RBP: Robust Broadcast Propagation in Wireless Networks

Fred Stann John Heidemann Rajesh Shroff Muhammad Zaki Murtaza
USC/ISI
4676 Admiralty Way, Marina Del Rey, CA, USA
fstann@alumni.usc.edu, johnh@isi.edu, rshroff@usc.edu, mmurtaza@usc.edu

Abstract

Varying interference levels make broadcasting an unreliable operation in low-power wireless networks. Many routing and resource discovery protocols depend on flooding (repeated per-node broadcasts) over the network. Unreliability at the broadcast-level can result in either incomplete flooding coverage or excessive re-flooding, making path maintenance either unreliable or expensive. We present RBP, a very simple protocol that bolsters the reliability of broadcasting in such networks. Our protocol requires only local information, and resides as a service between the MAC and network layer, taking information from both. We show that RBP improves reliability while balancing energy efficiency. RBP is based on two principles: First, we exploit network density to achieve near-perfect flooding reliability by requiring moderate (50-70%) broadcast reliability when nodes have many neighbors. Second, we identify areas of sparse connectivity where *important links* bridge dense clusters of nodes, and strive for guaranteed reliability over those links. We demonstrate, through both testbed experiments and controlled simulations, that this hybrid approach is advantageous to providing near-perfect reliability for flooding with good efficiency. Testbed experiments show 99.8% reliability with 48% less overhead than the level of flooding required to get equivalent reliability, suggesting that routing protocols will benefit from RBP.

Categories and Subject Descriptors

C.2.1 Network Protocols—*wireless communications*. C.2.2 Network Protocols—*routing protocols.*

General Terms

Algorithms, Reliability, Experimentation, Performance

Keywords

Broadcasting, Reliability, Wireless communications, Sensor Networks

1 Introduction

Flooding is an integral part of many protocols and applications in wireless networks. Wireless routing protocols such as DSR [16], AODV [28], and ODRMP [19] flood route discovery messages. In sensor-networks, routing protocols (such as in [39]), resource discovery (such as directed diffusion [12]), and network-integrated database systems (such as TinyDB [22]) all depend on flooding to construct efficient data collection trees. Flooding-limitation protocols like SPIN [17] and BARD [34] resort to flooding when query history is not applicable. A range of applications, including query response [31], target tracking [15, 41], and signal processing [21], build on these systems or flood at the application-level. These systems depend on the completeness and timeliness of flooding to identify available resources, discover efficient paths, and coordinate in-network computation.

Flooding in wireless networks is most often achieved via each node *broadcasting*[1] the request to its neighbors, taking advantage of the shared wireless media [26]. While many wireless MAC protocols use link-layer ARQ (automatic repeat-request, often via an RTS-CTS-data-ACK exchange) to protect unicast traffic from collisions and corruption, they generally do not use ARQ for broadcast traffic because of the need to avoid control-traffic implosion [18]. Broadcast reliability often therefore directly reflects the reliability of the wireless channel. Numerous studies have documented a wide variance in broadcast reliability in low-power [33, 39, 40] and 802.11 [1] networks, thus it is not uncommon for individual broadcast messages to be lost or received by an incomplete subset of neighbors.

Broadcast packet losses can result in performance problems for flooding, and ultimately in routing and applications. In networks where density varies, broadcast failures near the source or in sparser areas can effectively disconnect a large portion of the network from a flood. Reliability of a flood can be defined as the percentage of all nodes that successfully receive the information [35]. An unreliable flood can result in slow setup for query-response applications [31], slow discovery of new routes, create inferior data collection trees, or provide incomplete information for target tracking or signal processing [21].

Prior work has sought to improve broadcast reliability, either by reducing sources of loss or providing broadcast retransmission. Examples of loss reduction include PHY-

SenSys'06, November 1–3, 2006, Boulder, Colorado, USA.

[1] Prior work uses several definitions of the term "broadcasting." In this paper, we use *broadcasting* only for single-hop wireless transmission from a node to all its reasonably available neighbors. (We clarify the definition of neighbor in Section 3.) We use *flooding*, by contrast, for network-level approaches that send information to all nodes in a connected, multi-hop network.

layer capture [37], MAC-layer TDMA [7], random slot selection [38], and application-layer jitter. These approaches may address collisions, but do not protect against other sources of loss, particularly in sparse topologies. Retransmission of broadcasts can be done at the MAC, routing, or application layers, but each can incur unnecessary overhead. It is difficult for the MAC layer to know when neighbors are unimportant or use implicit ACKs, and for applications to track the local topology. We review related work in detail in Section 2.

The contribution of this paper is to develop RBP, the *Robust Broadcast Protocol*, to provide adaptive reliability for broadcasts and improve the end-to-end reliability of flooding. RBP operates between the MAC and routing layers, taking information from both. It is distributed in the sense that every node makes its own decisions about retransmissions without any global or hard state. RBP exploits two observations. First, the level of reliability required of a broadcast is dependent on the local network density. Lower density requires a higher likelihood of retransmission after imperfect propagation, while in denser regions, transmissions from other nodes will likely compensate for loss. Second, network topologies sometimes consist of well-connected components joined by *important links*; identifying and increasing the reliability of these links is essential to provide both high reliability and efficiency. We describe these observations in detail in Section 3, and the RBP implementation in Section 4.

We build on prior analytic work [35, 39] to understand the effect of density on reliability in simple topologies, and confirm the effects of density and important links in simple topologies through simulation (Sections 5 and 6). Our primary results are based on a 20-node senor testbed (Section 7), augmented in Section 8 by simulation to explore a controlled range of density and packet loss. We show that the cost of RBP is small, and the net result is much greater efficiency when one considers net overhead to reach a targeted reliability. Timeliness is a natural side effect of our protocol, because a single flood will have a higher probability of achieving global coverage. Without our protocol, simple flooding may need to be repeated multiple times to achieve the same reliability.

RBP was motivated by our experiences implementing applications using diffusion [12]. One application implements spatio-temporal search, where repeated queries trace correlated events across several motion detectors to a camera [31]. Broadcast loss forced application-specific changes to diffusion to get adequate query latency. More generally, tuning periodic flooding in soft-state protocols like diffusion requires balancing control overhead against robustness to changing networks. With different applications and topologies there is no single good default for flood and timeout frequency, but nor can the application designer make an informed decision. We believe RBP simplifies these applications by provided consistent, efficient flooding.

2 Related Work

There is a very large body of related work in reliable broadcasting, both in wired and wireless networks. We briefly review prior work in wired networking, and wireless approaches that provide perfect and improved reliability. We then consider two different applications of broadcast in sensor networks today: software distribution and path discovery.

Wired networking: Reliable multicast algorithms for wired IP, such as SRM [8] and RMTP [27], provide scalable solutions that avoid the substantial overhead associated with strict reliability guarantees and worst-case requirements. To that extent our goals align with these protocols. We are interested in providing reliability for sensor network applications where the penalties of poor broadcast reliability are severe, but the requirement of 100% reliability can be relaxed slightly for efficiency. Both SRM and RMTP provide local repair triggered by the recognition of missing sequence numbers in a stream of packets. Our protocol differs in that it is intended for individual broadcasts that are not associated with a sequence. These protocols assume an existing routing tree; RBP instead focuses on single-hop communication and assumes any routing will be provided at higher layers.

Wireless for improved reliability: Probabilistic broadcasting simultaneously addresses two problems endemic to wireless networks: power limitation and collisions [25]. Typical wireless MAC layers can avoid broadcast collisions via random slot selection, but do not solve the hidden terminal problem. In probabilistic broadcasting nodes rebroadcast with probability p to reduce collisions and energy consumption. Probabilistic broadcasting exploits *phase transition* where networks with densities around 8 or more are connected with very high probability [32]. RBP instead supports a range of densities, including sparse (less than 8), weakly connected clusters, and variable density networks as are common in real deployments.

Gossiping approaches can provide high reliability, and are used for directory and database replication in wired networks [6], and as bimodal multicast in wired [2] and wireless [3] networks. Gossiping often employs multiple rounds of exchanges, alternating floods with local repairs of missed data. Like probabilistic broadcasting, bimodal techniques make partial delivery unlikely. RBP also follows flooding with repair, but recognition of failed delivery is accomplished by overhearing rather than meta-data exchange.

