Technical report

Smart Gossip: Infusing Adaptivity into Gossiping Protocols for Sensor Networks*

Pradeep Kyasanur+ Romit Roy Choudhury Indranil Gupta

Department of Computer Science, University of Illinois at Urbana-Champaign
Email: {kyasanur,croy}@uiuc.edu, indy@cs.uiuc.edu

***Abstract*— Probabilistic techniques have been used to address many challenges in sensor networks. However, little work exists on developing adaptive versions of probabilistic protocols. In this paper, we consider the class of probabilistic gossiping protocols that are useful for disseminating information. Information dissemination is often required in sensor networks for code updates, TAG-type queries, etc. We propose adaptive techniques that enable a gossip-based protocol to automatically and dynamically adapt itself to the network topology. Our techniques are capable of coping with wireless losses and unpredictable node failures that affect network connectivity over time. The adaptive techniques also allow the sensor network applications to specify a desired reliability for disseminating messages. Nodes automatically adapt their behavior to satisfy such requirements. The resulting protocol is completely decentralized. We present thorough experimental results to evaluate our "Smart Gossip" proposal, and demonstrate its benefits over existing gossip protocols.**

***Index Terms*— Sensor networks, information dissemination, gossiping, probabilistic techniques**

I. Introduction

In sensor networks, broadcast (i.e., one-to-all communication) is an important service primitive for disseminating information. Broadcast can be used by sink nodes for disseminating TAG-type queries, code updates, alarms, etc. Flooding is a simple solution for supporting broadcast, where each node is required to forward every packet it receives. However, the resource constraints of sensor nodes prohibit solutions that rely on flooding, since flooding leads to the *broadcast storm problem* [1], which in turn depletes energy at the sensor nodes. In contrast, *probabilistic broadcast protocols* avoid the broadcast storm problem, and are simple, energy-efficient, alternatives to flooding. In addition, sensor network applications may also require broadcast protocols that support different degrees of reliability. Probabilistic protocols can satisfy such application requirements as well, while trading off energy consumption with reliability, as required.

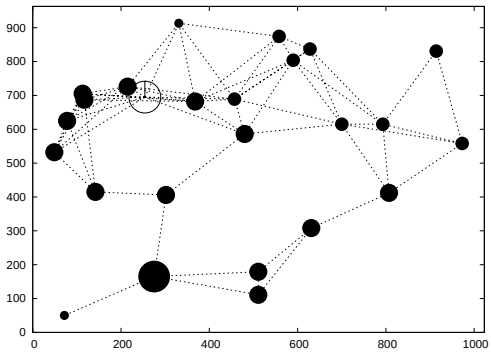


Fig. 1. Adapting gossip probability based on topology: The radius of each node is proportional to its gossip probability. The unshaded circle denotes the gossip's source node.

One instance of a probabilistic protocol is *gossiping*, where each node forwards a packet probabilistically. While there exist several gossiping protocols, e.g., [2], [3], [4], [5], they are predominantly *static* in nature, i.e., they require the gossip probability to be pre-configured, and cannot adapt to the changing topology and the changing application requirements. Therefore, static protocols may require the network designer to conservatively pre-configure the gossip parameters, on a case-by-case basis, to allow for changes to network topology and application requirements. A few protocols, particularly [2], [4], propose adapting gossip probability based on node degree or the number of duplicate broadcasts. However, as demonstrated later, such heuristics may not be sufficient in several network scenarios.

*This research was supported in part by Vodafone and Motorola graduate fellowships, and by NSF grants ITR CMS-0427089 and CAREER CNS-04448246.

Canonical gossip protocol and the case for adaptivity: The canonical static gossiping protocol for multihop networks works as follows. Each node forwards the packet with some probability $p \leq 1$. It has been shown that there exists a probability $p_{threshold}$, such that if every node transmits with this probability $p = p_{threshold}$, then almost all the nodes receive the disseminated information. If $p_{threshold}$ for some network topology proves to be 0.7, then gossip protocols can clearly outperform flooding (which uses $p = 1$) in terms of overhead, and yet retain the load-balancing properties, lacking in the optimized variants of flooding. However, a problem arises here. *What should be the correct value of $p_{threshold}$ if information about the network topology is not available a priori?*

Existing gossip-based algorithms mostly assume that the value of p is globally assigned to each node during network initialization. Such practices may be appropriate in many applications, such as in peer-to-peer networks, where by design, the structure of the network topology is often known (e.g., all nodes may be designed to have roughly the same number of neighbors). However, in sensor networks that are formed by scattering sensors over an open area, it is unclear how a global value of p can be pre-specified. *Moreover, in the presence of topological irregularities, a global choice of p may not be suitable.* For example, Figure 1 depicts a typical topology with randomly generated node locations (this may correspond to randomly scattering sensors), and we can see that the node density varies in different parts of the network. Where the topology is sparse, a higher value of p might be necessary to achieve gossip percolation. Where the topology is dense, a lower value of p may suffice. If a single gossip probability p is used for the whole network, then p has to be conservatively chosen to ensure gossip percolation in sparser areas of the network, leading to unnecessary gossip messages being sent in the denser parts of the network.

Figure 1 depicts the gossip probabilities that were automatically chosen by the proposed "Smart Gossip" protocol. In our protocol, each node uses information about dependencies on other nodes (and not just simple information about its own degree) to decide how to set its gossip probability. Observe that with our protocol, dense regions in the network are generally characterized by lower gossip probabilities, and vice versa. As a special case, notice that for a leaf node (degree=1), its neighbor transmits with high probability (e.g., consider the two nodes in the lower left corner of Figure 1).

In addition to adapting to topology, different applications may require different reliability guarantees. For instance, a code dissemination application may require 99% of the packets to be delivered correctly to all nodes, while 75% may suffice for an application broadcasting sensing instructions. Moreover, the reliability requirements may need to be met even under node failures and wireless losses. Choosing global, conservative values *a priori* will be sub-optimal under such stiff requirements. In contrast, the proposed smart gossip protocol can adapt to application-specified requirements as well.

To summarize our contributions in this paper, we have proposed a smart gossip protocol by infusing adaptivity into gossiping. The "Smart Gossip" protocol has the following innovative capabilities:

1) The protocol can automatically adapt to different topologies, precluding the need for case-by-case pre-configuration.
2) The protocol is capable of achieving application-specified reliability requirements, even in the face of wireless losses and node failures, while incurring low overheads.
3) The protocol is light-weight in view of the limited CPU and energy resources available at sensors.

The rest of this paper is organized as follows. We present related work in Section II. We formulate the gossip problem in Section III, and outline the basic smart gossip protocol in Section IV. Section V describes extensions to smart gossip protocol for handling wireless losses and node failures. Section VI presents evaluation of smart gossip. We discuss future extensions to smart gossip in Section VII, and conclude in Section VIII.

II. Related Work

Probabilistic techniques have been employed in building robust and scalable distributed systems. One example of a probabilistic technique is *gossiping*. Demers et al. [6] were among the first to demonstrate the benefits of gossip-based protocols for distributed systems. More recently, gossip has been used in wired networks [7], peer-to-peer networks [8], mobile ad hoc networks [9], [3], [10], [11] and sensor networks [12], [4], [13], [5]. Probabilistic techniques have been used in other network protocols, for example, in mobile agent protocols [14], [15], [16].

Many past research proposals on gossip have implicitly or explicitly incorporated features for adaptive gossiping. The basic gossip mechanism proposed by

Demers et al. [6] can be characterized as *uniform gossiping*, where a node may forward the gossip message to any other node in the network with equal probability. One adaptation mechanism proposed to improve over uniform gossip is *spatial gossiping* [17], [18], wherein "nearby" nodes are chosen with higher probability as the destination of a gossip message. In wireless networks it may be expensive (if at all feasible) to directly forward a gossip message to a node far away in the network. Therefore, typical wireless network solutions use an extreme case of spatial gossip, wherein the gossip message is sent to only the neighboring nodes with some non-zero probability, and further away nodes with probability zero. Our proposal may be viewed as adding adaptation to spatial gossiping.

Another adaptation mechanism proposed in literature improves gossip by using network information. There are many proposals for wired networks that use network information for adapting gossiping [19], [20], [21]. Some wireless network specific techniques for reducing gossip overhead are proposed in [2], [4], [22]. Adaptive techniques have also been used to restrict the scope of gossip to a desired region of the network [23]. This scoped gossiping technique can be used in conjunction with our proposal.

