio, the packet is correctly received; otherwise, it is corrupt. In the simplest model, all the reception events are independent of each other, even the ones from the same transmission.

In ns2, we implemented this by creating a new ErrorModel. We kept the rest of the simulation model the same to keep as many factors the same as possible in the simulator. We even kept the two-ray ground reflection model, but disabled its main function by placing all nodes sufficiently close to each other so that they were within each other's coverage area. This meant that packet errors were only created by the error model and that all nodes were within each other's interference and carrier sensing range. Hence, no hidden terminal problems would be present in the simulator, for instance. However, it is not likely that there were any significant hidden terminal problems in the test-bed either, since all our measurements were done in a confined area where all nodes would be able to sense each others' transmissions but obviously not always correctly receiving all transmissions.

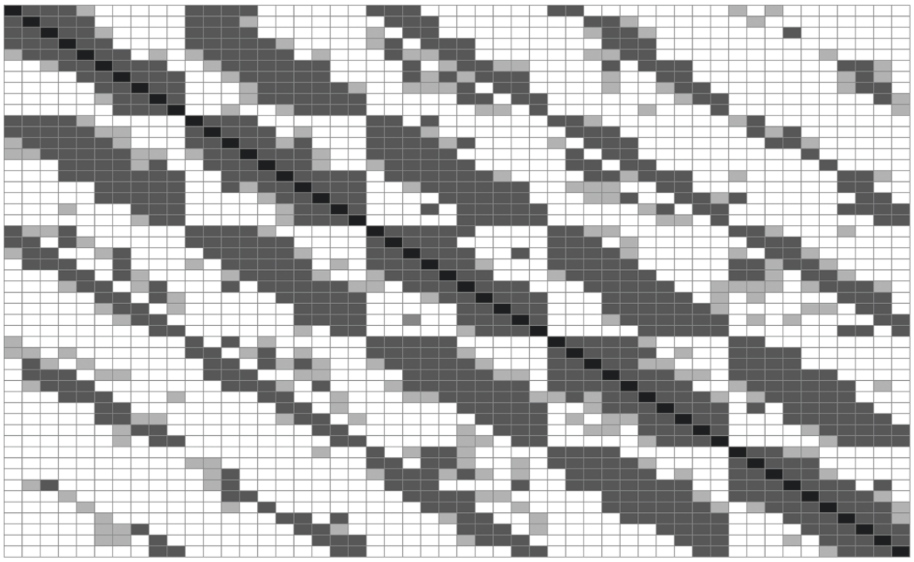

Figure 2 Trace-based topology information. Each square represents one link. A dark gray square represents a 90% to 100% packet delivery ratio, a white less than 10%, and a gray somewhere in between.

3.2 Aspects affecting the simulation results

There are a number of aspects that may influence the performance of the protocols and therefore needs some attention. The following three aspects are different between the trace-based model and the two-way ground reflection model, and may have significant impact on the results:

- *Links without in-between qualities* In the two-ray ground reflection model, links are binary, i.e., either perfect or non-existing, but never in between. In the trace-based model, the links can have in-between qualities, which is reflected in that the link qualities are expressed as a percentage. To make a trace into binary, we can replace the delivery ratios with either 0% or 100% based on whether that link has a higher or lower packet delivery ratio than a certain threshold. To maintain the network sparseness, the threshold should be set in such a way that the average node degree (based on Equation 1) remains as unchanged as possible.
- *Symmetric links* In the trace-based model, the link quality in the different directions may be different. To make the trace-based model symmetric, we can take the traces and set the delivery ratio in both directions to the same, namely to the average of the delivery ratios in the two directions.
- *Topology* If the links of the trace-based model are made both binary and symmetric, then the main difference between the trace-based model and the two-way ground reflection model would be the topology. The two-ray ground reflection model would have a unit disc graph topology, while the topology of the trace-based model is much more random, with the existence of some long links and some short links missing. Hence, the topology is different, and this is also expected to have an effect on the performance.

The different protocols may have difficulties in handling some of these aspects depending on whether the aspects are present or not. In Section 4.3 and also Section 5.1, we will study these aspects, by modifying the trace data so that the model becomes binary and/or symmetric.

In addition to these three aspects, there are two more aspects that we need to give some attention. Aspects that are also not included in the basic trace-based model, namely short-term effects and the effects of different packet sizes. Using a synthetic approach, they can be included in the trace-based model as well. The two are discussed in the following two subsections, and their impact on the simulation results are studied in Section 4.2.

By studying this, we will know what aspects are important to model in a simulator in order to get good simulation results. Furthermore, we can tell how the different protocols respond to the different aspects and predict how they will perform in networks where these aspects are present or not.

3.3 Short-term packet error behavior

Our trace information tells us how each of the links behave in the long term but does not tell us anything about the short-term behavior. We know that if one packet is lost, the probability that a consecutive packet transmitted shortly afterward also will be lost is higher, i.e., a wireless link does not have the memoryless property. A simple way to model this is to use a two-state continuous Markov model, i.e., to assume that a link has two states: a good state and a bad state. Packets that start their transmission during a good state are delivered, while packets starting their transmission during a bad state are lost. Each link has their own state machine and changes between the states over time. The time the link spends in one state before changing to the other is exponential, and the ratio between the two waiting times is determined by the long-term packet delivery ratio of the link.

The average cycle time, the time between change from one state to the other and then back again, would determine how often the link transit between the states per second. In Figure 3, two links that have the same long-term delivery ratio (50%) are shown. However, the number of transitions per seconds is double in the second link, which means that in the short-term, the two links behave quite differently. To capture this difference, we need one more parameter, which we call c. We define c as the average cycle time and is measured in milliseconds. A smaller c means more state transitions per second. When c becomes really small, the link will start to appear as memoryless, and we can ignore this model and just use independent random packet losses again. When we do that, we will say that $c = 0$.

To make the model even more accurate, we could also introduce packet loss correlation between nearby links and/or nodes. For instance, an interferer would affect any link going to the same node and also links going to nearby nodes. However, as we will show later in this article, it is not necessary to take this much details into consideration when defining our simulation model. Therefore, we will not attempt to make our model take this into account as it would also make the model unnecessarily complex.