Heuristics can be added to what is essentially probabilistic broadcasting [10]. These heuristics prevent the death of a probabilistically forwarded broadcast around the source node and in areas of low density. In low density neighborhoods, the decision to not forward a broadcast (due to failed probability) can be overruled when an insufficient number of neighbors have been overheard to transmit the same broadcast. RBP is similar to this approach in that it is sensitive to local network density. RBP, however, does

repeated local repairs geared to local density. RBP also singles out problem links that interconnect network clusters and will resort to unicast repairs to ensure there is no partitioning of the network.

Area-based and neighborhood-based methods aim at minimizing the bandwidth consumed in broadcasting in MANETs while retaining near perfect accuracy [39]. Using complete or near-complete knowledge of node locations or hop-count distances, they strive to reduce redundancy during flooding. Nodes suppress broadcasts if they believe they are redundant. By contrast, RBP emphasizes high reliability as the primary goal, rather than efficiency, while requiring only local information.

Wireless with perfect reliability: Several TDMA schemes have been proposed that target contention-free reliable broadcast [5]. We focus on non-TDMA approaches because experimental results suggest that wireless connectivity is extremely volatile [39, 40], so it seems quite challenging to maintain perfect information to avoid collisions in practical networks.

Several protocols consider the problem of perfect distribution of code or related data. One important application in this space is reprogramming an entire sensor network, as in Trickle [20] or Deluge [4]. Trickle propagates small code updates (binaries of subroutines that fit in a network packet) using gossiping and link-layer broadcast, while Deluge propagates large code images. Another class is moving large objects to a specific node, as in PFSQ [36] and RMST [33]. PFSQ considers efficient streaming large objects with quick, NACK-based repair, and RMST explores trade-offs in caching and repair for point-to-point distribution. RBP differs from these algorithms by targeting good rather than perfect reliability, and by targeting propagation of small messages with high probability for routing and resource discovery as the primary application, rather than distribution of often multi-packet objects with certainty.

Applications to Routing and Resource Discovery: The primary application we consider is routing or resource discovery. AODV, DSR, ODMRP, TinyDB, Diffusion, IDSQ, and BARD all rely on flooding to determine good multi-hop paths in wireless networks. Some focus on IP-layer routing, while others combine path discovery (routing) with resource location (tracking services or resources such as sensors that meet a particular criteria). In general, these on-demand protocols flood when a user initiates a route or query, then select a specific path or multicast tree for further data transfer. These protocols use a variety of approaches to flood with high reliability, including local repair, and soft-state, periodic rebroadcast. RBP uses additional local information to improve the reliability of the flood, lowering the latency of initial discovery and potentially allowing less frequent subsequent floods. Specifics of diffusion are discussed in the implementation section in order to clarify the context of our experiments.

In conclusion, although there are many effective techniques for reliable multicast, efficient flooding, and replicating data to all nodes in a sensor network, we feel that RBP provides a unique contribution for applications that benefit from reliable single broadcasts employed for periodic resource discovery.

3 ROBUST Broadcast Algorithm

RBP addresses the problem of poor reliability for broadcasting in low power wireless networks of non-uniform density. As we stated in Section 2, we are not trying to make a single broadcast more efficient. Rather, we wish to make a single broadcast more reliable, thereby reducing the frequency with which an upper-layer protocol needs to invoke flooding. If an unreliable broadcast is received 90% of the time, then an application that required 99% reliability would need to broadcast twice to reach its target (assuming loss is independent). In Section 5 we quantify this tradeoff, showing that better reliability can also result in lower energy consumption.

The goal of RBP is to make broadcasting reasonably reliable. We next describe the four steps needed to meet this goal: tracking neighbors and floods, basic retransmission to reach a target reliability, adapting that target to network density, and identifying important links that require successful transmission.

First, RBP requires that a node know the identity of its one-hop neighbors. We define the one-hop neighborhood as nodes with which a node has inbound and outbound connectivity above a configurable threshold, over a moving window of broadcasts. This definition eliminates distant and weak neighbors, as well as neighbors with strongly asymmetric links. The threshold must be adjusted to account for the lowest common level of connectivity across bottleneck links. Details of how the neighbor list is acquired and maintained can be found in Section 4.

RBP also must understand when broadcasts by different nodes correspond to the same flood. It therefore generates and propagates a unique identifier when a new flood is initiated.

Second, RBP uses a simple algorithm for retransmission. The first time a node hears a broadcast it retransmits the packet unconditionally, as in a normal flood. As additional neighbors transmit the same packet, the node listens (snoops) ands keeps track of which neighbors have propagated the broadcast. Armed with one-hop neighbor knowledge, a node can ascertain the percentage of its neighbors that are guaranteed to have seen a packet. We call a transmission by a neighbor an *implicit ACK*, a term that typically refers to inbound traffic that indirectly acks outbound traffic [29]. When the number of implicit ACKs seen by a node falls below a predetermined threshold, a node will again retransmit the broadcast packet.

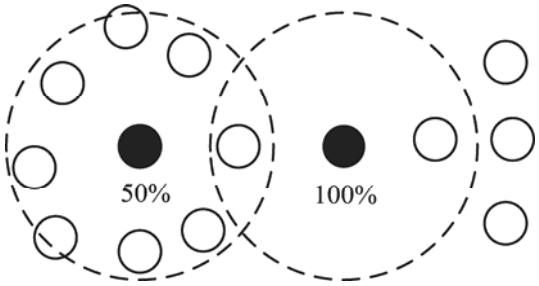


Figure 1: A sample topology with varying density and appropriate thresholds.

To recover from destination nodes that fail to rebroadcast and missed implicit or explicit ACKs, receivers send an explicit unicast ACK when they hear a repeated broadcast from the same sender. The broadcaster also switches to unicast when doing so reduces the expected number of packets (when more neighbors have received than not).

Third, a key optimization in RBP is that both retransmission thresholds and the number of retries are adjusted for neighborhood density (Figure 1). Higher density neighborhoods require lower thresholds with fewer retries, since other neighbors are likely to broadcast as part of the same flood. If, for example, there are three or less neighbors, a node will make up to three attempts to propagate the message to all neighbors. For four to six neighbors the threshold is 66% and the number of retries is two. When there is a dense local neighborhood (i.e. eight or more neighbors), propagation is considered successful when half of the neighborhood has received, and only one retry is attempted. The analysis in Section 5 and feedback from early test runs was used in selecting appropriate thresholds.

Finally, an additional important improvement is directional sensitivity by detection of important links. Real networks frequently have variable densities (this case arose frequently in our testbed), as shown in Figure 2. When an upstream dense section meets a downstream sparse section, there can be a node at the edge of the dense section that has a large neighbor count, but is the sole provider of traffic to the first downstream node in the sparse section. For example, in Figure 2 the black node is the sole provider of traffic to the grey node, but it resides in a dense neighborhood. When the grey node fails to hear a broadcast from the black node, retries will rarely happen with our basic algorithm because there is a high probability that at least 50% of the black node's neighbors will have acked the broadcast. The black node is incapable of recognizing its special relationship with the gray node, but the gray node can easily do so because all of its upstream traffic will come from the black node. Since bottleneck nodes are identified by their downstream neighbors, this approach generalizes to include a number of weakly connected networks, including multiple broadcasters on opposite sides of a special node.

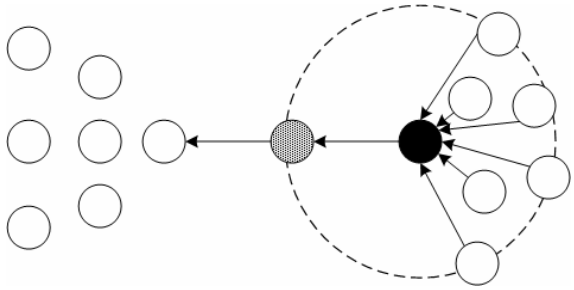


Figure 2:The path from the black to gray node forms an *important link*.

With our directional sensitivity optimization, nodes keep a histogram of which neighbor was the first to transmit a previously unheard broadcast. For the black node in Figure 2 there would be a relatively uniform distribution in such a histogram. The introduction of jitter for message forwarding in dissemination protocols guarantees that no single neighbor dominates. The gray node, however, sees most upstream traffic (for a time) from a single node. Our solution includes a moving window of time in which a directional histogram is maintained. If a single neighbor has a majority of the histogram, the node sends that upstream neighbor a control message indicating that it has a special relationship to this node. Any node that gets such a message will do up to 4 retries when that downstream node does not ack. The combined algorithm is shown in Figure 3.