III. Gossip Problem Formulation

We consider a wireless network with N nodes. Our goal is to provide a network-wide broadcast service for efficiently sending multiple broadcast messages. Such a broadcast service can be used by a sink node to send periodic query messages, code updates involving multiple packets, etc. We define "gossip originators" to be nodes that utilize the network-wide broadcast service (e.g., a set of sink nodes).

Each gossip message[1] initiated by a gossip originator is marked with a sequence number. When a non-initiator node X receives a gossip message originated by a node O with sequence number k, node X forwards the gossip message with some probability $p_X(O,k)$.

The gossip problem is to choose the probabilities $p_X(O,k)$ such that the desired application requirements are achieved, while incurring minimal overheads. To evaluate reliability, we propose a metric called *Average Reception Percentage*. Reception percentage of a node X, with respect to a gossip originator O, is the percentage of messages originated at O that are received at X. *Average reception percentage* is the reception percentage, averaged over all nodes in the network. Overhead is evaluated by a metric, *Average Forwarding Percentage*. The forwarding percentage of a node X, with respect to a gossip originator O, is the percentage of gossip messages from O that is forwarded by X. *Average forwarding percentage* is the forwarding percentage, averaged over all nodes in the network. Observe that the *average forwarding percentage* also proves to be a measure of the average energy consumed at a node in transmitting gossip messages. As a consequence, reduction in *average forwarding percentage* will also lead to reduction in energy consumption, thereby increasing the lifetime of sensor networks.

The gossip problem can be formally defined as *choosing the gossip probabilities $p_X(O,k)$ to meet the application-specified average reception percentage, while minimizing the average forwarding percentage*. Other desirable properties of a gossip algorithm include correct and efficient operation under changing network topologies, without requiring case-by-case protocol tuning.

A. Distinguishing our protocol from existing gossip strategies

We show that existing variants of gossip protocols are classes of the broader gossip problem formulated above. We also state precisely how our protocol is different from the existing protocols.

- *Static Gossip:* In this strategy, all nodes choose the same gossip probability p for all gossip packets, i.e., for all gossip originators O and all nodes X, $p_X(O,k) = p$. Typically, the gossip probability p is assigned during network initialization.
- *Adaptive Gossip:* In this strategy, all nodes do not use a single gossip probability. Adaptation may be on a per originator basis (a node X chooses some probability $p_X(O)$ independent of packet k), or may be on a per packet basis ($p_X(O,k)$ is recomputed at each node X for every packet k).

While most of the existing gossip strategies are static [2], [3], [5], there are some solutions that adapt gossiping. Haas et al. [2] have proposed an adaptive gossip protocol, wherein a node chooses its gossip probability based on the number of neighbors it has. We designate this strategy as the "Adaptive Neighbor" approach for

[1] A message may be designated as a gossip message by setting a flag in the packet header. In certain scenarios, when only one or a small number of packets have to be disseminated, a non-adaptive solution might suffice. In such scenarios, the adaptive protocol can be bypassed, by turning off the flag.

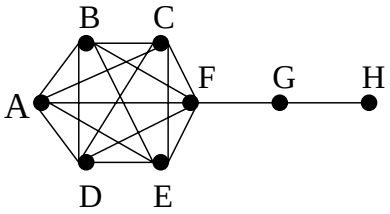


Fig. 2. Toy example motivating the need for a smarter adaptive gossip protocol

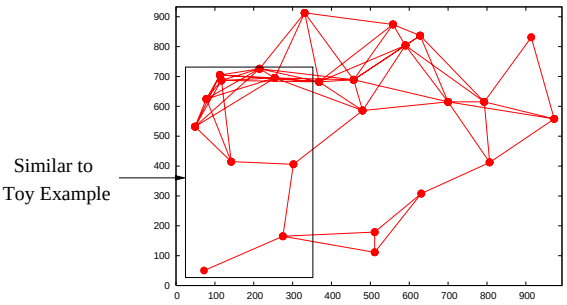


Fig. 3. A sample topology generated by random placement of nodes

reference in later comparisons. Levis et al. [4] have proposed a gossip protocol where a node chooses its gossip probability for a message with sequence number k, based on the number of duplicate messages that were overheard for message $k-1$. We designate this strategy as the "Adaptive Overheard" approach. In the following section, we discuss why the above adaptations may not be sufficient. Our proposal presents an improved adaptation strategy, which is based on additional topological characteristics, to provide an efficient framework for information dissemination in resource-constrained, failure prone, sensor networks. The benefits of our approach over earlier works are demonstrated in Section VI.

B. Need for adaptive gossip

We point out the need for adaptive gossiping by analyzing a toy topology in Figure 2. Although the topology is artificially created, it is representative of parts of random topologies. For example, Figure 3 represents a sample topology generated by random placement of 25 nodes (this may correspond to randomly scattering 25 sensor nodes). Observe how the portion of the topology, demarcated by the labeled box, is similar to our toy topology in having a dense cluster of nodes attached to a less dense group of nodes. Put differently, we argue that realistic topologies are often characterized by heterogeneous degrees of connectivity, and therefore lend themselves to adaptive gossip[2], as explained next.

Consider the case of static gossiping on the topology in Figure 2 – suppose that the problem is to choose a global probability such that the gossip always reaches every node. Clearly, if a gossip is initiated by node A, it reaches node H only if nodes F and G forward the gossip with probability 1. The global gossip probability must therefore be $p = 1$. This choice of probability will force nodes B, C, D and E to unnecessarily transmit the gossip. This may result in a large *average forwarding percentage*. Moreover, observe that the global value of p was estimated based on knowledge of the topology. In the absence of this knowledge, the problem is even more difficult, and in most cases would lead to either over conservative estimates resulting in high overheads, or aggressive estimates leading to low reception percentage.

The *adaptive neighbor* and *adaptive overheard* strategies may reduce the gossip overhead by reducing unnecessary transmissions, but may fail to propagate the gossip correctly. For example, in Figure 2, node F must gossip with probability 1 for nodes G and H to receive the gossip. However, since F has a large number of neighbors, F will choose a small gossip probability when using the adaptive neighbor metric. Even when using the *adaptive overheard* technique, F will do the same because it overhears multiple duplicate packets from its neighbors. Choosing small gossip probabilities at node F will degrade *average reception percentage* for the network. Moreover, if there was a large cluster of nodes on the right of node H (not shown in the figure), then a large number of nodes would fail to receive the gossip, significantly degrading reliability.

The existing adaptations to gossip are not effective primarily because *a node chooses its gossip probability independent of how many other nodes depend on this node* for reception of a gossip message. For example, if node F can identify that node G depends only on F to receive the gossip, while if nodes B, C, D, and E can identify that they are never required to forward the gossip, then we can achieve both efficiency and reliability. This is the key intuition of our approach, as elaborated in the next section.

Our proposed strategy takes into account the originator of gossip as well, as different originators may specify different *average reception percentages*, while previous

[2]Evaluation results in Section VI justify the need for adaptive gossip.

solutions do not offer such flexibility. Past empirical studies [24], [25] have noted the need for an adaptive gossip solution that adapts to wireless losses and changes in network topology. Drawing on this need, our proposed strategy aims to provide robustness against wireless losses and topology changes.

IV. Protocol Description

We begin with a high level description of our approach. Thereafter, we present our protocol incrementally, arguing at each step why enhancements to the basic design are necessary.

A. The Intuition

On analyzing the behavior of gossip percolation, we observed the notion of *dependence*. To be precise, we observed that a node X depends on a subset of its neighbors, defined as *parents of X, Parent(X)*, to receive a new gossip message. Similarly, a disjoint subset of X's neighbors, defined as *children of X, Child(X)*, depend on X to receive this same gossip message. It is also possible that node X does not depend on some of its neighbors, and neither do those neighbors depend on X, to receive the gossip message – these neighbors are defined as *sibling of X, Sibling(X)*. Observe that the parent, child, and sibling relationships are only logical.

In view of such dependences, forming *dependency graphs* might appear to be a good solution. However, we realized that unlike well understood *dependency graphs*, the nature of dependence in gossip percolation is probabilistic. In other words, node X does not depend on any particular parent, Y, to receive the gossip. Instead, it depends on a group of nodes, expecting at least one member of this group to probabilistically deliver the gossip to it. Intuitively, the gossip probabilities chosen by each node should be a function of this group size. If X has a large number of parents, it may suffice for each of its parents to choose a small value of p_{gossip}, and the vice versa. As a result, where the topology is denser, nodes would choose lower gossip probabilities than where the topology is sparser. The main idea of our protocol is pivoted on this observation. However, to leverage benefits from this idea, the protocol first needs to efficiently identify the group of parents. The details are discussed next.