3.4 Packet size effects

Another aspect we could introduce is the fact that longer packets are more prone to packet errors than shorter packets. There are simply more bits in the packets that can be erroneously received, and this effect can sometimes be substantial. Unfortunately, we did not collect this information from the test-bed. The packet size used in the network density measurements was the same as the hello packets used by PFS (19 bytes including headers). To find the actual packet loss for the flooding packets (42 bytes including headers), we need to model this and extrapolate an estimate for the actual packet loss for flooding packets.

To do this, we assume that packets are correctly received by a probability expressed as $p_L = p_0 p^L$, where L is the packet length in bits, $1 - p$ can be considered the bit error rate (BER), and p_0 is a non-size-dependent delivery probability component, such as the receiver failing to synchronize to the preamble and collisions. Note that p and p_0 can be uncorrelated with each other. This model is in line with common ways used in wireless communication research when translating between BER and packet error rate (PER) (e.g., [17, p. 215]).

To be able to use our collected trace data, which only contains delivery rates for one packet size, we have to assume that a good p always correlates with a good p_0 and often this is the case. However, for future studies, we would recommend to measure the network topology using multiple packet sizes as that could lead to better accuracy. We assume that $p_0 = p^a$, which means that we can calculate the packet delivery ratio for flooding packets, based on the delivery ratio of hello packets by the following formula:

$$p_{\text{flooding}} = (p_{\text{hello}})^{\frac{1+a|\text{flooding}|}{1+a|\text{hello}|}}$$

Note that this correction cannot be combined with our link model when $c > 0$.

4 Trace-based simulation parameterization and accuracy

We again simulated the six deployment scenarios (with the varying network densities) but with our improved trace-based propagation model and compared with the test-bed measurements. Before comparing the results, we need to find the parameters of our model that give the

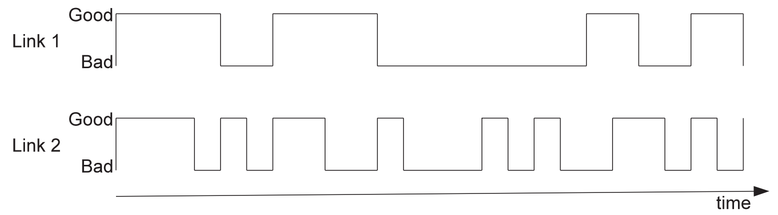

Figure 3 Two links with the same delivery ratio. Shows two different links with the same packet delivery ratio, but the top one has longer periods of constant packet losses and longer periods of constant packet delivery compared to the bottom one.

best accuracy. This includes the LL delay parameter, the *c* parameter, and the *a* parameter. For clarity, we only show three representative deployment scenarios for this tuning, namely the fully connected scenario, the 0.6-m scenario, and the 2.0-m scenario.

4.1 LL delay

We start by looking at the LL delay parameter and its effect on the simulation results. Figure 4 shows different LL delay parameters for both CBB and PFS. The horizontal lines show the values from the measurements and those values are what we are aiming to mimic. We can clearly see that different LL delay values are best for different scenarios and measurements. Reachability (Figure 4a,d) shows almost no influence at all from the LL delay parameter. In the case of retransmissions (Figure 4b,e) and end-to-end delay (Figure 4c,f), a higher LL delay means a higher value, and this is to be expected as we explained in Section 2.2.

For CBB, an LL delay parameter around 4 to 6 ms seems right in all deployment scenarios except for retransmissions in the 2.0-m scenario, where a value of about 16 ms seems better. The retransmissions of PFS seem to be simulated more correctly when the LL delay parameter is higher. However, we need to find the LL delay parameter that best approximates all scenarios and measurements. To do this, we study the simulation errors with respect to our real network observations. We choose to use the squared errors and therefore make use of the following error formula:

$$\sum_{p,s}(y_{p,s} - \tilde{y}_{ll,p,s})^2/y_{p,s}, \tag{2}$$

where $y_{p,s}$ is the measurement from the test-bed scenario *s* and protocol *p*. $\tilde{y}_{ll,p,s}$ is the corresponding simulation result using simulation parameter *ll* (LL delay). For each measurement type (reachability, retransmissions, delay), we calculated the error for different *ll* to find which one produces the smallest error. Figure 5 shows the resulting graph. Note that the different curves cannot be compared with each other despite that we plot them in the same graph. They refer to different measurements and have different scales.

We can see that for reachability, the LL delay value has very little influence and no clear trend. Hence, we can safely ignore the reachability when deciding the LL delay. For the other two measurements, we clearly see that 5 ms is the best for delay and 6 ms is best for retransmissions. It should therefore be perfectly alright to select a LL delay of 5 ms. This is also in line of what can be expected from the actual processing delay of our sensor motes.