Our current algorithm assumes unlimited memory in support of any number of neighbors. The testbed experiments described in Section 7 were run on Stargate-class nodes, and simulations were run under EmSim [9] without any memory constraints. To deploy RBP on platforms with limited

```
rbp_snoop_send (pkt) {
        if (broadcast)
                set rbp_timeout
        pass through to MAC
}

rbp_snoop_rec (pkt) {
        if ( (overheard broadcast by neighbor) OR (explicit ACK) )
                mark neighbor ACKed
        pass through to Routing
}

rbp_timeout {
        if ( (percentACKed < threshold (neighborCount) AND
                (retryCnt<maxRetry (neighborCount) )
                rebroadcast = TRUE
        else if ( (any important links) AND (retryCount < 4) )
                rebroadcast = TRUE
        else
                rebroadcast = FALSE
        if (rebroadcast){
                forward copy of bcast to MAC
                retryCnt ++
                reset rbp_timeout }
}
```

Figure 3: RBP PseudoCode

resources (like motes), modifications are required to basic RBP. A state-limited version of RBP might need to give preference to important bottleneck neighbors. Broadening RBP to handle a limited table size is a possible direction for future work.

4 RBP Implementation

To maximize modularity, we implemented RBP as a pass-through module that snoops on network traffic between the routing and MAC layers. We selected Directed Diffusion as the routing layer [12] and constructed RBP as a diffusion filter. We run diffusion over EmStar [9] and use the EmStar neighbor discovery services and B-MAC for media access [14]. Although our implementation depends on these services, the concepts are not specific and alternatives to each are available.

The RBP module tracks the status of each new broadcast packet, indexed by the unique flood identifier, to handle multiple overlapping broadcasts. RBP starts a timer (currently 10 seconds) and records the implicit and explicit ACKs for the broadcast. The length of this timer is not dictated by RBP. We selected a relatively long timer to detect relevant neighbors broadcast, since diffusion has a 1 second forwarding delay with 800ms of jitter. If the forwarding constants of the routing protocol were shorter or longer, we would have chosen a different constant. When the timer expires, the handler decides whether or not to retransmist the message. RBP keeps a duplicate of the original broadcast packet for retransmission. A cleanup timer periodically removes obsolete records. Explicit ACKs and directional sensitivity messages were implemented as unique control messages in diffusion.

Our choice for a data dissemination and routing layer was One-Phase-Pull Diffusion [11]. One-Phase-Pull employs a single flood for resource discovery that is repeated at randomly selected intervals in order to cope with changing network conditions. Node failure, energy depletion, mobility, fluctuating connectivity, and failed broadcasts are some of the reasons that diffusion takes a soft-state approach to route maintenance. Sinks flood an *interest* packet describing the desired attributes, and sources with matching attributes then establish a multicast routing tree back to the sink. Diffusion provides a convenient *filter* facility for the in-network monitoring and/or modification of data as it moves through a network [13]. The filter mechanism allows for the selectable modification of semantics between the routing layer and layers above or below. We encapsulated RPB in a loadable filter module in order to accomplish our dual requirements of layering and modularity. Our RBP module intervenes between the MAC and routing layer, and will be made available in released diffusion as the *RBP filter*, named for the algorithm.

An important requirement of RBP is that each node know its one-hop neighbors. This information is conveniently available in EmStar by including certain modules (linkstats, neighbord, and blacklist) in the EmStar stack of network

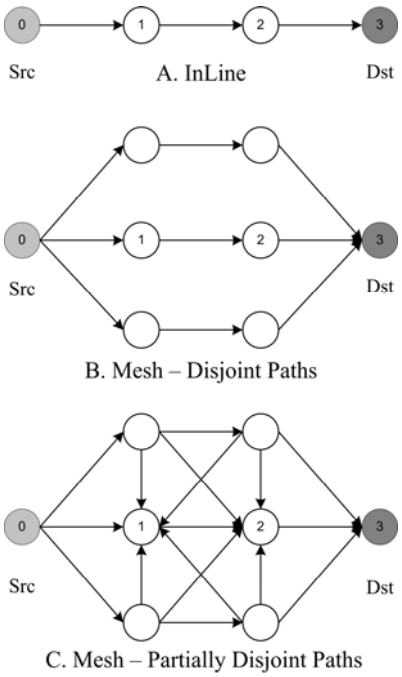


Figure 4: The Influence of Multiple Paths on Probability

modules. These modules are routinely included when running diffusion over EmStar, in order to support the informed selection of end-to-end unicast routes. To implement the neighbor service, EmStar attaches sequence numbers to broadcast packets and piggybacks inbound connectivity numbers in a round robin fashion (some in each broadcast) so that neighbors are aware of their outbound connectivity. Connectivity is determined as a weighted average by examining inbound sequence numbers and connectivity numbers from neighbors across a window of 12 broadcasts. To avoid list churn, the EmStar client informs the neighbor service of upper and lower reliability thresholds (70% and 60% in the ISI testbed). Once a node drops below the lower threshold, it must rise above the upper threshold in order to be again considered a neighbor.

5 Analysis

Previous work has provided detailed analysis of flooding reliability in wireless networks [35, 39]. Viswanath and Obraczka define flooding reliability as the total number of nodes reached by a broadcast (N_T) divided by the total node count for the network (N_R) [35]. They characterize flooding as a *wave* of broadcasts extending from the source, with each *tier* an additional hop distant.

$$\frac{N_T}{N_R} = \frac{1}{N_R}\left(P_S N + P_S N \frac{(P_b N \beta)^l - 1}{P_b N \beta - 1} \right) \qquad (1)$$

In this equation $P_S N$ is the expected number of neighbors that will successfully receive a transmission from a source node, $P_b N$ is the expected number of nodes that will receive at least one secondary transmission, β is the expected increase in area coverage (around 41%), and l is the hop count of the flooding wave. In Equation 1, increasing density will increase the

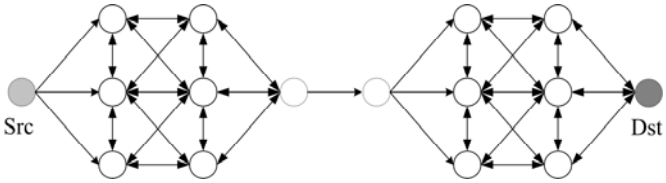


Figure 5: Topology with a Bottleneck

expected number of nodes that successfully receive a broadcast packet. Most analysis of wireless flooding is simplified by an assumption of uniform density. Because our protocol is density sensitive, a more interesting analysis for our purposes deals with variable density. Wireless broadcasting can be modeled schematically as a set of discrete connections [35]. Shared-edge analysis [30] is an effective tool for predicting the probability of eventual delivery of a broadcast packet to a set of nodes given their exact connection patterns. Topologies with high density near the source of a broadcast and lower density away from the source will have misleadingly good reliability numbers using Equation 1 and associated formulas.

We use Figure 4 to demonstrate how the addition of multiple paths increases end to end success probability. We ignore the possibility of broadcast collisions and simply focus on point to point probability. If the per link reliability is 85%, then end-to-end reliability in 4A will be 61%. Pu et al. demonstrated that the greatest influence in reducing end-to-end failures comes from multiple disjoint paths that do not share edges [30]. In 4B we can calculate the end-to-end probability as:

$$P[E2E] = 1 - \left(1 - p_1{}^{h_1}\right) * \left(1 - p_2{}^{h_2}\right) * \left(1 - p_3{}^{h_3}\right) \quad (2)$$

Using this equation, we see that adding two disjoint paths brings our end to end probability up from 61% to 94%. If we now consider that broadcasts can propagate in any direction, there is an additional increase in end-to-end reliability. In 4C we show how the reliability of the path through nodes 0-1-2-3 is increased by side and back transmissions. Using basic conditional probability, the reliability of that path becomes 99.9%, which becomes a lower bound on the overall probability.