B. Design of smart gossip protocol

We use a light-weight design, based on promiscuous overhearing of broadcast messages (all gossip packets are broadcast), to minimize the overheads of our protocol. Typically, all nodes are required to decode broadcast messages, and therefore promiscuous overhearing does not impose any additional cost. However, if promiscuous hearing feature is not available, then our protocol can be easily modified to infrequently exchange explicit control messages. We also assume that all links are bi-directional. If underlying links are not bi-directional, then our protocol can be extended to ignore packets arriving over a unidirectional link, while computing dependencies. Specifically, a node X will include another node Y in any of its dependency relations only if X and Y can mutually communicate.

Nodes extract information from overheard messages, and by applying simple rules, attempt to deduce whether the sender of the message is a parent, child, or a sibling. It is possible for a node to have multiple parents, siblings, and children. Once the parent set is identified, the child node can calculate the probability with which it thinks its parents are required to transmit. A child i, announces this required probability, $p^i_{required}$, by piggybacking it on every gossip it forwards. A parent node overhears such announcements from its (potentially) multiple children, determines the maximum of the announced values, and thereafter assigns its own gossip probability as $p_{gossip} = max(p^i_{required})$. Channel fluctuations, node failures, and other faults impact the parent-child dependencies, which automatically translate into re-assignments of forwarding probabilities. For example, if one of two parents fail, then the child is left with a single parent; in such cases the child announces a higher value of $p_{required}$ to ensure that its single parent transmits with a much higher probability. In addition, we optimize our protocol to recover from lost packets. We quantify protocol performance using simulations in Section VI.

Identifying Parent-Sibling-Child Relationships:

The simplest means of identifying parents would be to record the sender of the overheard gossip and call it a parent. However, this is not sufficient because when a child forwards the packet, the parent may overhear this packet and incorrectly identify the child as a parent. This problem can be resolved by requiring nodes to include its own parent's identifier in the packet[3]. On receiving a packet from node X, node Y checks if X's parent is either Y or one among Y's parents. If Parent(X) is Y,

[3]In our basic version of the protocol, when a node discovers multiple parents, it includes only the first identified parent in the packet. We outline a scheme in which a node includes a randomly chosen parent in each of its outgoing packet later in Section VII.

then Y adds X as its child. If Parent(X) is one among Y's parents, then Y adds X as its sibling (i.e., Sibling$(Y) = \{X\}$). Failing both these conditions, Y adds X as its parent. Applying the rules to Figure 4, we get

Child$(A) = \{B, X\}$
Parent$(B) = \{A\}$, Sibling$(B) = \{X\}$, Child$(B) = \{C\}$
Parent$(X) = \{A\}$, Sibling$(X) = \{B\}$
Parent$(C) = \{B\}$

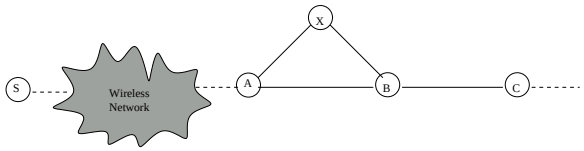


Fig. 4. An example topology with node S at the gossip originator. For node C to receive the gossip, nodes A and B must always forward the gossip. If links are reliable, then node X need not forward the gossip.

Since nodes B, X and C have a single parent, they all would announce $p_{required} = 1$ implying that nodes A and B must always transmit. Also, since node X does not have a child, it need not transmit at all[4]. With respect to this topology, this is exactly what we intend to achieve.

In view of more complex scenarios, we need another refinement. Consider receiving a packet from a node whose parent belongs to the set of siblings. This happens in Figure 5, when node B receives a gossip from node Y (assume that Y specifies X as its parent). Ideally, node Y should not be a parent of B because both Y and B receive packets from the same node A. This condition means that B might be able to deliver gossips to Y because Y is a child of its sibling. The analogy of a human family is relevant here wherein the young can rely on its parent as well as its parent's siblings. Therefore, in this context, node B adds Y to its list of children.

The final refinement follows easily from the above discussion. On receiving a packet from a node whose parent is a child, the node is added to the list of children. In Figure 5, this results in node C being B's child even if C specifies Y as its parent. The pseudo-code for our protocol is presented in Figure 6.

C. Analysis: Meeting Application-specified Reliability

The application that utilizes this underlying gossip primitive specifies its reliability requirement as an *average reception percentage*, τ_{arp}. For example, $\tau_{arp} = 90\%$ implies that the application at the gossip source expects each node in the network to receive at least 90 out of 100 packets sent out from the source, with high probability. Now, if we require each node in the network to only ensure that its own children receive at least 90% of the packets that the node itself receives, then observe that nodes multiple hops away from the source would achieve fewer than 90% of the packets disseminated by the source. However, since the application requires even the furthest node to achieve at least τ_{arp}, we translate τ_{arp} into a per-hop reception probability, τ_{rel}. In other words, if each node ensured that its children received τ_{rel} fraction of the packets that the node itself received, then each node in the network would receive τ_{arp} fraction of the packets disseminated by the gossip source. Since each node independently decides whether to forward a packet, τ_{rel} can be estimated by the equation $(\tau_{rel})^{\delta} = \tau_{arp}$, where δ is the estimated diameter of the network.

With this notion of per-hop reliability, a node with a single parent requires its parent to choose a gossip probability greater than τ_{rel}. However, when a node Y

[4]However, to account for probable wireless losses over link A-B, we require node X to transmit with low probability – a trade-off between redundancy and reliability.

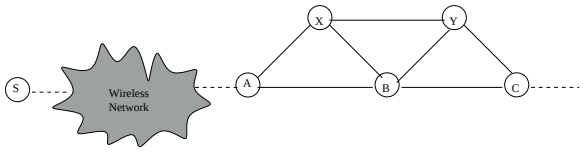


Fig. 5. An example topology in which nodes experience different types of dependencies between each other.

```
Initialization at node i:
NeighborList = i;
SiblingList = i;
ParentList = null;
ChildList = null;

Overheard_Message(fromNode j)
{
 AddToNeighborList(j);
 if(parent(j) not in NeighborList)
  AddToParentList(j);
 else {
  if(parent(j) in ParentList)
   AddToSiblingList(j);
  else
  if(parent(j) in SiblingList)
   AddToChildList(j);
 }
}
```

Fig. 6. The pseudo code for smart gossip.

has K parents, the probability that at least one of them transmits ($p_{required}$) must exceed the per-hop reception probability. The following equation dictates the choice of gossip probability.

$$(1 - p_{required})^K < (1 - \tau_{rel}) \quad (1)$$

In this equation, $(1 - p_{required})$ denotes the probability that a particular parent does not transmit. Assuming that parents independently decide whether to transmit or not, the probability that all K parents choose not to transmit can be calculated as $(1 - p_{required})^K$. This probability needs to be less than the failure probability, denoted by $(1 - \tau_{rel})$. Equation 1 is derived from these set of observations. Now, since node Y is capable of determining the value of $p_{required}$ from this equation, it includes this value of $p_{required}$ within every gossip packet it sends out. A node X that has included Y in its list of children must forward the gossip with at least the probability specified by Y. More precisely, node X chooses the maximum of probabilities specified by all its children, thereby meeting the per-hop reception probability at each of them.

Static gossip protocols have been previously analyzed by applying percolation theory (e.g., [2], [5]). Analytical results have shown that if all nodes in the network gossip with a probability higher than some threshold, then most nodes in the network receive the gossip message. However, to the best of our knowledge, these analytical techniques assume that all nodes use a common gossip probability, and it is intractable to apply these techniques to our adaptive approach, where different nodes may use different gossip probabilities. However, as we discussed above, our protocol carefully selects gossip probability to ensure application-specified reliability is met. Detailed evaluations in Section VI empirically validate our claim that smart gossip protocol suitably adapts to the network topology to ensure gossip percolation. The results also show that the gossip probability chosen by each node quickly converges to a stable value.