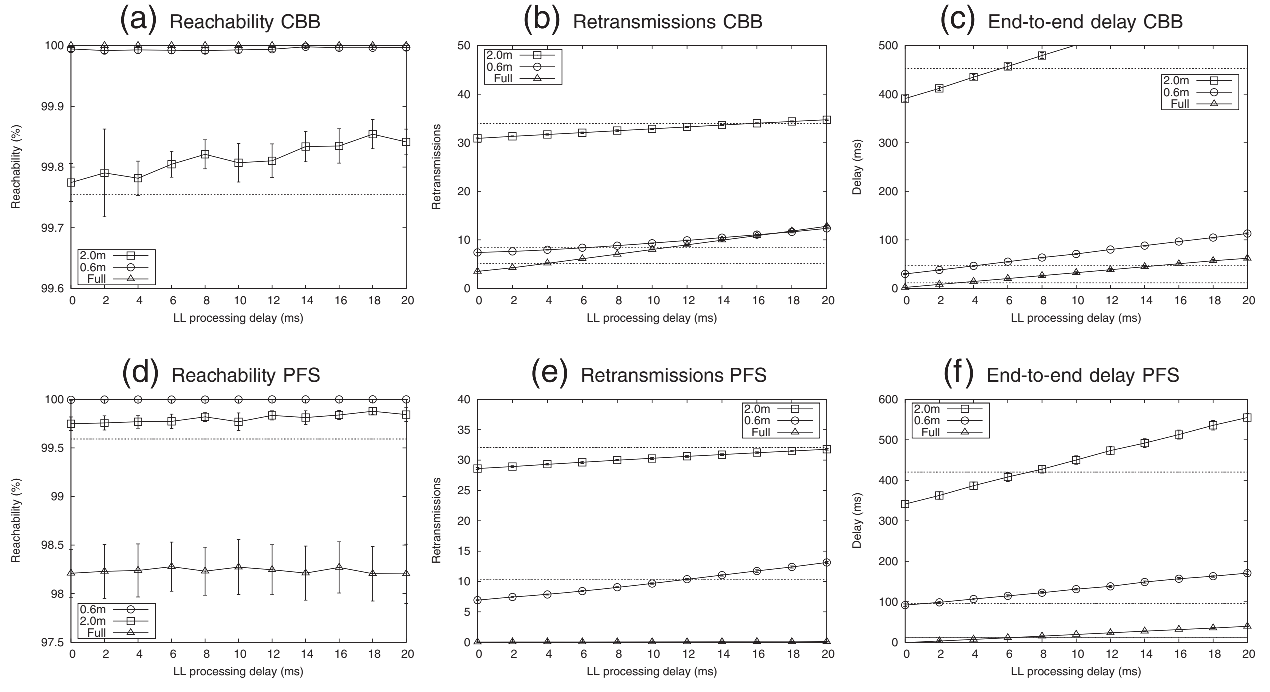

Figure 4 Different LL delays for CBB and PFS. The graphs show how LL delay affects the different measurements and protocols. (**a**) Reachability CBB, (**b**) retransmissions CBB, (**c**) end-to-end delay CBB, (**d**) reachability PFS, (**e**) retransmissions PFS, and (**f**) end-to-end delay PFS.

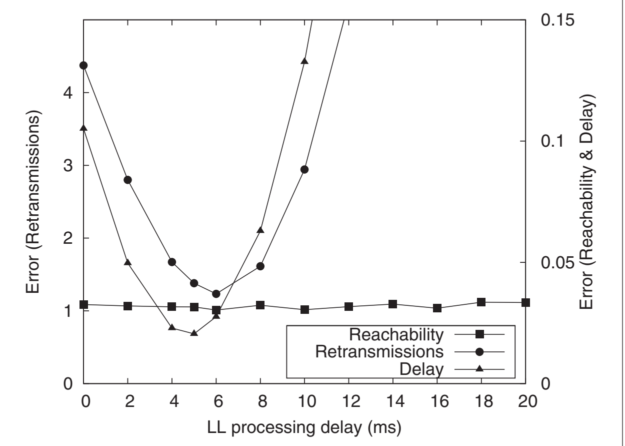

Figure 5 Errors for each measurement and different LL delays. Shows the error of each measurement for different LL delay values. Note that you cannot compare the different graphs with each other as they represent different measures with different scales. A value between 5 to 6 ms seems the best fit.

4.2 Parameter c and a

Let us now turn our attention to the c parameter that regulates the short-term packet loss effects. The c parameter has almost no effect on CBB results (results not shown), which is to be expected as CBB does not keep track of links anyway. For PFS, which uses neighborhood information, we do see different results, which are shown in Figure 6a,b,c. The straight horizontal lines are what we measured in the test-bed.

In some deployment scenarios and with some measurements, a bigger c improves the accuracy, while in others, it decreases. To better see if there is any real improvements, we again calculated the errors in the same way as we did for the LL delay parameter. The results are shown in Figure 6d for PFS. As before, it is not possible to compare the different curves with each other as they have different scales. From the results, we can see that only for reachability, the error goes down if $c > 0$. Hence, it is not clear that this is an enhancement. However, this is actually good news since we can work with a simpler model.

The reason that the links appear as memoryless in our model may be due to the fairly large intervals between transmissions in our test-bed experiments. In other experiments, where packets are sent closer to each other or even back-to-back, the results could be different. However, for flooding protocols, the links can just be assumed to behave as memoryless.

We did a similar simulation for the packet size effect and the a parameter. We varied a between 0 (meaning no packet size compensation) and 0.035, which is slightly higher than what could be expected for this network. The resulting error graph for PFS is shown in Figure 7.

The results from the packet size effect is similar to that of the short-term packet loss effects. We can see that retransmissions get an improved accuracy as we increase a, but other measurements are unaffected or degrading. Also for CBB, the errors point in different directions when we vary c. Hence, it is not clear that using the packet size effect and $a > 0$ is an improvement and if there is an improvement, it is going to be minor.

We can conclude that neither $c > 0$ nor $a > 0$ yield any improvement, allowing us to stick to the original and simpler model. For the packet size effect, perhaps it is still possible to find an improvement if we had access to the actual packet loss of large flooding packets instead of deducing it from the small packet loss using a synthetic approach.

4.3 Validation of the trace-based model

To validate the trace-based propagation model, we will now use the values we found in Section 4.1 and Section 4.2. That is, an LL delay of 5 ms, $c = 0$, and $a = 0$. We again repeat the simulations done in Section 2.3 but with our new model. The new and improved simulation results are shown in Figure 8 for PFS and again compared with the test-bed measurements. The binary model is the one we introduced in Section 3.2. We will discuss those results later in Section 5.1. For CBB, very little change can be noted in the same way as shown before in Figure 1a, and therefore, we did not include those graphs. On the other hand, we can clearly see the improvements for PFS brought by the trace-based propagation model, especially in the retransmissions (Figure 8b). Note that the two-ray ground reflection model simulation appears more erratic mainly due to the extra number of results.

In Table 2, we present the errors as numbers between the two simulation models and the test-bed. For each protocol, deployment, and measurement type, we calculated the absolute error. In Table 2, we show the average of those values and compare between the two-ray ground model and the trace-based model.

To further validate the new model, we also included counter-based PFS (CB-PFS) in the comparison. CB-PFS is a combination of CBB and PFS that we introduced in [2]. CB-PFS works exactly as PFS but with a counter like CBB. A node using CB-PFS can refrain from retransmitting either due to self-pruning according to PFS or due to the counter exceeding the threshold according to CBB. In Section 5.2, we show how CB-PFS perform in comparison with the other protocols.