Flooding is inherently reliable when there is uniform density due the existence of numerous partially disjoint paths. Testbed experience has shown us that wireless environments often display uneven density in their connectivity patterns. Dense pockets are often interconnected by small numbers of reliable links. Figure 5 shows an idealized topology in which two dense neighborhoods are interconnected by a single link. In this case the end- to-end reliability will have an upper bound dictated by the single link in the middle of the topology (85%). Even though our testbed in Figure 10 appears to have a relatively uniform topology, we often saw similar poorly connected components, with weak links between rooms 1165/1167 and 1117/1119 nearly partitioning the topology in the middle.

Several authors have examined the importance of transmission failures and collisions on wireless broadcast propagation [10, 35]. These works conclude that the most important broadcast is the initial broadcast. If it fails, the inherent reliability of multiple paths is lost. Looking again at Figure 5, the context of the initial broadcast from the source is replicated at the destination node of the important link in the middle of the topology. Bolstering the reliability of broadcasts only at the source node will not deal with situations such as that presented in Figure 5.

6 RBP Initial Simulation Results

Having conceived of an algorithm that adapts to poorly connected topologies like that of Figure 5, we wanted to evaluate the feasibility of our scheme. We therefore ran a series of preliminary simulation experiments based on our analysis.

6.1 Methodology

The simulation environment we employed was EmSim to allow our network code to run unaltered while providing simulated radios, channels, and error models [9]. Our simulation MAC layer provides multiple access with collision avoidance. The EmSim MAC does not do retries for broadcasting, as is typical in most wireless protocols. Software loaded above the MAC layer included the diffusion filter core, the one-phase-pull routing filter, the RBP filter (when using robust broadcast), and simple source and sink apps. We configured One-phase-pull Diffusion [11] to initiate a resource discovery broadcast every 60 seconds. The flooding rate was not accelerated to yield more samples per unit of time because we did not want floods to overlap one another.

RBP does not take into account loss due to congestion induced by overlapping floods. We consider congestion orthogonal to our focus and did not want to complicate our analysis with its effects at the MAC and network levels. Diffusion initiates flooding at an mean rate of once per minute, which is quite modest. We programmed sources to not transmit actual sensor data, because such transmissions are unicast and so are not the focus of this paper. Other than diffusion control messages, there is therefore no other competing traffic in our experiments. Every data point on a graph represents the average of 10 runs, and 95% confidence intervals are shown. Each run was one-hour long allowing for 60 broadcasts. We collected per-node statistics by instrumenting our code to log relevant events and byte counts.

6.2 Simulation Error Model

Because our protocol is intended to handle reliability problems endemic to wireless networks, we wanted an error model for our simulations that approximates what happens in the real world. The EmSim error model uses transmission power, distance, and a normal distribution of receive failures

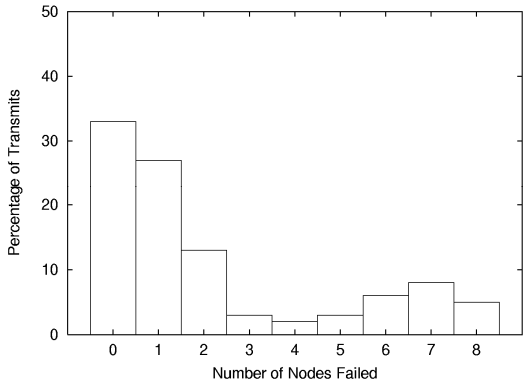


Figure 6: Observed Distribution of Node Failures for Eight Real-World Neighbors

to calculate the probability of packet loss between a pair of nodes. This calculation is done independently for each combination of transmitter and potential receiver (i.e. every other node in the simulation). We noted that the error model did not have any correlated receiver loss.

We wished to validate, therefore, if real-world packet losses could be spatially correlated. To that end we profiled simple loss between a real-world transmitter and a set of receivers. We placed eight Stargate nodes in a circular pattern around a ninth node which transmitted a 68-byte packet once per second for an hour. Nodes were approximately 12 feet from the transmitter and transmit power was set to -4 dBm. We conducted several experiments, with and without obstructions, as described in detail elsewhere [23].

The histogram in Figure 6 shows the distribution of how many receivers failed individual transmissions (as a percentage of 3600 attempts) when no obstructions were inserted between or near nodes. There are several interesting aspects to this bimodal distribution. First, the pair-wise independence of receive errors is evident on the left side of the histogram, as shown by its exponential-like decay. Second, the non-trivial peak on the right shows that correlated transmission failures are present even when there are no line-of-sight obstacles.

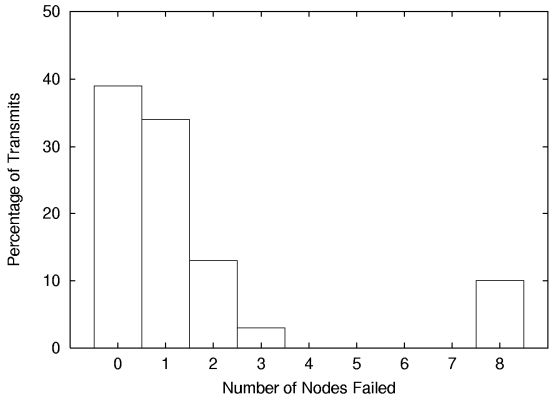


Figure 7: Error Model Employed in Simulations

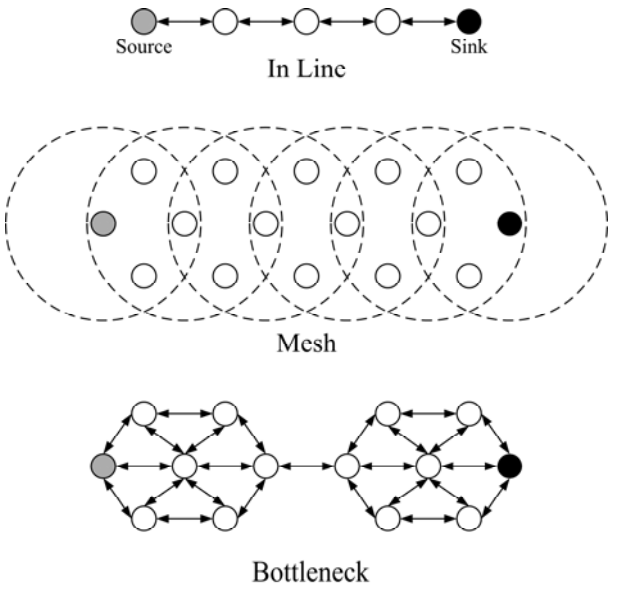


Figure 8: Three topologies used in Simulation Experiment

The fact that loss of real-world broadcast traffic is sometimes independent at each receiver, and sometimes correlated across all receivers, is essential to capture. This presence of both spatially-correlated and independent loss allows the potential for neighbor-based repair in dense topologies.

Influenced by this data, we added a *correlated* failure model to EmSim's existing *independent* failure model. Although modeling real-world interference patterns in simulation is difficult, we wish to provide simulations representative of real world circumstances that RBP will be required to handle. With our addition, packets have some chance of correlated failure, in which case they are received by no neighbors, then they are subject to the existing, pair-wise loss model at each receiver. Figure 7 shows the bimodal combination of these effects. We feel this is a reasonable approximation of the histogram in Figure 6.

6.3 Metrics

We express experimental results in terms of: reliability, bytes-per-flood, and a reliability cost metric (RCM). As we noted with analysis (Section 5), when reliability is defined as the percentage of nodes that receive a broadcast, topologies with high density near the source of a broadcast and lower density elsewhere may have misleadingly good numbers. We therefore define reliability as the percentage of floods that traverse the network diameter (reaching the outermost tier) divided by the total number of floods initiated in an experiment. In our initial measurements, using either definition of reliability did not significantly affect the reliability of RBP. Simple flooding, however, often had significantly lower network-diameter reliability than node-percentage reliability.

In all of our experiments we define an event as the initiation of a broadcast by a sink node. Bytes-per-flood is the

sum total of byte transmissions in a network triggered by a single event. Without RBP, it equates to the number of bytes contained in the broadcast packet times the number of nodes that transmitted it. The number of nodes may be less than the total node count if the flood fails to reach all nodes. RBP adds the cost of retransmissions and explicit ACKS.