The proposed smart gossip protocol requires each node to maintain $O(G)$ state, where G is the number of active gossip originators. As discussed earlier, gossip originators are typically expected to be sink nodes (and applications running on them), which is expected to be small with respect to total nodes (N) in the network. Therefore, the storage requirements of the proposed smart gossip protocol will be fairly small. Furthermore, with increasing memory available on newer generation sensor nodes, storage capabilities are not expected to be a constraint.

D. Example operation of smart gossip

We refer back to the toy example in Figure 2 and explain how our protocol can adapt to the topology. To determine connectivity at each node, our protocol requires every node to forward the first gossip packet with probability 1. Assuming node H as the initiator of the gossip[5], nodes G and F identify their dependences as follows. Parent$(G) = \{H\}$, Child$(G) = \{F\}$, Sibling$(G) = \{\}$, and Parent$(F) = \{G\}$, Child$(F) = \{A, B, C, D, E\}$, Sibling$(F) = \{\}$. Since node F has a single parent, using equation (1), it specifies $p_{required} = \tau_{rel}$. Similarly, since node G has a single child, F, it assigns $p_{gossip} = \tau_{rel}$ as required by F. Clearly, node F achieves gossip reliability of τ_{rel}.

Nodes belonging to the cluster identify their dependences, and by virtue of being directly connected to F, all of them designate F as their parent. In addition, each node in the cluster adds the other nodes to its list of siblings. We enumerate the dependences of only node A here; Parent$(A) = \{F\}$, Child$(A) = \{\}$, Sibling$(A) = \{B, C, D, E\}$. The overall dependences have been shown in Figure 7. Notice that the nodes in the cluster (except node F) do not have children, and therefore do not forward the packet after the first round of gossip. However, since each of them only has a single parent, they all request F to transmit the gossip with probability $p_{required} = \tau_{rel}$. This ensures that the gossip percolates through the sparse portion of the network, and yet is not redundantly transmitted where the topology is denser.

V. Resilience to wireless losses and failures

Empirical studies from the past have emphasized the need for considering wireless losses while designing a communication protocol. With gossiping, wherein packets are transmitted using MAC broadcasts (which are typically not retransmitted by the MAC layer), the need to handle wireless losses is even more relevant. The algorithm presented earlier, for estimating the gossip probabilities needed to meet application-specified reliability requirements, does not account for wireless losses. When the wireless loss rate is high, a probabilistic approach based on gossip may not be sufficient. In view of this, we propose the following refinement that adds some *determinism to the gossip protocol*.

[5]The protocol works correctly when node A is the originator as well.

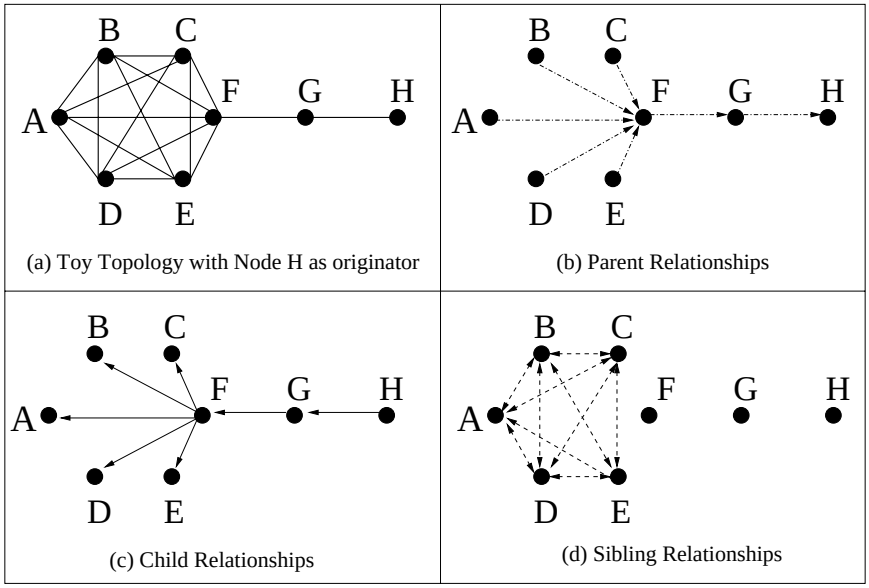


Fig. 7. Dependence relationships for toy topology from Figure 2, with node H as gossip originator. (a) The toy topology, (b) Arrows pointing from child to parent, (c) Arrows pointing from parent to child, (d) Arrows pointing between siblings.

Every node in the network tracks the sequence number of the packets that it receives. If a node X finds a missing sequence number(s), it selects one of its parents, and explicitly requests it to retransmit the missing packet(s). Missing sequence numbers are identified when X has received all packets up to a sequence number j, and then receives a packet with sequence number $(j+k)$, $k > 1$.

Observe that it is possible that the children of X may also have not received the packets that X did not receive (for example, if X is the only parent). We have to ensure that the children (and their children and so on) do not initiate a retransmission request while X is still in the process of obtaining the lost packet. Otherwise, we may have a *retransmission request storm*. To prevent child nodes from initiating a request, X piggybacks the value $(j+1)$ (the smallest missing sequence number) when forwarding the $(j+k)^{th}$ packet. Dependents of X deduce that X itself is trying to recover from the $(j+1)^{th}$ packet, and thereby do not initiate explicit recovery requests themselves. If the parent of X provides X with the missing packets, then X forwards those packets downstream. X sets a timeout after initiating any retransmission request. If the missing packet cannot be recovered before the timeout expires, then the packet is deemed to be permanently lost. The children of X, eventually do the same as well. In our simulations, we have evaluated our protocol with at most one retransmission attempt, since a large number of retransmissions may significantly increase the energy consumption.

A parent of X can retransmit a lost packet only if it has a stored copy of the requested packet. We require each node to maintain a small buffer of recently received packets per gossip originator. A node cannot satisfy a retransmit request if the requested packet is not in its buffer. Sensor nodes may have memory constraints that preclude the usage of large buffers. However, our simulation results show that even when using a buffer size of only 5 packets per originator, up to 20% packet losses can be tolerated. To tolerate higher loss rates, buffer sizes can be increased. Thus, the size of the packet buffer, and the number of retransmission attempts, are knobs that can be adjusted to trade-off between resources and robustness.

Wireless sensor networks may also exhibit high node failure rate. Node failures may arise out of battery draining out, harsh environmental conditions, etc. As a result, sufficient redundancy in the number of nodes is typically included in a sensor network deployment. On account of node failures, the network topology evolves over time. Initially, when most nodes in the sensor network are alive, a lower gossip probability suffices. When fewer nodes are alive later in the network lifetime, it is important to use higher gossip probabilities for meeting the application-specified reliability requirement. The proposed adaptive gossip protocol is suitable for operation in this setting with a few modifications as described below.

When node failures are prevalent, the dependency lists maintained by a node have to be periodically updated

to remove entries corresponding to failed nodes. For example, when a node X fails, neighbors of X need to remove X from their dependency lists. We implement the removal of failed nodes by associating a timeout with each entry in the dependency lists. When no messages have been received from a node for a duration more than the specified timeout, entries associated with the node are removed from the dependency lists. Once dependency lists are pruned of old entries, the adaptive gossip protocol updates the required gossip probabilities, and new parent-child relationships are eventually built, if necessary. Thus, the smart gossip algorithm can adapt to changes in topology brought about by node failures.

The choice of timeout to be used depends on the frequency of node failures, and is a design parameter. If node failures are infrequent, a large timeout value suffices. On the other hand, frequent node failure will necessitate a small timeout. By choosing appropriate timeouts, our proposal can effectively handle node failures. If node failures are excessive, it may result in the network being partitioned. In that case, application-specified reliability requirements may not be met (but, any other dissemination approach will also fail to meet the reliability requirement).

VI. Performance Evaluation

In this section, we evaluate the performance of the proposed "Smart Gossip" protocol, and compare it with "Static Gossip", "Adaptive Overheard", "Adaptive Neighbor" approaches (see Section III-A for details of each approach). All protocols have been implemented in Qualnet simulator, version 3.7 [26]. The protocols have been implemented at the application layer of the simulator, and gossip packets are transmitted on the wireless channel using MAC broadcasts (we use IEEE 802.11 MAC protocol model in Qualnet for ease of evaluation). Since sensor networks may use alternate MAC protocols, and gossiping is a technique that is not inherently tied to any specific MAC protocol, we use the following steps to avoid the side-effects that may arise out of using IEEE 802.11 MAC.