From Table 2, we can conclude that the trace-based model reduces the simulation errors in most cases. Quite often by more than 50%. The only exception is reachability for PFS and CB-PFS, where the errors are small to start with. The increase in error is mainly attributed to the results in the most dense deployments as can be seen in

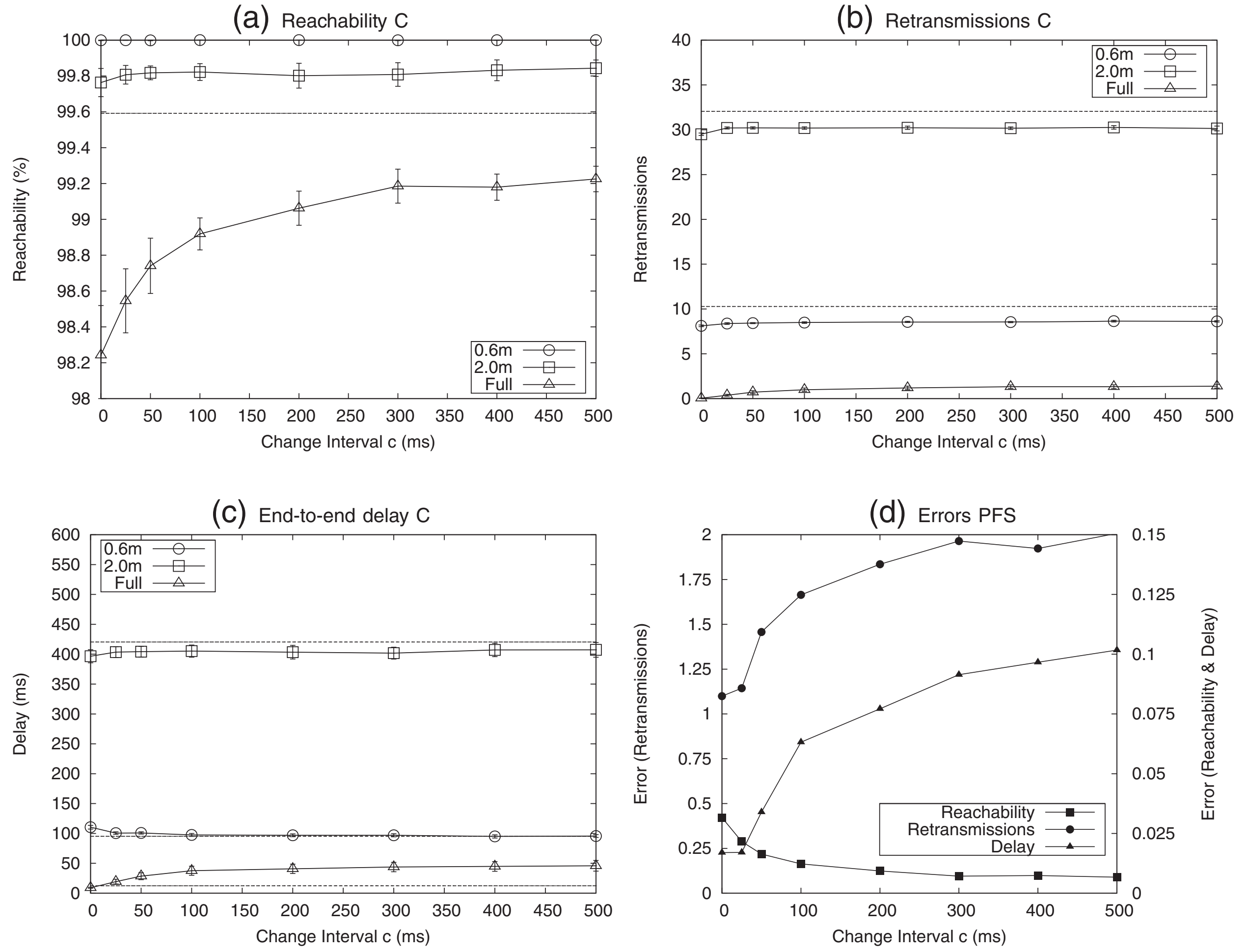

Figure 6 Tuning the *c* parameter for PFS. (**a, b, c**) Shows how PFS is affected by different *c* values, all three measurements are shown. (**d**) The last graphs shows the mean absolute errors of all scenarios and for different *c* values.

Figure 8a, where the trace-based simulation significantly drops in reachability. In theory and after analyzing the workings of PFS and CB-PFS (also see Section 5.2), the simulator results appear to be correct. Why this is not observed in the test-bed is not clear. On the other hand, the two-ray ground reflection model has too good reachability (always more than 99.975%), which clearly also is wrong, just closer to the test-bed results. A final explanation could be the limited number of measurements in the test-bed and the rarity of the event when a node does not receive a flooding message.

To quantify the model errors, we calculated the CDF of the retransmission, reachability, and delay errors and presented them in Figure 9. To be able to plot it in the same graph and to be able to compare the errors to each others, we chose to use the relative error in this graph. Using this graph, we are able to estimate how far away from the actual values the trace-based simulation results are. For instance, we could say that with a probability of around 75%, reachability results are within ±0.2%, retransmission results are within ±10%, and delay results are within around ±20%. The same numbers for the two-ray ground reflection model are ±0.1%, ±40%, and ±30%, respectively. The delay results have a relatively high uncertainty in high-density topologies where different forwarding paths over a few hops have a large impact on the relative error.

Given the variation of considered flooding protocols and topologies, we expect the presented numbers to be generalizable for comparable scenarios. However, similar validation has to be done for scenarios not involving flooding protocols or different types of network deployments.

5 Using the trace-based model

Now that we have improved the accuracy of our simulator, it makes sense to use the model and, for instance, simulate other flooding protocols for performance comparison.

