Smart Gossip: An Adaptive Gossip-based Broadcasting Service 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*— A network-wide broadcast service is often used for information dissemination in sensor networks. Sensor networks are typically energy-constrained and prone to failures. In view of these constraints, the broadcast service should minimize energy consumption by reducing redundant transmissions, and be tolerant to frequent node and link failures. We propose "Smart Gossip", a probabilistic protocol that offers a broadcast service with low overheads. Smart gossip automatically and dynamically adapts transmission probabilities based on the underlying network topology. The protocol is capable of coping with wireless losses and unpredictable node failures that affect network connectivity over time. The resulting protocol is completely decentralized. We present thorough experimental results to evaluate our "Smart Gossip" proposal, and demonstrate its benefits over existing protocols.**

I. Introduction

Several sensor network applications rely heavily on a *network-wide broadcast service* for disseminating information. For example, sink nodes may often need to broadcast code-updates, TAG-type queries, alarms, etc., to sensor nodes. Flooding, considered as the simplest means of broadcasting, does not prove to be applicable in the context of sensor networks. This is because flooding leads to collisions and redundant packet receptions [1] that together deplete sensors of valuable battery power. An ideal solution for sensor networks needs to be one that delivers a single copy of the broadcast packet to each sensor, using minimum number of transmissions. There have been several approaches that aspire toward this goal, broadly classified into deterministic and probabilistic approaches.

Several deterministic approaches have aimed to minimize redundant communications by identifying a suitable subset of nodes, and assigning them the responsibility of forwarding messages. The *deterministic subset* of nodes may either be cluster-heads in the network graph [2], multi-point relays [3], or members of a backbone tree [4] – whatever the name, they essentially build a connected dominating set of the graph, D_G, and use it to disseminate information. However, this class of solutions suffer from two main problems. First, if a member of D_G fails, sensors that depend on the failed member may not receive the broadcast message. This affects the reliability of the network, and requires expensive mechanisms to maintain D_G over time. Second, using only members of D_G to disseminate information affects the load-balancing properties of the network, and drains the battery at some sensors much earlier than others. Sensor network applications often desire uniform energy consumption across nodes, and therefore, strict hierarchical assignment of responsibility may not be suitable.

Probabilistic broadcast approaches, broadly called *gossip*, offer a simpler alternative to deterministic approaches. With gossiping, nodes in the network are required to forward packets with a pre-specified probability, $p_{gossip} \leq 1$. The key idea is that when p_{gossip} is chosen correctly, the entire network receives the broadcast message with very high probability, even though only a *non-deterministic subset* of nodes have forwarded the message. Gossiping is a simple solution, yet capable of achieving better reliability and load-balancing. However, choosing the correct value of p_{gossip} is a difficult problem. Observe that the correct value of p_{gossip} is closely associated with the topology of the network, and in the absence of topology information, estimating the value of p_{gossip} is risky. Moreover, the topology of a sensor network evolves over time due to sensor failures and link errors, and therefore, a suitably chosen p_{gossip} may become sub-optimal over the network lifetime. Clearly, pre-assigning sensors with a common gossip probability can be inefficient.

Our aim in this paper is to obtain the benefits of

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

both the above approaches, while incurring minimal overheads. Put differently, we intend to retain the benefits of forming an efficient dissemination backbone (like dominating sets), and yet have the flexibility to achieve load-balancing and reliability characteristics (like gossip).

In *Smart Gossip*, we quantify the "importance" of each node in achieving dissemination. The importance of a node, X, increases when other nodes heavily *depend* on X to receive a disseminated message. In such a case, we require X to transmit with a proportionally higher probability. Other nodes that are less crucial for achieving dissemination still transmit for purposes of reliability, but with a lower probability. As detailed later, *Smart Gossip* quantifies the importance of a node using a distributed algorithm that takes into account the local topological properties around a node. For example, sensors use relatively smaller gossip probabilities in regions where sensor density is higher, and vice versa. As a result, the sensor network adapts itself to the changing topology. Figure 1 shows an example topology on which *Smart Gossip* has chosen gossip probabilities based on the "importance" of nodes.