First, since 802.11 uses a CSMA/CA approach with the possibility of packet collisions (while an alternate TDMA approach may not have any collisions), we introduce sufficient jitter between packet transmissions to reduce the probability of collision. Second, each round of gossip (a round starts when the gossip originator sends a gossip message) is separated by large intervals of 1 second, which ensures gossip message of two different rounds do not collide with each other. By using these techniques, we have separated out the MAC effects from the gossip protocol performance, and we have verified in all our simulations that there are no MAC collisions. However, we do account for the impact of wireless losses (which can be viewed as arising out of collisions as well) by randomly dropping gossip packets based on the specified packet error probability. Node failures are similarly simulated by turning off a node when it is supposed to fail based on the failure model. It is part of our future work to evaluate the smart gossip protocol using a real sensor network.

In most of this section, we present results for 100 randomly chosen topologies (occasionally we present graphs with randomly selected 5 topologies for clearer presentation). Unless otherwise specified, each topology has 50 nodes. Each node has a transmission range of 280m (default range in Qualnet). For achieving reasonable node densities, the 50 nodes are placed in a square of side 1000m (realistic sensor networks may have smaller transmission ranges and proportionately smaller area of deployment). We also present some results for topologies having 1000 nodes placed in a square of 3000m to simulate large sensor networks. The position of nodes in the square are generated uniformly at random. We have also evaluated the protocols for grid and chain topologies, but we do not include them here for lack of space. Recall that the metrics for evaluation, *average reception percentage* and *average forwarding percentage*, were defined earlier in Section III. Unless otherwise specified, the target application-specified *average reception percentage* is set to 90% (i.e., the application can tolerate at most 10% message losses). One node in the network is randomly chosen to be the gossip originator. The gossip originator sends a total of 150 gossip messages.

We first justify the need for a new adaptive protocol by studying the performance of static gossip and existing adaptive gossip protocols in Section VI-A. We then study the performance of basic smart gossip protocol in Section VI-B, and evaluate the extensions to handle wireless losses and node failures in the rest of the section.

A. Need for an adaptive gossip protocol

We first study the performance of static gossip. Recall that in the static gossip approach, every node in the network uses a common global gossip probability. We consider two models of static gossip, based on the mechanism used to select the global gossip probability.

Static gossip "with adaptation": Under the first model, designated as static gossip "with adaptation", we assume that *for each topology* (i), the minimum gossip probability (P^i_{topo}) that meets the application-specified reliability requirement is used as the global gossip probability for that topology. Therefore, this approach uses *different global gossip probabilities for different topologies*. Hence, the "with adaptation" results may be viewed as the minimum overheads that may be possible with static gossip when the best global gossip probability is selected by an omniscient protocol on a per-topology basis.

Static gossip "without adaptation": Under the second model, designated as static gossip "without adaptation", we assume that a *single global gossip probability* (P_{global}) is used for all (100) topologies such that application-specified reliability is met for all topologies ($P_{global} = \max(P^i_{topo})$). Therefore, this approach chooses the minimum gossip probability that works for all topologies. This models the scenario wherein the global gossip probability is pre-configured at protocol design time without full knowledge of the network topology. The difference between the overheads of the two versions of static gossip may be viewed as the benefits obtained by static gossip using *characteristics of different network topologies*. In addition to these benefits, the proposed adaptive gossip protocol can also benefit by adapting to *density variations within a specific topology.*

For each topology, we measured the application-specified reliability for gossip probabilities increasing from 0 to 1 (in steps of size 0.05 for tractability). The metrics for a topology under static gossip "with adaptation" was measured using that gossip probability (P^i_{topo}) which met the application-specified requirements for *that* topology. The metrics for a topology under static gossip "without adaptation" was measured using the common gossip probability (P_{global}) that met the application-specified requirements for *all* (100) topologies under consideration.

Figure 8 plots the global gossip probability required to meet 99% and 90% application-specified reliability for static gossip "with adaptation", over 100 random topologies. As we can see from the figure, different topologies require widely different gossip probabilities for achieving the application-specified reception percentage. Furthermore, for any given topology, different application-specified reliability requires different gossip probabilities. Hence, we see that the appropriate gossip probability for the network depends on the *network topology and application-specified reliability requirement*.

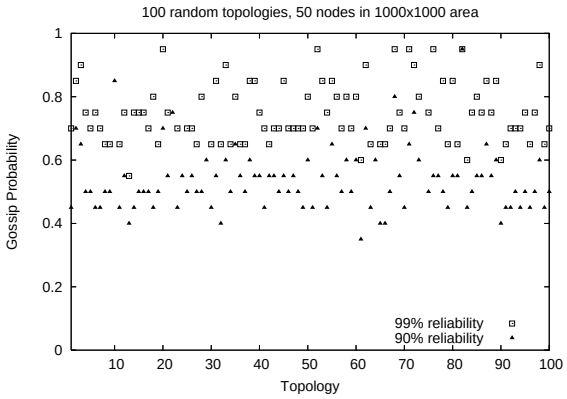


Fig. 8. Average reception percentage for static gossip with adaptation: Different topologies require widely different gossip probabilities to meet the application-specified reliability.

Under static gossip "without adaptation", we will have to choose a single gossip probability for all possible topologies. We can see from Figure 8 that there are some topologies which require gossip probability close to 1, although there are other topologies for which gossip probability close to 0.5 will suffice. Hence, under the "without adaptation" model, all topologies will have to use a gossip probability close to 1. Therefore, if the network designer is not aware of the exact structure of the network topology, then a conservative estimate of the gossip probability that suffices for all topologies will be close to 1.

Figure 9 plots the overhead, measured as the *average forwarding percentage*, for different global gossip probabilities, for a randomly selected set of 5 topologies. The main observation from the figure is that *gossip overhead increases linearly with the gossip probability*, and therefore using a conservatively chosen gossip probability (that is close to 1) may result in a significant higher overhead in many topologies (where a value closer to 0.5 may suffice). Hence, it is desirable that the gossip protocol adapt to the network topology.

The above results suggest that finding the appropriate gossip probability with static gossip requires knowledge of the network topology. However, even if full network knowledge is available, static gossip may have higher overheads than smart gossip because static gossip uses a single common gossip probability for the whole network. Smart gossip can adapt to the heterogeneity in node distribution in any given topology to further reduce overheads (see Section VI-B for further details).

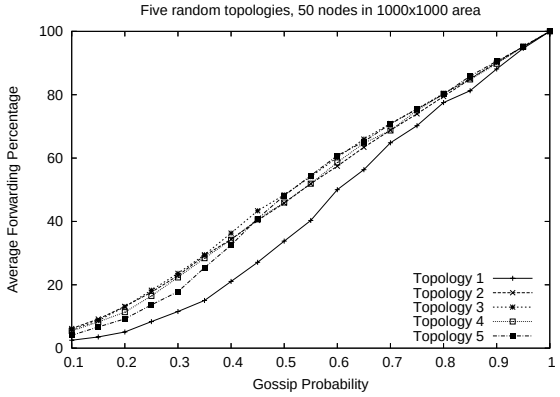


Fig. 9. Average forwarding percentage for static gossip: The overhead of static gossip increases linearly with gossip probability.

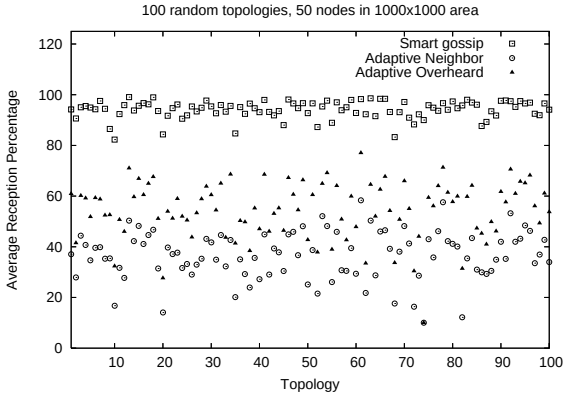


Fig. 10. Comparison of reception percentage for different gossip strategies: Smart gossip successfully meets application-specified reliability, while Adaptive neighbor and Adaptive overheard strategies are unsuccessful.