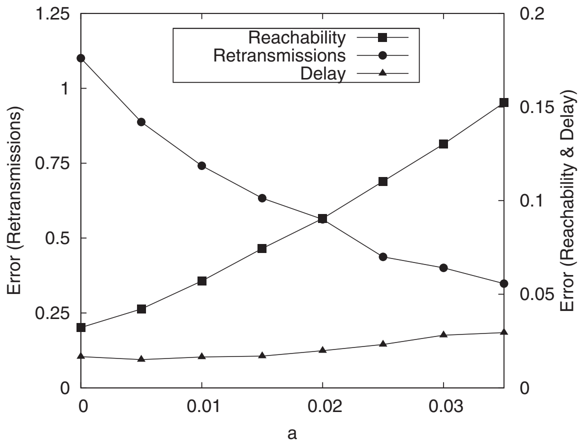

Figure 7 Tuning the *a* parameter for PFS. The graph shows the mean absolute errors for PFS for all measurements and scenarios when changing the *a* parameter.

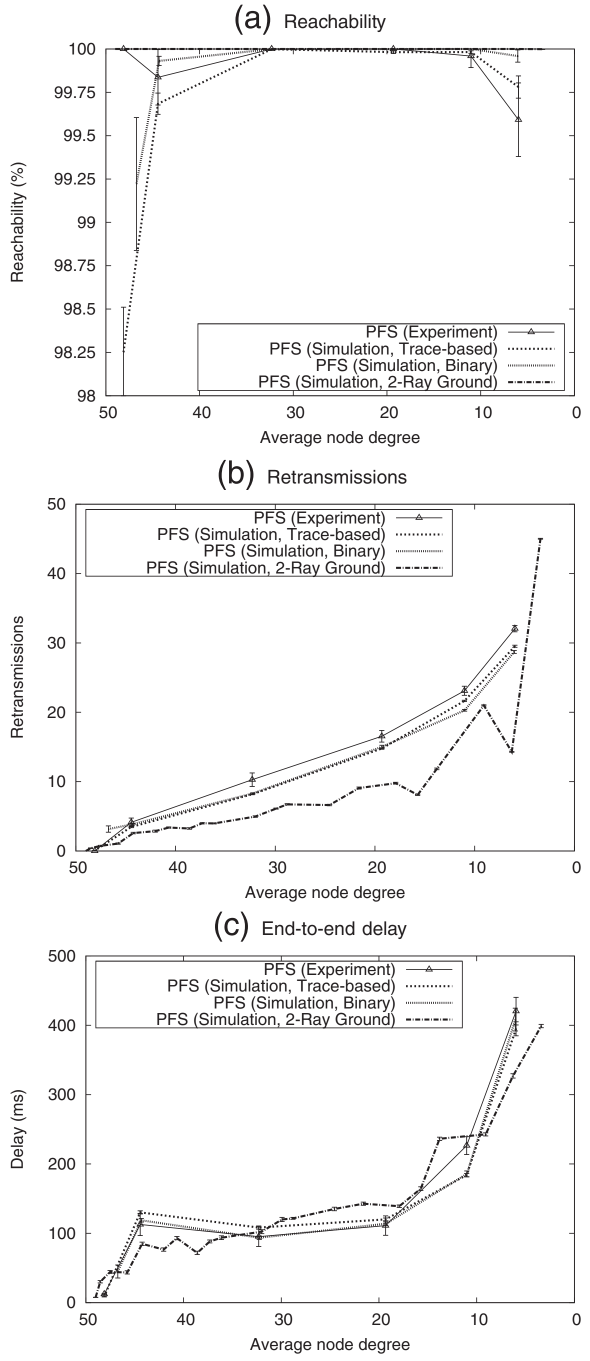

Figure 8 Trace-based simulation of PFS. Results from the PFS simulation using the different simulation models and compared with the test-bed results. Each graph shows a different measurement, (**a**) reachability , (**b**) retransmissions, and (**c**) end-to-end delay.

With the more accurate simulation model, we should get results close to a real network, and our conclusions should become more reliable compared to earlier simulation attempts. Since we have the error CDFs, we can use them to find the confidence intervals of our results and hence would know when statistically certain comparisons can be made.

In the remaining of this section, we will first look at how PFS performs under different assumptions. Then, we will extend the flooding performance study with two more flooding protocols.

5.1 PFS protocol performance when modifying the traces

In this first use of the trace-based model, we are actually going to validate PFS itself and see how it behaves under different assumptions. That is, what real-world effects influence the performance of PFS. To do that, we modified the traces as mentioned in Section 3.2. The results are shown in Figure 8 when we made all links binary (the curves marked Binary). We can see that the binary aspect does not affect the results very much. Also, when we make the links symmetric or both symmetric and binary (neither shown), we cannot see a big difference in performance.

The conclusions from this must be that PFS is able to handle in-between links very good and that the main influencing factor compared to the original ns2 simulations actually is the topology of the network instead.

5.2 AHBP and SBA

In this subsection, we choose to simulate the other two flooding protocols that we tried in [1] but did not implement in the test-bed, namely, scalable broadcasting algorithm (SBA) and *ad hoc* broadcast protocol (AHBP). We will compare them with CBB, PFS, and CB-PFS.

SBA was proposed by Wei Peng and Xi-Cheng Wu in [18]. This protocol is similar to PFS but requires two-hop hello messages and has designed the RAD differently.

Table 2 Mean absolute errors for different simulation methods

	CBB			PFS			CB-PFS		
	Reach	**Retran**	**Delay**	**Reach**	**Retran**	**Delay**	**Reach**	**Retran**	**Delay**
Two-ray ground	0.057	1.4	12	0.13	6.3	33	0.18	4.6	39
Trace-based	0.014	0.74	4.3	0.36	1.4	19	0.41	1.1	19

In this table, we show the errors introduced by the original two-ray ground propagation model and the trace-based model as compared to the experimental measurements. Reachability is presented in percent and delay in milliseconds.

The RAD is a uniformly distributed delay between 0 and a function of the highest node degree of its neighbors divided by its own node degree. This means that nodes with more neighbors are more likely to retransmit faster, and this should make the neighbor elimination more efficient.

AHBP [19] also requires two-hop hello messages. Unlike PFS and SBA, it is the sender that decides which of its neighbors should retransmit a flooding message. This decision is based on the two-hop neighbor information that the node has. The selected neighbors are called broadcast relay gateways (BRGs) and are listed in the header of the flooding message. The BRGs are selected in such a way that if they retransmit the flooding message, they will together cover all two-hop neighbors of the sender. This should guarantee that all connected nodes in the network will receive the flooding message as long as the two-hop neighbor information is accurate. To decide the BRGs, a greedy algorithm is used that tries to minimize the number of BRGs but still cover all two-hop neighbors.