Bytes-per-flood, by itself, is a poor metric to use for comparing protocols because it does not take reliability into account. If several floods are required to guarantee a certain level of reliability, then the true cost of reliability is the combined cost of the floods. We needed a metric that reflects the true cost of achieving a robust broadcast. To that end we use a Reliability Cost Metric (RCM). To calculate the RCM we first solve for the number of floods required to achieve near-perfect reliability (.99), given the propagation reliability.

$$.01 = (1 - P(S))^F \quad (3)$$

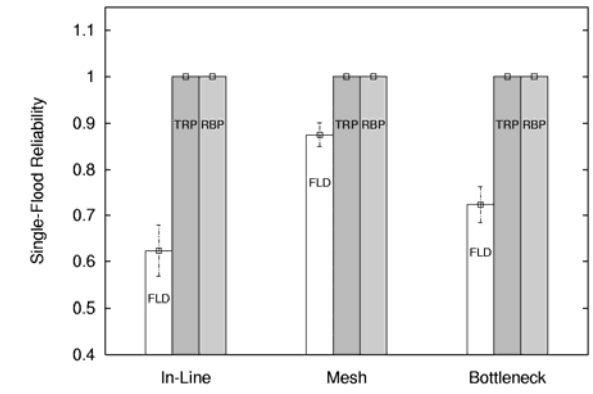
We then multiply the number of floods required to achieve near-perfect reliability by the bytes-per-flood and normalize by the cost of a "perfect" flood. A perfect flood achieves 100% reliability with each node transmitting the broadcast packet exactly once.

$$RCM = (F * BytesPerFlood)/(Nodes * PktSize) \quad (4)$$

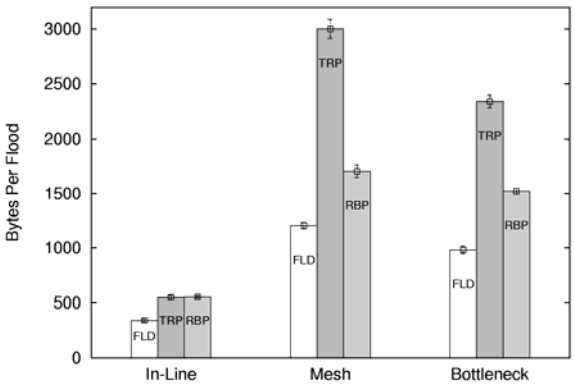
As an example of RCM, suppose there are 20 nodes, an 80 byte broadcast packet, propagation reliability of 85% (F is 2.4), and a measured bytes-per-flood of 1200; RCM will be 1.8. Although fractional broadcasts are not possible in the real world, we allow continuous values of F. RCM is a metric for fine-grained normalized comparison. We do not want broadcasts with RCMs of 1.1 and 1.9 to both quantize to 2.0.

6.4 Effect of topology on flooding

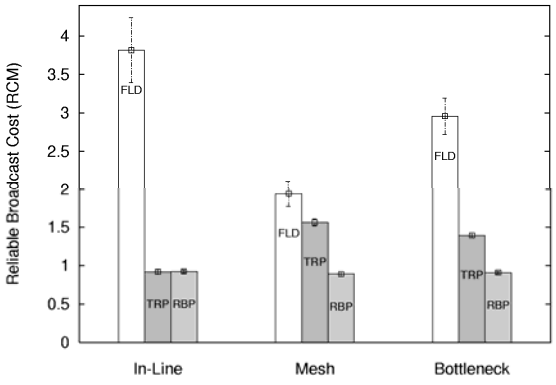
For these initial simulations, our goal was to demonstrate the basic concept of RBP and to explore the effects of density and topology. We created three different topologies with a varying degree of shared edges. The three topologies are shown in Figure 8: inline, mesh, and bottleneck. The inline topology consists entirely of shared edges, the mesh topology has no shared edges, and the bottleneck topology introduces a single shared edge halfway between source and sink nodes. Our hypothesis is that simple flooding performance relative to RBP would deteriorate with an increase in the number of shared edges. We expected that floods without RBP over the inline topology would display the exponential decay in reliability predicted by Equation 2. The mesh topology should demonstrate the inherent reliability of flooding in uniform topologies with multiple neighbors. Between these two extremes, the bottleneck topology should demonstrate how a single shared edge can negatively affect end-to-end reliability.



A. Reliability per Single-Flood



B. Bytes per Flood



C. RCM Cost per Algorithm

Figure 9 Reliability, Bytes-per-flood, and RCM for FLD, TRP, and RBP over Three Simulation Topologies

We also wanted to understand how RBP compares to a protocol that attempts to guarantee broadcasts are received by all neighbors without adapting to density. Such an approach is typical of MAC-only reliable broadcast. In order to answer that question we created the Total Reliability Protocol (TRP), a version of RBP that retries any broadcast up to the threshold (4) whenever less than 99% of the neighbors received the broadcast. We evaluate TRP with the same normalized cost (RCM). With our error model one can show that TRP should easily achieve near perfect reliability for flooding over the topologies in Figure 8. Figure 9 plots our

three metrics for each protocol over the three simulation topologies. Simple flooding is designated as FLD; RBP and the total reliability protocol are marked with their acronyms.

The reliability results, summarized in Figure 9A, are in line with our expectations. Reliability in a purely in-line topology, without RBP, decays exponentially with hop count ($.88^h$). Pure flooding (FLD) reliability was 62% for the in-line topology, and both RBP and TRP had perfect results (1.0). The mesh topology run with FLD had a relatively good reliability of 88% (in line with our analysis), and again RBP and TRP were perfect. The bottleneck topology resulted in a non-RBP reliability of 72%, which is only slightly worse than the result predicted by Equation 2 (77%).

As shown in Figure 9B, the transmission cost for a single flood is always higher for RBP and significantly higher for TRP. The high costs of TRP are because it forces all nodes to retry persistently with each neighbor, even if another neighbor has sent the same information through another path. Notice that when multiple disjoint paths provide inherent reliability the difference in bytes per flood between RBP and FLD is less (as a percentage). Also, considering the graphs in 9A and B, RBP is providing the same reliability as TRP at a much lower cost.

The most important metric in our estimation is RCM. Looking only at graphs 9A and 9B, it is difficult to gauge the relative cost of providing reliability with each protocol. The normalized cost (RCM) shown in Figure 9C demonstrates the significantly lower cost of achieving high reliability with RBP than simple flooding. For example, in the bottleneck topology the exponent in our RCM equation for FLD is 3.5. In other words it would take 3.5 floods (unquantized) to achieve 99% reliability. This number of floods multiplied by the bytes-per-flood and normalized by a perfect flood results in an RCM of 3.8 which is 3.1 times worse than RBP. A very interesting result in Figure 9 is how the cost of TRP changes relative to RBP in different topologies. The in-line topology RCM is nearly identical for both TRP and RBP. This is because RBP degrades into TRP in low density networks. Every neighbor is effectively important for RBP in the in-line topology. The Mesh topology provides the greatest density, and the greatest difference in RCM between TRP and RBP, with TRP 76% worse. TRP is unnecessarily attempting to achieve perfect reliability in dense neighborhoods. The bottleneck topology is between the in-line and mesh topologies with TRP 54% worse than RBP.

This experiment had the primary result that we had predicted. The cost of doing multiple unreliable floods in order to guarantee reliability is actually higher than a single robust flood using RPB, and RBP is cheaper than a protocol which does not exploit the inherent reliability of dense neighborhoods. We next apply our protocol to a real-world environment with a less predicable problem space.