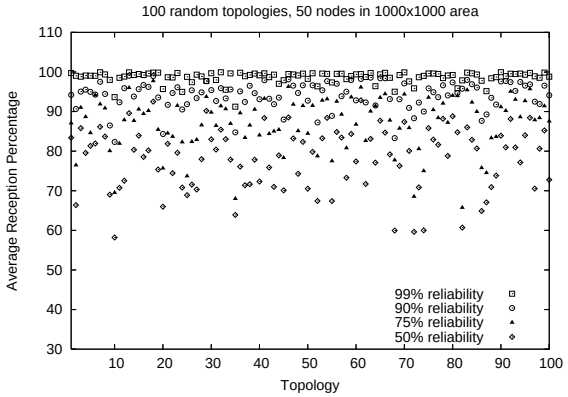


Fig. 11. Reception percentage with different application-specified reliability requirements: In almost all cases, smart gossip successfully achieves the application-specified reliability.

We now compare the proposed smart gossip protocol with two different adaptive gossip approaches proposed in the literature - "Adaptive Neighbor" and "Adaptive Overheard" (see Section III-A for a description of these approaches). Figure 10 plots the *average reception percentage* for smart gossip and the different adaptive gossip strategies. We assume that the target reception percentage is 90%. "Adaptive Neighbor" and "Adaptive Overheard" are not aware of the notion of application-specified reliability - hence, we assume that these protocols attempt to achieve as high a reliability as possible. As we can see from the figure, our proposed solution meets the target reception percentage, while other adaptive strategies achieve widely different reliability with different topologies, all below the desired reception percentage. It may be possible to improve the performance of "Adaptive Neighbor" and "Adaptive Overhead" by *carefully tuning* these adaptive protocols, but published literature does not describe how the tuning has to be performed. Furthermore, even if a procedure for tuning the protocol is available, such a tuning may have to be done on a *case-by-case basis depending on the topology*. This highlights the need for an adaptive protocol to be self-tuning.

We will next evaluate the performance of the proposed smart gossip protocol.

B. Evaluation of proposed smart gossip protocol

1) Adapting to application-specified requirements: In this section, we evaluate the ability of the proposed smart gossip strategy to adapt the gossip probabilities based on the application-specified reliability requirements. We compare the overheads of smart gossip with static gossip (under both the "with adaptation" and "without adaptation" models).

Figure 11 plots the *average reception percentage* achieved for different application-specified reliability requirements over 100 random topologies. As we can see from the figure, smart gossip protocol meets the application-specified reliability requirement in almost all cases. In fact, the achieved reliability is often higher than the desired reliability. Since the smart gossip algorithm chooses parent gossip probability to meet the requirements of *every child*, it may result in higher than desired reliability at some children. We have chosen such a conservative approach to ensure reliability requirements are met with high probability.

Table I compares the average overhead of smart gossip and static gossip over 100 random topologies with different application-specified reliability requirements. As we can see from Table I, the overhead of smart gossip reduces when lower application-specified reliability suffices. This indicates that by using an adaptive approach, significant reduction in overheads can be obtained when lower reliability suffices.

Application-specified Reliability	Smart Gossip	Static Gossip (with adaptation)	Static gossip (without adaptation)
99%	68.3%	74.9%	88.9%
90%	53.1%	51.6%	72.7%
75%	42.9%	34.2%	62.7%
50%	33.1%	15.8%	38.5%

TABLE I

OVERHEAD COMPARISON OVER 100 RANDOM TOPOLOGIES HAVING 50 NODES.

Furthermore, we can see from Table I that smart gossip has overheads comparable to static gossip "with adaptation". Since the "with adaption" approach corresponds to the use of an omniscient protocol, the results imply that smart gossip is fairly successful in adapting to the network topology and application-requirements (while using a distributed algorithm). Only when the application-specified reliability is small, smart gossip has higher overheads than static gossip with adaptation. Note that smart gossip conservatively chooses gossip probabilities, as explained earlier. Also, smart gossip requires all nodes that are deemed to be "child" nodes to gossip with a non-zero probability to ensure neighboring parent nodes are aware of the presence of child nodes. Such extra gossip messages have a larger contribution to the overhead when application reliability requirements are small (and small gossip probabilities are used by parent nodes as well). As a result, when a lower application-specified reliability suffices, we are not optimally reducing the overheads, and it is part of our future work to better handle this scenario. However, existing gossip approaches do not support application-specified reliability at all, and cannot adapt to application requirements. Furthermore, smart gossip has significantly lower overhead than static gossip "without adaptation" (up to 20% in many cases). This substantiates our claim that *significant performance improvement is possible by adapting to the network topology.*

We next evaluate the performance of smart gossip in large networks having 1000 nodes. These results were generated by porting the Qualnet implementation into a custom simulator, to allow evaluation of large networks.

From Figure 12, we can see that even in large networks, smart gossip meets the application-specified reliability requirement in almost all cases. Therefore, these results imply that even in large networks, smart gossip adaptively chooses gossip probabilities that ensure percolation of gossip messages throughout the network.

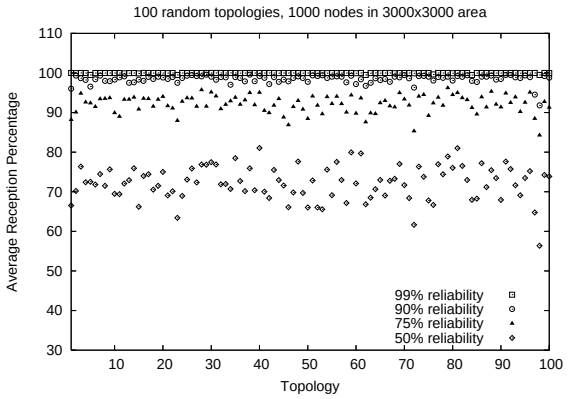


Fig. 12. Reception percentage with different application-specified reliability requirements: In all cases, Smart gossip successfully achieves the application-specified reliability.

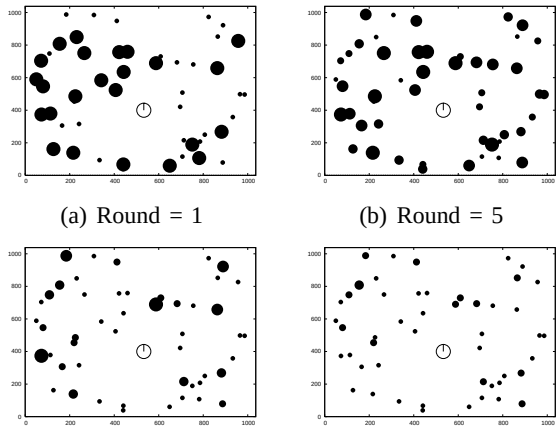
Application-specified Reliability	Smart Gossip	Static Gossip (with adaptation)	Static gossip (without adaptation)
99%	40.28%	66.29%	95%
90%	31.00%	44.44%	80.01%
75%	25.29%	34.55%	65.02%
50%	17.73%	25.75%	50.02%

TABLE II

OVERHEAD COMPARISON OVER 100 RANDOM TOPOLOGIES HAVING 1000 NODES.

Table II compares the average overhead of smart gossip and static gossip with different application-specified reliability requirements. Comparing with the results in Table I (which was for 50 node networks), we see that the overhead of *both* smart gossip and static gossip is lower in larger networks. Therefore, as the network size and density increases, gossiping techniques have a significantly lower cost than flooding.

Another key observation from Table II is that smart gossip has significantly lower overhead than static gossip "with adaptation" in larger networks for all application-specified reliability requirements (in small networks, the overhead reduction is less significant). This is because, as the network becomes larger, the variation in node density across different parts of the network may become larger. As a result, static gossip (by choosing a common global gossip probability) performs significantly worse than the proposed smart gossip. In addition, static gossip "without adaptation" has extremely high overheads in large networks because there are some worst case topologies which require a high gossip value. When this high gossip value is used in all topologies, it results in a significantly higher cost.



(a) Round = 1 (b) Round = 5

(c) Round = 10 (d) Round = 20

Fig. 13. The gossip probability chosen by a node is proportional to its size. The unshaded circle is the gossip originator. Gossip probability evolves with time and quickly converges at each node to the minimum probability that meets application-specified reliability.