Unlike dominating sets, where a unique subset of nodes are assigned responsibilities, *Smart Gossip* achieves dissemination by distributing responsibility among multiple subsets. As a result, load is balanced among multiple nodes. Also, sharing dissemination responsibilities leads to better fault-tolerance properties.
Unlike existing gossiping techniques, *Smart Gossip* assigns different gossip probabilities to different nodes, based on the topology that the sensors have formed after deployment. The probabilities are revised periodically in view of the potential changes in topology (due to node failures and channel outages). The features of *Smart Gossip* can be summarized as follows:
(1) Smart gossip can adapt to different topologies, precluding the need for pre-deployment configuration.
(2) The protocol is capable of achieving desired reliability requirements, even in the face of wireless losses and node failures.
(3) The protocol is distributed and light-weight, an important requirement for resource-constrained sensor networks.

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

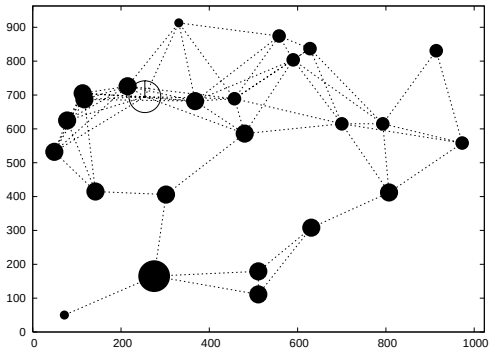

Fig. 1. Adapting gossip probability based on topology: The radius of each node is proportional to its gossip probability. The unshaded circle denotes the gossip's source node.

II. Gossip Problem Formulation

We consider a wireless network with N nodes, where node locations may be unknown. 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 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. The desired reception percentage is specified by the sensor applications.

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 protocol tuning.

Based on whether all nodes use the same gossip probability, gossip protocols can be characterized as *static* or *adaptive* approaches.

A. Static gossip strategy

In this strategy, all nodes choose the same gossip probability p for all gossip packets. Typically, the common gossip probability p is assigned to all sensors during network deployment. Most of the gossip protocols described in literature are static (c.f., [5], [6], [7]).

Static gossip is unsuitable for wireless sensor networks because of several reasons. In many sensor deployments, the network topology is not known before deployment (e.g., when deploying sensors by dropping them from an aircraft). Without the knowledge of topology, the gossip probability each node uses has to be conservatively set to a sufficiently high value to ensure reliable dissemination in all possible topologies. However, such a high gossip probability may be unnecessary for most topologies, and therefore, results in high overheads. Even if the network topology can be predicted *a priori* with controlled deployment, different parts of the sensor network may have different node densities. Since denser regions can achieve dissemination using a relatively smaller value of p, assigning a common conservative gossip probability across the whole network is unnecessary.

Static gossip uses a *single* gossip probability for the whole network, and this gossip probability has to be chosen to be large enough to ensure reliable gossip even in sparse parts of the network. However, such a high probability is not necessary in dense parts of the network, resulting in unnecessary transmissions.

Over time, some nodes in the sensor network may fail. Therefore, even if the gossip probability was chosen correctly initially, the probability has to be updated over time. Similarly, wireless link losses may vary with time, and gossip probability may have to be modified to account for losses as well. From this discussion, it is clear that static gossip is not sufficient for sensor networks.

B. Adaptive gossip strategy

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). The proposed *smart gossip* protocol adapts the gossip probabilities of each node on a per originator basis (the probabilities may also be modified over time).

There are a few adaptive gossip approaches proposed for wireless networks. Haas et al. [5] have proposed an adaptive gossip protocol, wherein a node chooses its gossip probability in inverse proportion to the number of neighbors it has. We designate this strategy as the "Adaptive Neighbor" approach for reference in later comparisons. Levis et al. [8] have proposed a gossip protocol where a node chooses its gossip probability for a message with sequence number k, in inverse proportion to the number of duplicate messages that were overheard for message $k-1$. We designate this strategy as the "Adaptive Overheard" approach.