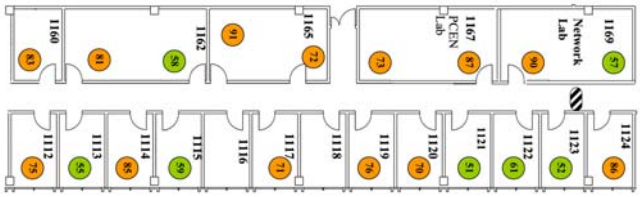


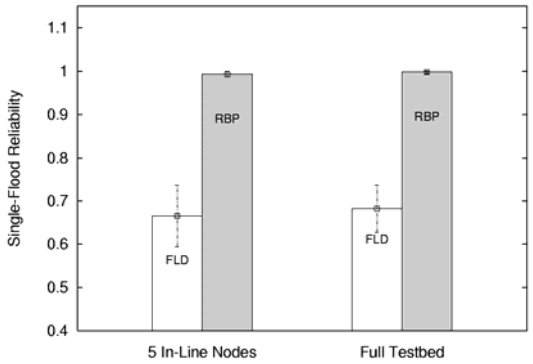
Figure 10: Testbed Topology at USC/ISI

7 RBP Testbed Results

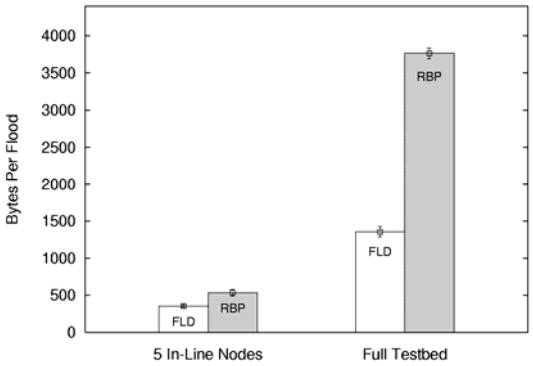
Having validated some basic assumptions about RBP in simulation, we wanted to validate the protocol on a real-world testbed. Our simulations helped narrow the design, but we depend on testbed experiments to capture real-world nuances that may be missed in simulation. The testbed that we used consisted of 20 Stayton and Stargate nodes. These are small form factor systems running Linux over a 32-bit Intel processor, with 64M of SDRAM, and 32M of flash memory. They are fitted with a multi-channel radio capable of 38.4 Kbaud. Radio range can be modified to enable multi-hop experiments. The actual layout of our testbed is shown in Figure 10. Nodes are placed in private offices and labs separated by walls. Connectedness was nonuniform with pockets of good connectivity separated by areas of low density. Power was intentionally set to enable multi-hop.

7.1 Five In-line Nodes

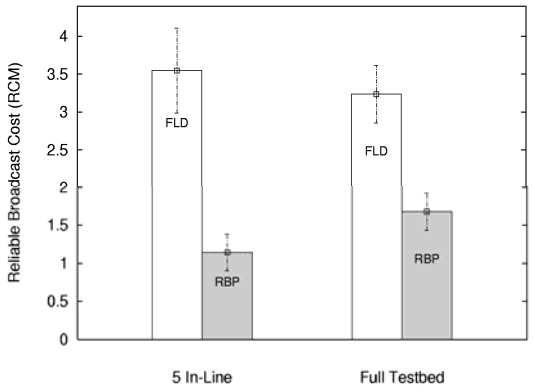
In our first testbed experiment we enabled 5 nodes from the testbed such that each node had one and only one upstream and downstream neighbor. We thereby created one of the topologies that we presented in both the Analysis and Simulation Sections. This allowed us to do a limited comparison of our simulation model to the real world and also gauge the average error rate on single links in the testbed. Looking at the left side of Figure 11A we see that FLD over four hops had an average end-to-end loss rate of 66%. This is very close to loss rate we saw in simulation for 5 in-line nodes (62%). The average per-link reliability for the four testbed links was 89%.If we compare the RCMs for 5 in-line testbed nodes in Figure 11C to the RCMs for in-line simulation in Figure 9C, we see that FLD numbers are very close (3.55 vs. 3.82). We concluded that we had a good correspondence between simulation and testbed for a simple topology. We omit TRP from our testbed experiments since simulations validate our assertion that TRP is not competitive.



A. Reliability per Single Flood



B. Bytes per Flood



C. RCM Cost per Algorithm

Figure 11: Single-Flood reliability, Bytes Per Flood, and RCM for two topologies in the USC/ISI testbed.

7.2 Full Testbed Experiment

The primary experiment that we performed on the testbed was to place a source and sink node at opposite ends of the testbed, ensuring that the maximum possible network diameter would be traveled by each flood. Initial results from this experiment were disappointing. RBP was only slightly more reliable for a single flood than simple flooding, and the RCM was worse for RBP because the higher per flood cost did not yield significantly better reliability. Log analysis revealed a flaw in our initial algorithm, which motivated our work on important link detection as described in Section 3.

The result of adding directional sensitivity to our protocol is reflected in the results shown in Figure 11 A, B, and C. Single flood reliability averaged 99.8% for RBP vs. 68.2% for simple FLD. Although the bytes-per-flood is 2.7 times higher for RBP, the RCM cost for RBP is 48% less than FLD. In addition to motivating important-link detection, our testbed experiments demonstrate that directional sensitivity can be as important as density sensitivity in order for RBP to achieve reliability at a reasonable cost.

8 Further Simulation Results

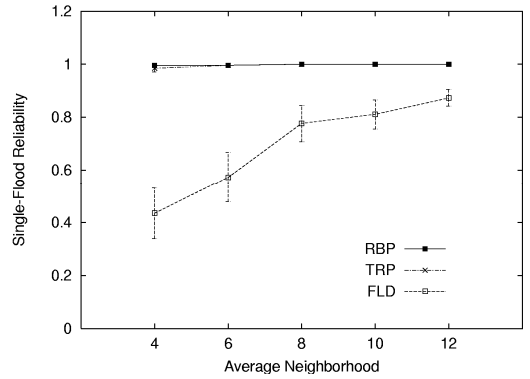
Having shown the need for density and directional sensitivity, we now turn to a systematic exploration of the design space. Simulation allows for the exploration of a multi-dimensional problem space in a manner that is impractical with an actual testbed.

8.1 Methodology

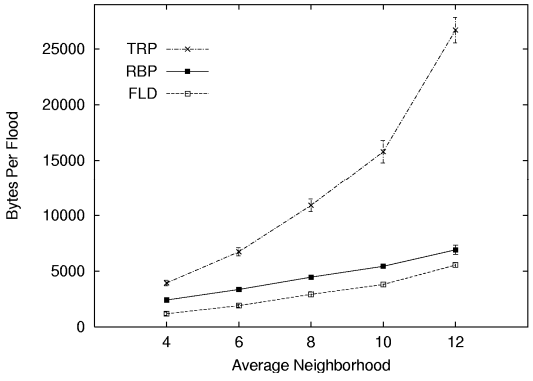
For these experiments we used random node placements. Every data point presented in this section represents an average over ten randomly generated topologies. For random node placement we employed an existing topology-generator program [11] capable of creating EmSim topologies of programmable size and node count. The generator has useful options such as approximate corner placements of sources and/or sinks in order to maximize the distance (hop count) between data producers and consumers. The generator tests topologies for connectedness given the radio range. In order to increase the number of tiers traversed by a broadcast we employed a 90 by 30 meter area and placed the source and sink nodes at diagonally opposite ends. Average local neighborhood size was calculated as $(N\pi*R^2)/A$, where N is the total node count, R is the maximum typical radio transmission range (12.5 meters), and A is the total simulation area. The packet error rate was achieved by the error model that we evolved in Section 6.1. Our goal in these experiments was to explore the role of density and packet-loss rates on protocol performance.

8.1 Density Effects on Overhead

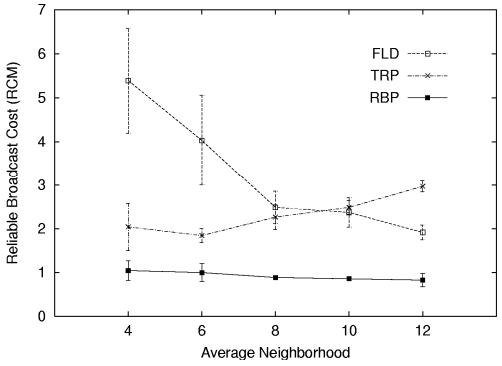
Because density plays a major role in our analysis, we felt that an obvious investigation was to make density the independent variable in an experiment. We therefore used simulation to vary node density (average neighborhood size) while holding the packet error rate constant. The primary method RBP uses to achieve efficiency is by adapting to local density. TRP is really just RBP without density sensitivity, and initial simulations seemed to indicate that RBP achieved the same reliability at much lower cost. Sensitivity to density allows us to economize in dense neighborhoods where the existence of multiple disjoint paths can facilitate the propagation of a broadcast.