2) Adapting to network topology: Figure 13 depicts the adaptation of gossip probabilities with gossip rounds[6] for a sample topology with 50 nodes (topology 1 in earlier figures). In the figure, the gossip probability chosen by a node is proportional to the size of the circle. Initially, nodes that deem themselves to be parents gossip with probability 1. Child nodes gossip with a very small probability. Over time, the probabilities chosen evolve, and after a few iterations, most nodes converge toward the minimum gossip probability that suffices in meeting application-specified reliability requirement. Figure 14 plots the change in the overhead (average forwarding percentage) per gossip, with gossip rounds, for the sample topology. As we can see from the figure, the overheads quickly converge to an average value (within 15 rounds). Our results have shown that in all topologies, the gossip probabilities at each node, and hence the overheads, quickly converge to a stable value. Thus, smart gossip is *fairly responsive in adapting* to the network topology.

C. Impact of wireless losses

Wireless medium is inherently unreliable. As a result, some of the gossip messages may be lost. In this section, we evaluate the ability of the retransmission strategy used in the smart gossip protocol to recover from wireless losses. We vary the degree of wireless losses by varying the packet loss probability. We set the buffer size at each node for supporting retransmissions to 5 packets. Figure 15 plots the *average reception percentage* for different packet loss probabilities ranging

[6]Each round consists of one packet sent from the gossip originator.

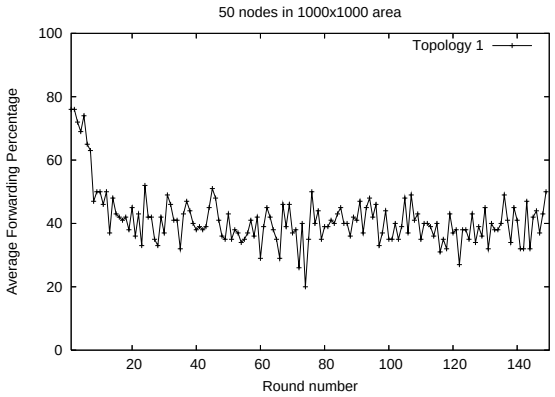


Fig. 14. Variation of overhead with gossip rounds: The overhead is initially high when the gossip probability is evolving, but quickly converges to a stable value.

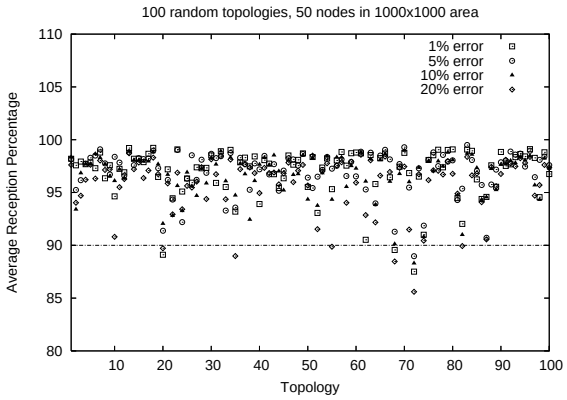


Fig. 15. Average reception percentage for adaptive gossip under link errors: Application-specified reliability is achieved in most cases even in the face of wireless losses.

from 1% to 20%. The proposed retransmission strategy is quite successful in recovering from wireless losses even with fairly high loss probabilities (a 20% packet loss rate is quite high even for wireless networks). Only in few scenarios, the proposed protocol fails to meet the application-specified reliability. In our simulations, a missing gossip packet is explicitly requested for only once. With a high error probability, it may be necessary to use multiple retransmission attempts in some scenarios.

Table III presents the average overhead of gossip protocol for different packet loss probabilities over 100 topologies. The overhead measurement accounts for retransmitted packets and packets sent to explicitly request for retransmission. As we can see from the figure, the overhead is almost constant across different loss probabilities, with a slight increase in the overhead as the packet loss probability increases. When the loss rate is low, the redundancy of gossip often masks an occasional packet loss, thereby not requiring explicit

Packet loss probability	Overhead
1%	56.57%
5%	56.78%
10%	57.34%
20%	58.49%

TABLE III

OVERHEAD COMPARISON OVER 100 RANDOM TOPOLOGIES WITH DIFFERENT PACKET LOSS PROBABILITIES.

retransmission in most cases. Only when the packet loss probability increases, explicit retransmission may become necessary. In summary, smart gossip protocol *can handle small packet error probabilities with minimal increase in overheads*.

D. Impact of node failures

Wireless sensor networks may exhibit a non-negligible rate of node failure, on account of harsh environmental conditions, battery drainage, etc. Consequently, such networks are often initially setup with a sufficiently high node density to cope with node failures. In this section, we evaluate the ability of the proposed gossip protocol to adapt to the varying node density brought about by node failures. The reception percentage of a gossip packet sent in a round is now measured with respect to the number of nodes that were alive at the beginning of the round. The *average reception percentage* is computed as the average of the reception percentages over all rounds. The *average forwarding percentage* is similarly measured based on the nodes alive at the beginning of each round. The results presented here are over 100 random topologies, each topology initially having 100 working nodes (we start off with a higher node density compared to the earlier scenarios that had 50 nodes). Over the duration of simulation, randomly selected nodes fail. The number of nodes that fail are varied from 0% (no failures) to 50% (half the nodes fail). We assume that the target reception percentage is 90%.

Figure 16 plots the *average reception percentage* for different node failure rates. As we can see from the figure, even with a fairly high node failure rate (25%), our protocol meets the application-specified reliability requirement in most cases. Only with a large node failure rate (50%), the reliability requirement is not met. When there is a high node failure rate, the network is often separated into multiple disjoint partitions. Therefore, nodes that are not in the same partition as the gossip originator fail to receive the gossip, reducing the measured reception percentage.

Table IV presents the overhead of gossip protocol with

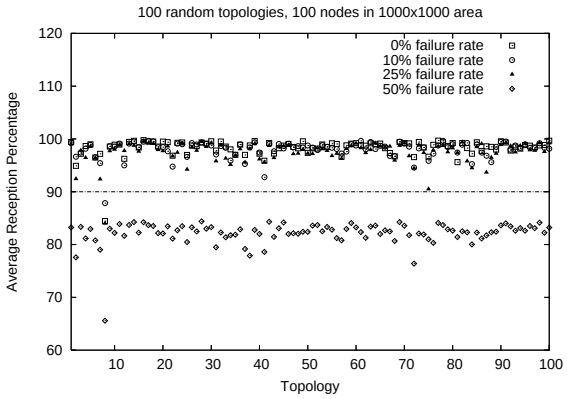


Fig. 16. Average reception percentage for adaptive gossip under node failures: Application-specified reliability is achieved in the face of node failures, except when failures are very high (50%).

Node failure rate	Overhead
0%	38.95%
10%	40.18%
25%	41.42%
50%	36.69%

TABLE IV

OVERHEAD COMPARISON OVER 100 RANDOM TOPOLOGIES WITH DIFFERENT NODE FAILURE RATES.

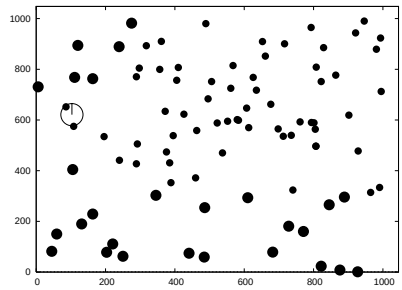
different node failure rates. When the node failure rate is low (implying high node density), it suffices for each node to forward fewer messages on an average, leading to lower overheads (the overheads in this experiment are lower than overheads in Table I because of higher node density). When the node failure rate is high, there is a greater burden on the surviving nodes to forward messages, leading to higher overhead. Paradoxically, the overheads with 50% failure rate is the lowest. This is because, at that high node failure rate, only a few nodes (belonging to the partition containing gossip originator) forward the gossip, though overheads are measured over all surviving nodes, leading to low overheads (as well as a low reception percentage).

Figure 17 depicts the adaptation of gossip probabilities when node density changes with time. In the figure, the gossip probability chosen by a node is proportional to the size of the circle. When the node density is initially high, most nodes use low gossip probability. When node density reduces with time, higher gossip probability is required at nodes that are sparsely connected.

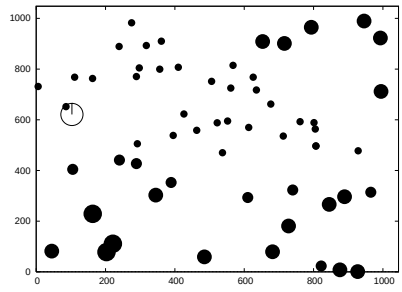
E. Summary of results

The main findings from the performance evaluations are:

- Existing gossip approaches cannot adapt to network topology and application requirements (Section VI-



(a) Time = 25s, around 95 nodes are alive



(b) Time = 130s, around 55 nodes are alive

Fig. 17. The gossip probability chosen by a node is proportional to its size. The unshaded circle is the gossip originator. When node density is high most nodes use small gossip probabilities, but larger probabilities are needed when node density reduces (e.g., compare the top right portion of the two figures).