Previous research [19,20] has shown that AHBP performs well in static networks. However, when mobility increases in the network, the two-hop neighbor information becomes inaccurate, the wrong BRGs are selected, and reachability decreases. To better cope with outdated neighbor information, the authors propose an extension to AHBP. This extension tells a node that receives a flooding message from an unknown neighbor to assign itself as a BRG and retransmit this message. This extension will somewhat increase the retransmissions, but more importantly, it will increase the reachability, especially in the case of mobility, and is therefore an important extension to AHBP since AHBP tends to reduce the number of retransmitting nodes a bit too much. We therefore used this extension in our AHBP simulations.

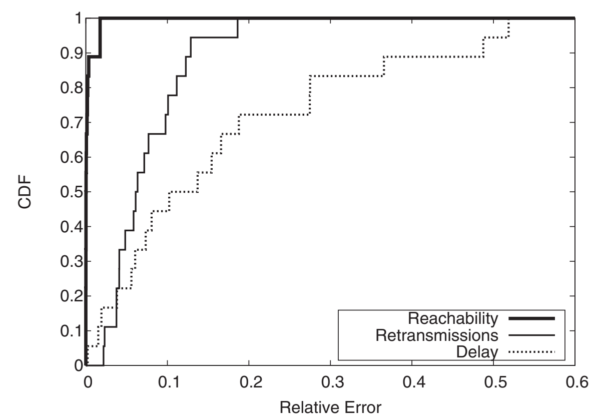

Figure 9 CDF of the relative errors of the trace-based model. Shows the validation results from the simulation using the trace-based model for all the different flooding protocols. For each of the measurements, we show the CDF of the relative errors as compared to the experimental results. The results include all 18 validation points (six deployments and three protocols).

Before comparing SBA and AHBP with the other protocols, we needed to tune their parameters. For SBA, we needed to set a constant *C*, which is a scalar constant multiplied with the slot length. We did this by trying different values and finding a good trade-off where a longer RAD does not decrease retransmissions much further. We found $C = 150$ ms to be a good value. Making it bigger would only make the end-to-end delay bigger without any significant improvements in reachability or retransmissions. For AHBP, only the maximum jitter was needed to be set. However, no impact on the results could be seen. We used 10 ms.

The simulation results of SBA and AHBP are shown in Figure 10 and compared with CBB, PFS, and CB-PFS for all three measurements. The error bars indicate the 99% confidence interval of the errors due to insufficient number of simulations added with the simulation model errors, as shown in Figure 9. For the model errors, we used the 50% values from the CDFs to not clutter the graphs too much. Hence, the bars should indicate the 50% confidence interval of the mean value. This might not look very impressive, but we need to remember that the performance of the protocols are quite similar to start with and the mean values are now inline with the test-bed conclusions. Furthermore, most network simulations are not able to give any model confidence at all, leading to incorrect conclusions.

From Figure 10a, we see that AHBP underperforms compared to the other protocols when it comes to reachability, except for in the really dense networks. One reason for this is AHBP's sensitivity to links with in-between quality. Another, equally important reason is AHBP's problem in dealing with asymmetric links. This was

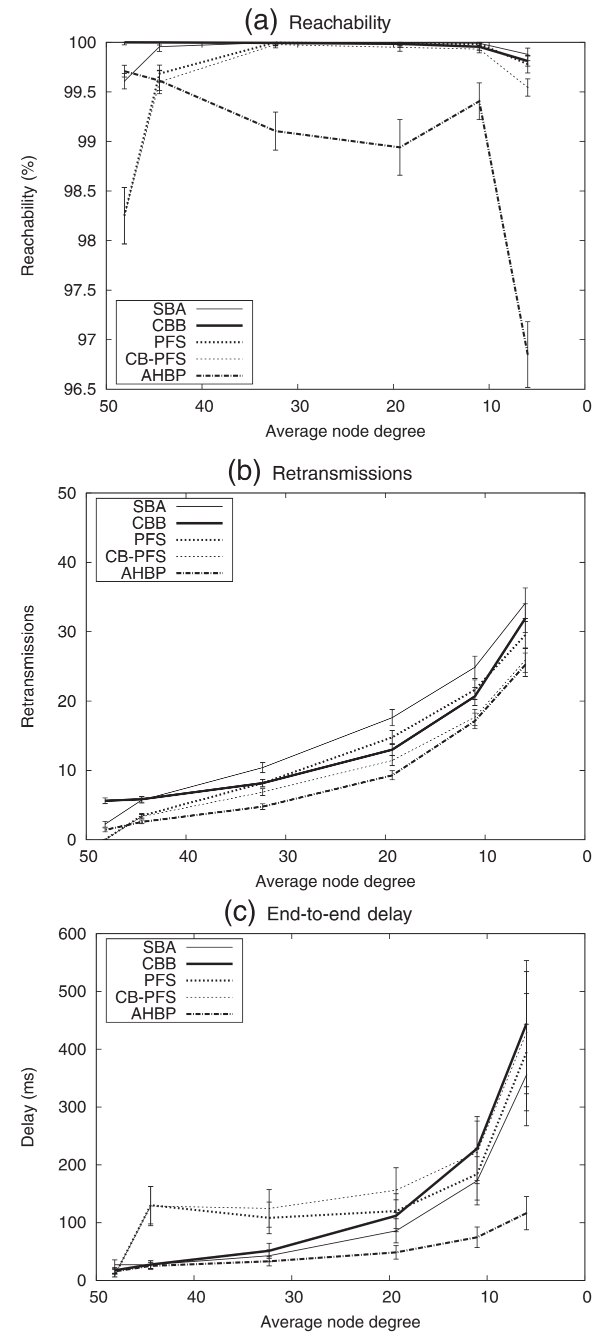

Figure 10 Comparison using the trace-based model. Shows the results from the simulation using the trace-based model for all the different flooding protocols. Each graph shows a different measurement, (**a**) reachability, (**b**) retransmissions, and (**c**) end-to-end delay.

verified by modifying the traces as we did in Section 5.1. When we made the links symmetric, we could see a significant increase in the reachability (not shown). Nearly half of the packet losses can be explained by this. When we make the links both binary and symmetric, then AHBP got a reachability near perfect (not shown). Only the rare but possible occurrence of packet collisions causes a little bit lower reachability for AHBP.