A. Reliability per Single Flood



B. Bytes per Flood



C. RCM Cost per Algorithm

Figure 12: Single-flood Reliability , Bytes Per Flood, and RCM with Varying Density as an Independent Variable

We wished to calculate if increasing density eventually obviates the need for a protocol in order to achieve robust broadcasting at a low normalized cost.

Our expectation for this experiment was that increasing density would increase the reliability of simple flooding (FLD) and thus minimize the difference in RCM between RBP and FLD. We also expected TRP to deliver the same reliability as RBP, but at a much higher price. Results for single-flood reliability appear in Figure 12A. As average density increased from 4 to 12 neighbors, the positive shift in reliability for flooding is quite dramatic. Reliability increased from 43% with 4 neighbors to 87% with 12 neighbors. Looking at Figure 12B, the difference in bytes-per-flood between RBP and FLD remains relatively constant, but TRP increases non- linearly. Because RBP and TRP achieve nearly identical reliability at vastly difference cost, the RCM graphfor this experment (Figure 12C) yields one of our most interesting results. With 10 neighbors or more, the RCM cost is higher with TRP than flooding. In other words, with high density, multiple floods to achieve 99% percent reliabiliity is cheaper than a single TRP flood. Because the reliability is converging in Figure 12A, we expect that guaranteed high density lessens the need for RBP.

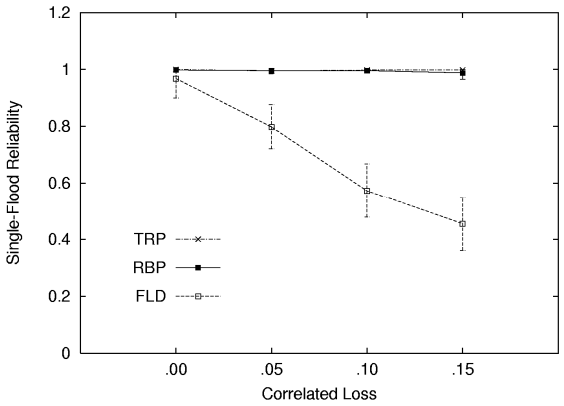
8.2 How does Error Rate affect the Performance of RBP vs. non-RBP?

We saw that high density reduces the performance and advantage of RBP. Will decreasing the error rate have a similar affect? Complicating this research question was the fact that our error model has two components: correlated errors and pair-wise errors. We decided to vary each of these components independently, while holding the other constant. Density was also held constant at 6 neighbors. Our expectation was that reducing correlated errors would, like increasing density, result in a reduced performance advantage of RBP over simple flooding. We did not, however, expect the same results from varying the pair-wise error rate. Our error-model analysis (Section 6.1) made the point that the likelihood of multiple pair-wise errors decays exponentially. The pair-wise error rate would need to get unrealistically high before a high percentage of disjoint paths between broadcast tiers failed to forward a flood.

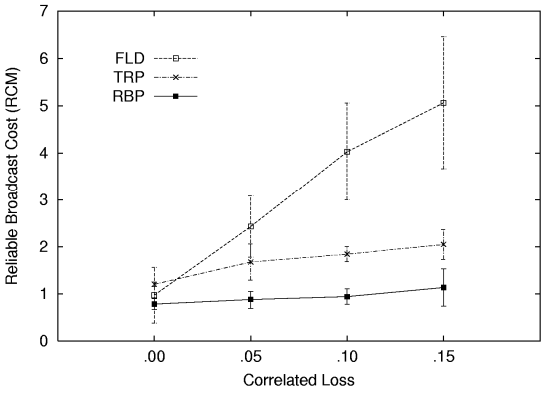
The result of varying the correlated loss is shown in Figure 13. The results are in line with our expectations. Similar to increasing density, decreasing the correlated error rate causes convergence between RBP and simple flooding in terms of both reliability and normalized cost (RCM). With correlated loss set to zero, the single flood reliability of FLD is 97% and RBP is 99.8%. At the other extreme, with correlated loss at 0.15 the single flood reliability of FLD is 46% and RBP is 99%. The RCM numbers, shown in Figure 13B follow suit. In effect, if there are no correlated errors (an unrealistic expectation in wireless networks) and uniform moderate density then simple flooding is reliable.

Varying the pair-wise errors was accomplished by shifting the center of the normal distribution for receiver noise in .05 increments while holding transmit errors constant at the value used in our standard error model. The receiver noise distribution, when centered on zero, gives the loss rates shown in the left side of the bimodal error model of Figure 7 in Section 6.1.

The results of this experiment, shown in Figure 14, show much less impact on simple FLD than when varying transmit failure rates. This is in line with our analysis and the distribution of our error model.

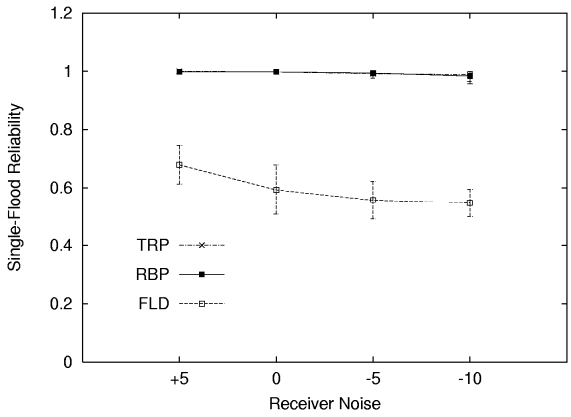


A. Reliability per Single Flood

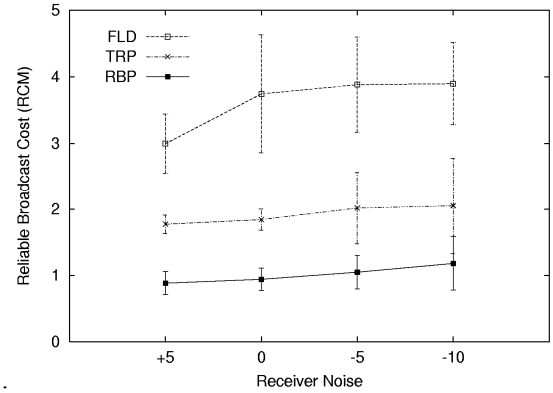


B. RCM Cost per Algorithm

Figure 13: Single-flood Reliability and RCM when Varying Correlated Loss



A. Reliability per Single Flood



B. RCM Cost per Algorithm

Figure 14: Single-Flood Reliability and RCM when Varying Receiver Loss

Although one, two, or three single link failures are far more likely in our error model than a correlated error, the likelihood of a large percentage of neighbors failing during pair-wise calculations is very small. Flooding is tolerant of individual link failures when network density provides multiple disjoint paths for the continued propagation of the broadcast wave. Pair-wise errors have a much larger influence in sparse neighborhoods. When a node has a single upstream or downstream neighbor, the effect of pair-wise errors is the exactly same as for correlated errors. Because RBP does up to 4 retries in sparse neighborhoods, it handles pair-wise errors in low density in the same manner as it handles transmit errors.

9 Future Work

Several issues not addressed in this paper provide fertile ground for future work. We have not considered mobility, which will greatly increase the rate of topology change. In a mobile, the identification of one-hop neighbors must occur at a rate that adapts to this variability, or updates must occur as needed. Also, the method used to identify one-hop neighbors may need to be adapted to a smaller window of samples that is updated more frequently.

Another direction for future work is to implement RBP in other frameworks. Because RBP is not specific to sensor networks or directed diffusion, it should be easily ported to other protocol stacks.

Finally, a state-limited version of RBP is a key challenge that needs to be met for deployment on motes, where memory limitations and static allocation preclude a complete neighbor list in dense topologies. Fortunately dense topologies have the least need of precise neighborhood knowledge, but important links need to be accounted for.

10 Conclusions

In this paper we explored how broadcast reliability interacts with flooding in wireless networks. While there is a great deal of prior work in the area of reliable broadcast, most of it focuses on efficient flooding, in simulated topologies, often with multi-hop topological information. We instead focused on the end-to-end reliability of flooding and the study of topologies with variable density as we found in our testbed. In addition, we developed a very simple mechanism that uses only local density information, sometimes augmented by indications from immediate neighbors of important links. We showed that this combination was both effective at obtaining near-perfect

reliability, and much more efficient than either repeated flooding or guaranteed transmissions. Most importantly, we demonstrated that the cost of achieving 99% reliability for broadcast propagation in a real testbed is on average 48% lower with RBP than simply increasing the flood rate. To evaluate RBP we developed a new wireless transmission error model, based on controlled experiments, that considers both pair-wise and correlated packet loss. Our key conclusions were shown by experiments in a 20-node wireless testbed, augmented with simulations to explore the parameter space.