A).

- Proposed smart gossip protocol automatically adapts to wide range of topologies, network sizes and application requirements (Section VI-B). The protocol can automatically reduce overheads when lower reliability suffices.
- The gossip probabilities chosen by smart gossip protocol quickly converge to a stable value (Section VI-B), and moderate rate of link errors and node failures are tolerated with minimal increase in overheads (Section VI-C and Section VI-D).

VII. Discussion and Future Work

The key contribution of this paper is a gossiping algorithm that can adapt to different topologies, while meeting application-specified reliability requirements. Often, there is a trade-off between meeting reliability and reducing overheads, and in our current proposal we have chosen to prioritize meeting application-specified reliability over reducing overheads. However, we have identified several optimizations that may reduce the overheads incurred, and it is part of our future work to evaluate the effectiveness of the optimizations. Below, we discuss some optimizations for reducing overheads.

In our smart-gossip proposal, a node specifies $p_{required}$ to its parent, based on the number of parents that it has. However if this node underestimates the number of parents that it has, it is possible that it specifies a higher probability to each of its parents. For example, in Figure 5, node X may have a disjoint path from the source to it (not shown in the figure). However, assuming node X receives the gossip from node A first, it announces that A is its parent. By our common-parent condition, node B would not consider X as its parent. However, notice that X might be able to deliver a gossip to B even if A does not. In such a situation, A would always transmit a packet, leading to redundancy. To resolve such issues, we intend to modify the parent announcement scheme. A node that discovers multiple parents, announces each of its parents in some schedule (say, round robin). In our example, if node X has two parents, A and W (not shown in figure), then B will quickly discover that X is its parent. This will happen when X announces W as its parent. Thereafter, A will not be required to transmit always, reducing on message overhead.

Another possibility would be to reduce overheads dynamically. Nodes can record the number of overheard messages, and on receiving too many duplicates, can suggest a lower probability to its parents. This approach can be integrated with the explicit retransmission strategy to build a mechanism that is reminiscent of the feedback control used in TCP. When a node looses a gossip message, it increases the gossip probability provided to its parents, and recovers from the lost packet by sending an explicit retransmission request. When a node has received multiple packets without any loss, and has overheard many duplicate messages as well, it reduces the gossip probability provided to its parents. Therefore, this optimization can allow fine-grained adaptation to the underlying topology, thereby further reducing message overheads.

VIII. Conclusion

This paper presents a smart gossip protocol, designed to offer reliable broadcast services to resource-constrained sensor networks. We propose adaptive techniques that allow a gossip protocol to automatically and dynamically adapt itself to any network topology. Simulation results have demonstrated smart gossip to be better than existing adaptive techniques in adapting to unknown network topologies. Smart gossip also adapts to application-specified reliability requirements. Our results indicate that smart gossip successfully achieves application reliability requirements, and significantly reduces

overheads when lower degree of reliability is sufficient. Smart gossip proves to be resilient to wireless losses and node failures that may be quite prevalent in wireless sensor networks. Our results have shown that moderate degree of wireless losses and node failures is successfully tolerated by the smart gossip protocol.

REFERENCES

[1] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu, "The broadcast storm problem in a mobile ad hoc network," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. 1999, pp. 151–162, ACM Press.

[2] Zygmunt J. Haas, Joseph Y. Halpern, and Li Li, "Gossip-Based Ad Hoc Routing," in *Infocom*, 2002.

[3] Ranveer Chandra, Venugopalan Ramasubramanian, and Kenneth P. Birman, "Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks," in *ICDCS*, 2001.

[4] Philip Levis, Neil Patel, David Culler, and Scott Shenker, "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," in *Symposium on Network System Design and Implementation (NSDI' 04)*, 2004.

[5] M. Miller, C. Sengul, and I. Gupta, "Exploring the Energy-Latency Tradeoff for Broadcasts in Energy-Saving Sensor Networks," in *ICDCS*, 2005.

[6] A. Demers, D. Greene, C. Hauser, W. Irish, and J. Larson, "Epidemic Algorithms for Replicated Database Maintenance," in *Proceedings of ACM Symposium on Principles of Distributed Computing*, 1987.

[7] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, "Bimodal Multicast," *ACM Transactions on Computer Systems*, vol. 17, no. 2, pp. 41–48, May 1999.

[8] I. Gupta, A. Kermannec, and A. Ganesh, "Efficient Epidemic-style Protocols for Reliable and Scalable Multicast," in *Proceedings of the 21st Symposium on Reliable and Distributed Systems (SRDS)*, October.

[9] A. Vahdat and D. Becker, "Epidemic Routing for Partially Connected Ad Hoc Networks," Tech. Rep., Dept. of Computer Science, Duke University, October 2000.

[10] J. Luo, P. Eugster, and J. Haubaux, "Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks," in *Proceedings of Infocom*, 2003.

[11] Anwitaman Datta, Silvia Quarteroni, and Karl Aberer, "Autonomous Gossiping: A self-organizing epidemic algorithm for selective information dissemination in mobile ad-hoc networks," in *International Conference on Semantics of a Networked World (IC-SNW'04)*, June 2004.

[12] David Kempe, Alin Dobra, and Johannes Gehrke, "Gossip-Based Computation of Aggregate Information," in *IEEE Symposium on Foundations of Computer Science (FOCS'03)*, 2003.

[13] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip Algorithms: Design, Analysis and Applications," in *Proceedings of INFOCOM*, March 2005.

[14] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz, "Ant-based load balancing in telecommunications networks," in *Adaptive Behavior, vol 5, issue 2, p.169-207*, 1996.

[15] N. Malpani, Y. Chen, N. Vaidya, and J. Welch, "Distributed Token Circulation on Mobile Ad Hoc Networks," in *IEEE Transactions on Mobile Computing*, 2004.

[16] R. Roy Choudhury, S. Bandyopadhyay, and K. Paul, "MARP: A Multi-Agent Routing Protocol for Mobile Wireless Ad Hoc Architectures," in *Accepted for publication in the Journal of Autonomous Agents and Multi-Agent Systems*. Kluwer Academic Publishers.

[17] David Kempe, Jon Kleinberg, and Alan Demers, "Spatial Gossip and Resource Location Protocols," in *Symposium on Theory of Computing (STOC)*, 2001.

[18] David Kempe and Jon Kleinberg, "Protocols and Impossibility Results for Gossip-Based Communication Mechanisms," in *IEEE Symposium on Foundations of Computer Science (FOCS'02)*, 2002.

[19] M. Lin and K. Marzullo, "Directional Gossip: Gossip in a Wide Area Network," in *Proceedings of European Dependable Computing Conference*, 2000.

[20] L. Rodrigues and J. Pereira, "Self-Adapting Epidemic Broadcast Algorithms," in *FuDiCo II: S.O.S. Survivability: Obstacles and Solutions 2nd Bertinoro Workshop on Future Directions in Distributed Computing*, June 2004.

[21] L. Rodrigues, S. Handurukande, J. Pereira, R. Guerraoui, and A. Kermarrec, "Adaptive Gossip-Based Broadcast," in *Proceedings of Dependable Systems and Networks (DSN)*, 2003.

[22] Donald J. Scott and Alec Yasinsac, "Dynamic Probabilistic Rebroadcast in Ad hoc Networks," in *International Conference on Wireless Networks*, 2004.

[23] Xiang-Yang Li, Kousha Moaveninejad, and Ophir Frieder, "Regional Gossip Routing for Wireless Ad Hoc Networks," in *IEEE International Conference on Local Computer Networks (LCN03)*, 2003.

[24] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "An Empirical Study of Epidemic Algorithms in Large Scale Multihop Wireless Networks," in *UCLA Computer Science Technical Report UCLA/CSD-TR 02-0013*, 2002.

[25] Yoav Sasson, David Cavin, and Andre Schiper, "Probabilistic Broadcast for Flooding in Wireless Mobile Ad hoc Networks," Tech. Rep. IC/2002/54, Swiss Federal Institute of Technology (EPFL), 2002.

[26] Scalable Network Technologies, "Qualnet simulator version 3.7," www.scalable-networks.com.