When we look at retransmissions, which is shown in Figure 10b, we find that AHBP performs the best in most cases or similar to CB-PFS in the most sparse networks. This explains why AHBP has problem with links with in-between quality and asymmetric links. It has very little redundancy and suffers from any error in the neighborhood information, while PFS and SBA both have enough redundancy to make up for such errors. It must also be said that AHBP has a lower number of retransmissions (and less delay) also due to its lower reachability, i.e., AHBP gets some of its good values in Figure 10b,c due to its lower reachability.

All neighbor knowledge-based protocols have problems with the reachability in the most dense deployment scenario, which may seem counterintuitive. However, this is due to the fact that the scenario is nearly fully connected, but not fully connected. This is especially true for PFS and CB-PFS, since they use a different hello protocol where links are said to exist if at least one of the last three hello messages are received. This fools the pruning algorithm to believe the network is fully connected and no retransmissions needed. We kept the standard hello protocol (the reception of the last hello packet determines the existence of the link) for AHBP and SBA, and that is why they have higher reachability than PFS. They are less likely to believe the network is fully connected.

In general, we can conclude that all protocols have very good reachability, except AHBP in the sparse deployment scenario and PFS/CB-PFS in the nearly fully connected scenario. On the other hand, AHBP has both the lowest retransmissions and end-to-end delay. CB-PFS has very few retransmissions but still very high reachability. The strength of SBA is its low end-to-end delay while still maintaining a good reachability. Finally, CBB has the best reachability and still reasonable retransmissions and end-to-end delay. Hence, different protocols have different strengths, and the best choice will depend on the application requirements.

6 Related work

Comparison of flooding protocols have been done before. An early study was done by Williams et al. [20], which was purely based on ns2 simulations using the standard two-ray ground reflection model. We conducted a similar study in the paper where we introduced PFS [1]. Others have used analytical modeling to study and understand the behavior of some flooding protocols. For example, Shah-Mansouri et al. [21] modeled CBB. A handful of test-bed measurement studies of flooding protocols have also been done. Most of them measured blind flooding or a variant thereof. Examples include [22-24].

Trace-based simulations have also been proposed and used by others. One of the first to use traces in wireless simulation was Nguyen et al. [11]. They collected traces between two hosts using WaveLAN and used in a trace-based simulation. However, the trace-based simulation was used as the reference to validate some synthetic radio propagation models. Only a single link was investigated.

A more recent study by Alan Marchiori et al. [12] was done on networking aspects of WSNs. They collected traces using USB-connected WSN motes and implemented their own WSN simulator on top of SimPy. They compared simulations based directly on the traces with synthetic radio propagation models based on the collected SINR measurements from the same traces. Finally, they studied the performance of the collection tree protocol, a simple and popular data collection protocol for WSNs, and compared with test-bed measurements. Their findings are inline with our findings. Only one network protocol was studied, and comparisons with the more common two-ray ground reflection model are missing.

Kotz et al. [9] evaluated common MANET simulation simplifications and their effect on routing protocol performance. Their work is similar to ours in the sense that they tried to quantify the loss in accuracy caused by non-realistic assumptions, such as symmetric links, links without in-between qualities, unit-disc transmission range, etc. However, they only used synthetic radio propagation models.

Another paper with similarities to ours is by Pham et al. [10]. They also compared simulators and test-bed measurements for a WSN broadcasting protocol. However, they tried to find a synthetic model to use in the simulator, which was not entirely successful according to the authors.

Also, Halkes and Langendoen [25] did a work similar to ours by comparing WSN protocols in a real test-bed with trace-based simulations. However, their work focuses on the MAC protocol and therefore is more about high intensity traffic, with lots of collisions. They test two ways of doing trace-based simulations. The first one uses packet traces but converts them into a binary reception model by removing all links without near perfect reception. As can be understood from this paper and observed in their paper, this leads to accuracy problems. The second one is based on SNR traces and performs better, but this is mainly due to better modeling of collisions, interference, carrier sense, etc. Effects that we have very little of in our setup.

The main drawback of any trace-based approach is that you limit your study to the networks that you have at hand. It is hard to generalize the results to other deployments. However, a greater amount of test-bed traces are being collected and published by others (e.g., [26,27]). Given that the right information is collected, we could continue this work using traces from different deployments. It is also possible to collect traces from the actual deployment of interest and use in off-line simulations similar to this study to determine the best protocols and parameters of that particular deployment.

7 Conclusions

In this article, we compared the performance of flooding protocols for wireless multi-hop networks, in particular, MANETs and WSNs. In earlier works, we first simulated in the standard simulation package ns2 and then tested three of the protocols in a real wireless test-bed. We noted discrepancies not only in the measurement results but also in the conclusions from the studies and wanted to investigate how these discrepancies could be avoided. To achieve this, we used a trace-based simulation model, which have the flexibility and ease of the simulator, but hopefully the accuracy for correct performance comparisons. In this article, we studied different approaches on how to use collected trace data and quantified the accuracy of the achieved model. We showed that accuracy for our study of flooding protocols could be improved by fairly simple simulation models. Finally, we compared the same protocols as we did in our first simulations and demonstrated new insights into the performance of the different protocols. By modifying the trace data, we could also study what real world effects influence the protocols the most and found that the network topology was the biggest contributor to the errors of the original two-ray ground propagation model.

Competing interests
Both authors declare that they have no competing interests.

Acknowledgments
This work is based on results funded in part by the Commission of the European Union under the project IST MAGNET, by the Dutch Ministry of Economical Affairs under the Freeband PNP2008 project, and by the Swedish Foundation for Strategic Research (SSF) under the project ProFuN: A Programming Platform for Future Wireless Sensor Networks. We would also like to thank Laura Feeney for her valuable feedback on the earlier version of this article.