ACKNOWLEDGMENTS

We would like to thank Fabio Silva and the other I-LENSE members for helping maintain our testbed. The comments of the anonymous SenSys reviewers, and particularly Phil Levis, our shepherd, greatly clarified the technical presentation.

References

[1] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. Link-level Measurements from an 802.11b Mesh Network. In Proceedings of the ACM SIGCOMM Conference, pp. 121-132. Portland, Oregon, USA, ACM. August, 2004.

[2] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, Y, Minsky. Bimodal Multicast. ACM Transactions on Computer Systems (TOCS), Volume 17 , Issue 2 Pages: 41 – 88, May 1999

[3] R. Chandra, V. Ramasubramanian, K. Birman. Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks. In Proc. 21st International Conference on Distributed Computing Systems (ICDCS), pages 275-283, 2001.

[4] Adam Chlipala, Jonathan Hui and Gilman Tolle, \Deluge: Data Dissemination in Multi-Hop Sensor Networks," UC Berkeley CS294-1 Project Report, December 2003.

[5] I. Cidon and M. Sidi. Distributed Assignment Algorithms for Multihop Packet Radio Networks. *IEEE Transactions on Computing*, 38:1353-1361 Oct.1989.

[6] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic Algorithms for Replicated Database Maintenance. In *Proc. ACM Symposium on Principles of DistributedComputing*, pages 1-12, 1987.

[7] A. Ephremides and T.V. Truong. Scheduling Broadcasts in Multihop Radio Networks. *IEEE Transactions on Communications*, 38(4):456-60, April 1990.

[8] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. In IEEE/ACM Transactions on Networking, Volume 5, Number 6, pp. 784-803, December 1997.

[9] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin. Em*: a Software Environment for Developing and Deploying Wireless Sensor Networks. *In Proceedings of the 2004 USENIX Technical Conference*. 2004

[10] Z. Haas, J. Halpern, L. Li. Gossip-Based Ad Hoc Routing. In *Proceedings of the IEEE Infocom*. Pages 1707-1716. New York, NY, June 2002.

[11] John Heidemann, Fabio Silva, and Deborah Estrin. Matching Data Dissemination Algorithms to Application Requirements. In Proceedings of the ACM SenSys Conference, pp. 218-229. Los Angeles, California, USA, ACM. November, 2003.

[12] John Heidemann, Fabio Silva, Charlemek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building efficient wireless sensor networks with low level naming. In *Proceedings of the Symposium on Operating System Principles*, pages 146-159, Chateau Lake Louise, Banff, Alberta, Canada, October 2001.

[13] John Heidemann, Fabio Silva, Yan Yu, Deborah Estrin, and Padma Haldar. Diffusion Filters as a Flexible Architecture for Event Notification in Wireless Sensor Networks. USC/ISI Technical Report 2002-556

[14] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System Architecture Directions for Networked Sensors. In *ASPLOS 2000*, Cambridge, USA, November 2000.

[15] Q. Huang, C. Lu, and G. Roman, Spatiotemporal Multicast in Sensor Networks. In Proceedings of the ACM SenSys Conference, pp. 218-229. Los Angeles, California, USA, ACM. November, 2003.

[16] D. Johnson, and D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. in Mobile Computing, pages 153-181. Kluwer Academic, 1996.

[17] J. Kulik, W Rabiner, and H. Balakrishnan. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pages 174-185, Seattle, Washington, USA 1999

[18] LAN MAN Standards Committee of the IEEE Computer Society, Wireless LAN medium access control (MAC) and physical layer (PHY) specification. IEEE Std. 802.11, IEEE 1997

[19] S.-J. Lee, W. Su, and M. Gerla. On-demand multicast routing protocol in multihop wireless mobile networks. *ACM/Kluwer MONET*, 7(6):441–453, Dec. 2002.

[20] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code maintenance and propagation in wireless sensor networks. In Proceedings of the 1st USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI), 2004.

[21] D. Li, K. Wong, Y.H. Hu, and A. Sayeed. Detection, Classification and Tracking of Targets in Distributed Sensor Networks. IEEE Signal Processing Magazine, vol. 19, no. 2, March 2002.

[22] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. The Design of an Acquisitional Query Processor for Sensor Networks. In *Proceedings of the ACM SIGMOD*, 2003.

[23] M. Murtaza, J. Heidemann, F. Stann. Studying the Spatial Correlation of Loss Patterns among Communicating Wireless Sensor Nodes. USC/ISI Directed Research Report 2005.

[24] Nahata, P. Pamu, S. Garg, and A. Helmy. Efficient resource discovery for large scale ad hoc networks using contacts. ACM SIGCOMM Computer Communications Review, 32(3):32, July 2002.

[25] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, pages 151–162, Aug 1999.

[26] K. Obraczka & K. Viswanath, *Flooding for Reliable Multicast in Mutlti-Hop Ad Hoc Networks*, PARSEC Workshop'99

[27] S. Paul, K. K. Sabnani, J.C. Lin, S. Bhattacharyya. Reliable Multicast Transport Protocol (RMTP). In IEEE Journal on Selected Areas in Communications, Vol. 15 No. 3, Pages 407-421, April 1997

[28] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 90–100, New Orleans, LA, Feb. 1999.

[29] Larry Peterson and Bruce Davie. Computer Networks A Systems Approach, Morgan Kaufman Inc., 2003. ISBN 1-55860-832-X.

[30] J.Pu, E.Manning, G.Shoja, Routing Reliability Analysis of Partially Disjoint Paths, In *Proceedings of PACRIM'01*, Victoria, 2001.

[31] Abhishek Rajgarhia, Fred Stann, and John Heidemann. Privacy-Sensitive Monitoring With a Mix of IR Sensors and Cameras. In *Proceedings of the Second International Workshop on Sensor and Actor Network Protocols and Applications*, pages 21-29, Boston, August 2004

[32] Y. Sasson, D. Cavin, A. Schiper. Probabilistic Broadcast for Flooding in Wireless Mobile Ad hoc Networks. IEEE Wireless Communications and Networking Conference (WCNC) - March 2003

[33] Fred Stann and John Heidemann. RMST: Reliable Data Transport in Sensor Networks. In *Proceedings of the First IEEE Intl. Workshop on Sensor Network Protocols and Applications*, pages 102-112, Alaska, May 2003.

[34] Fred Stann, and John Heidemann. BARD: Bayesian-Assisted Resource Discovery In Sensor Networks. In Proceedings of the 24th IEEE INFOCOM Conference. Miami, Florida, USA. March, 2005.

[35] Kumar Viswanath, and Katia Obraczka. Modeling the Performance of Flooding in MultiHop Ad Hoc Networks, *Symposium on Performance Evaluation of Computer and Telecommunication Systems* (SPECTS'04)

[36] C. Wan, A. Campbell, L. Krishnahmurthy. PSFQ: A Reliable Transport Mechanism for Wireless Sensor Networks. ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, Georgia, Sept 2002.

[37] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D. Cutler. Exploiting the Capture Effect for Collision Detection and Recovery. *The Second IEEE Workshop on Embedded Networked Sensors (EmNetS-II)*. Sydney, Australia. May 30-31, 2005.

[38] B. Williams, and T. Camp. Comparason of Broadcasting Techniques for Mobile Ad Hoc Networks. In *Proceedings of MOBIHOC'02,* Lausanne, Switzerland, 2002.

[39] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *Proc. First International Conference on Embedded Networked Sensor System (SenSys 01),* , pages 14-27. Los Angeles, 2003.

[40] J. Zhao, and R. Govindan. Understanding Packet Delivery Performance in Dense Wireless Sensor Networks. In *Proc. First International Conference on Embedded Networked Sensor System (SenSys 01),* pages 1-13. Los Angeles, 2003.

[41] F. Zhao, J. Shin, and J. Reich. Information-Driven Dynamic Sensor Collaboration for Tracking Applications. In *IEEE Signal Processing Magazine*, 19(2):61-72, March 2002