Received: 30 November 2012 Accepted: 4 June 2013
Published: 19 June 2013

References

1. M Jacobsson, C Guo, IGMM Niemegeers, in *International Workshop on Localized Communication and Topology Protocols for Ad hoc Networks (LOCAN'05)*. A flooding protocol for manets with self-pruning and priorited retransmissions (Washington, DC, USA, November 2005). doi:10.1109/MAHSS.2005.1542772
2. M Jacobsson, C Guo, IGMM Niemegeers, An experimental investigation of optimized flooding protocols using a wireless sensor network testbed. Elsevier Computer Networks. **55**(13), 2899–2913 (2011). doi:10.1016/j.comnet.2011.05.024
3. Network simulator 2. http://www.isi.edu/nsnam/ns/. Accessed June 2013
4. SY Ni, YC Tseng, YS Chen, JP Sheu, The broadcast storm problem in a mobile ad hoc network, in Fifth Annual International Conference on Mobile Computing and Networking (MobiCom'99). (USA, Seattle, 01 August 1999). doi:10.1145/313451.313525

5. D Cavin, Y Sasson, A Schiper, On the accuracy of manet simulators, in Second ACM International Workshop on Principles of Mobile Computing (POMC'02). (France, Toulouse, October 2002). doi:10.1145/584490.584499
6. L Feeney, Towards trustworthy simulation of wireless mac/phy layers: A comparison framework, in 15th ACM International Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'12). (Paphos, Cyprus, October 2012), doi:10.1145/2387238.2387288
7. A A Stetsko, T Smolka, V Matyas, F Jurnečka, On the credibility of wireless sensor network simulations: evaluation of intrusion detection system, in 5th International ICST Conference on Simulation Tools and Techniques (SIMUTOOLS'12). (Italy, Desenzano del Garda, March 2012)
8. A Stetsko, M Stehlik, V Matyas, Calibrating and comparing simulators for wireless sensor networks, in IEEE 8th International Conference on Mobile Adhoc and Sensor Systems (MASS'11). (Spain, Valencia, October 2011). doi:10.1109/MASS.2011.80
9. C D Kotz, RS Newport, J Gray, Y Liu, C Yuan, Elliott, Experimental evaluation of wireless simulation assumptions, in 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems (MSWiM'04). (Italy, Venice, October 2004). doi:10.1145/1023663.1023679
10. HN Pham, D Pediaditakis, H Boulis, From simulation to real deployments in wsn and back, in IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'07). (Finland, Helsinki, June 2007). doi:10.1109/WOWMOM.2007.4351800
11. A Marchiori, L Guo, J Thomas, Q Han, Realistic performance analysis of WSN protocols through trace based simulation, in 7th ACM workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks (PE-WASUN'10). (Turkey, Bodrum, October 2010). doi:10.1145/1868589.1868606
12. GT Nguyen, RH Katz, B Noble, M Satyanarayanan, A trace-based approach for modeling wireless channel behavior, in 28th Conference on Winter Simulation (WSC'96). (USA, Coronado, December 1996)
13. F Hassani Bijarbooneh, P Flener, E Ngai, J Pearson, Energy-efficient task mapping for data-driven sensor network macroprogramming using constraint programming, in 12th INFORMS Computing Society Conference (ICS'11). (USA, Monterey, January 2011). doi:10.1287/ics.2011.0016
14. J Polastre, J Hill, D Culler, Versatile low power media access for wireless sensor networks, in Second International Conference on Embedded Networked Sensor Systems (SenSys'04). (USA, Baltimore, November 2004). doi:10.1145/1031495.1031508
15. A Boulis, Castalia: Revealing pitfalls in designing distributed algorithms in WSN, in 5th International Conf. on Embedded Networked Sensor System (SenSys'07). (Sydney, Australia, November 2007)
16. A Köpke, M Swigulski, K Wessel, D Willkomm, PK Haneveld, T Parker, O Visser, HS Lichte, S Valentin, Simulating wireless and mobile networks in omnet++: the mixim vision, in 1st International Workshop on OMNeT++. (Marseille, France, March 2008)
17. W Stalling, *Data and computer communications*, 9th edn. (Pearson Prentice-Hall, Upper Saddle River, 2005)
18. W Peng, XC Lu, On the reducation of broadcast redundancy in mobile ad hoc networks, in Sixth Annual International Conference on Mobile Computing and Networking (MobiCom'00). (USA, Boston, August 2000)
19. W Peng, X Lu, *Ahbp: An efficient broadcast protocol for mobile ad hoc networks*. (Journal of Science and Technology, Beijing, 2002)
20. B Williams, T Camp, Comparison of broadcasting techniques for mobile ad hoc networks, in Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02). (France, Lausanne, June 2002). doi:10.1145/513800.513825
21. H Shah-Mansouri, B Khalaj, M Pakravan, A Khodaian, Counter-based broadcasting: Modeling and performance analysis in csma-based wireless networks, in 20th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'09). (Japan, Tokyo, September 2009). doi:10.1109/PIMRC.2009.5449973
22. D Ganesan, B Krishnamachari, A Woo, D Culler, D Estrin, S Wicker, Complex behavior at scale: An experimental study of low-power wireless sensor networks. Tech. Rep. UCLA/CSD-TR 02-0013, UCLA Computer Science (2002)
23. V Naik, A Arora, P Sinha, H Zhang, Sprinkler: A reliable and energy efficient data dissemination service for extreme scale wireless networks of embedded devices. IEEE Trans. Mobile Comput. **6**(7), 777–789 (2007). doi:10.1109/TMC.2007.1013
24. A Tarp, *Experimental evaluation of flooding in ad-hoc networks. Bachelor's thesis, University of Dusseldorf*. (Dusseldorf, Germany, 2004)
25. GP Halkes, KG Langendoen, *Experimental evaluation of simulation abstractions for wireless sensor networkmac protocols. EURASIP J. Wireless Commun. Netw. 2010*, (2010). doi:10.1155/2010/601892
26. Connectivity project at the University of California at Berkeley. (2013). http://wsn.eecs.berkeley.edu/connectivity/. Accessed June
27. Wisebed - wireless sensor network testbeds. http://wisebed.eu/site/. Accessed June 2013

doi:10.1186/1687-1499-2013-169
Cite this article as: Jacobsson and Rohner: **Comparing wireless flooding protocols using trace-based simulations.** *EURASIP Journal on Wireless Communications and Networking* 2013 **2013**:169.