We identify the shortcomings of existing adaptive strategies by analyzing a toy topology in Figure 2. Although the topology is artificially created, it is representative of heterogeneity in realistic topologies. In the figure, suppose that we need to choose gossip probabilities at each node such that the gossip messages reach 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. To eliminate redundant transmissions, nodes B, C, D, and E need not forward the gossip messages at all. In this scenario, the *adaptive neighbor* and *adaptive overheard* strategies fail to choose the appropriate gossip probabilities.

When using the *adaptive neighbor* approach, since F has a large number of neighbors, F will choose a small gossip probability. However, unless F gossips with probability 1, nodes G and H will not receive some messages. When using the *adaptive overheard* technique, F will again choose a small gossip probability as it may overhear multiple duplicate packets from its neighbors.

The existing adaptive approaches 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, and if nodes B, C, D, and

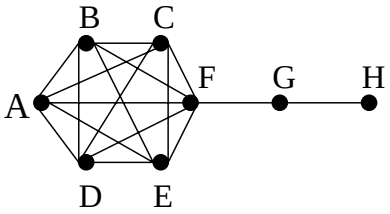

Fig. 2. Example motivating the need for *smart gossip*.

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.

Smart gossip takes into account the originator of gossip as well, as different originators may specify different *average reception percentages*. In addition, smart gossip provides robustness against wireless losses and topology changes. Previous solutions do not offer such flexibility.

III. Protocol Description

We first provide the intuition behind *smart gossip*, and then provide the protocol details.

A. Intuition

On studying the behavior of gossip percolation, we identified certain *dependencies* between nodes. Any 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 *siblings of X, sibling(X)*.

A node X does not require any single parent to forward all gossip messages. Instead, every gossip message has to be forwarded by at least one parent, with different parents possibly forwarding different messages. Therefore, it is possible for parents of a node to forward packets with a probability less than 1, as long as the probability that at least one parent forwards messages is sufficiently high. Hence, the gossip probability, p_{gossip}, which parents of a node X should use is inversely proportional to the number of parents that X has. In addition, some node Y may be the parent of multiple children, and therefore, the gossip probability used by Y should be suitable for all its children.

The main goal of our proposal is to correctly identify the parent, child, sibling sets of each node, and use this information to appropriately select gossip probabilities at each node. In the rest of this section, we present details of *smart gossip*.

B. Design of smart gossip protocol

We use a light-weight design, based on promiscuous overhearing of broadcast messages (all gossip packets are broadcast). Typically, all nodes are required to be awake while broadcast is in progress, 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, but our protocol can be easily extended to handle unidirectional links.

In smart gossip, nodes extract information from overheard gossip messages[1], and by applying simple rules, attempt to deduce whether the sender of the message is a parent, child, or a sibling. The header of each gossip message contains a parent identifier field (pid), and a required gossip probability field ($p_{required}$). When a node forwards a gossip message, it sets the pid field to the identifier of the node from which it received the gossip message.

Initially, when dependencies are not known, each node gossips with probability one. Over time, as nodes learn about their dependencies, the gossip probabilities are refined. Each node computes a gossip probability its parents should use (the calculations are described later), and sets the $p_{required}$ field to this value. Over time, a node remembers the latest $p^i_{required}$ value announced by each child i. The gossip probability used by a node is given by, $p_{gossip} = \max(p^i_{required})$. If a node has no children, it still gossips with a low probability to ensure its parents are aware of the presence of the node.

C. Identifying Parent-Sibling-Child Relationships

A node may designate the sender of a gossip message as its parent. However, this is not sufficient because when a child forwards a packet, its parent may overhear the packet and incorrectly identify the child as a parent. To avoid this, we add an additional check where a node Y, on receiving a packet from node X, checks if X's parent (specified in pid field) is either Y or one among Y's parents. If pid field is set to Y, then Y adds X as its child. If pid field is set to one of 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.

[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.

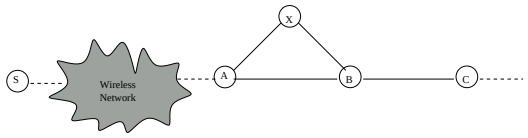

Fig. 3. An example topology with node S at the gossip originator. For node C to receive the gossip, nodes A and B must always forward the gossip. If links are reliable, then node X need not forward the gossip.

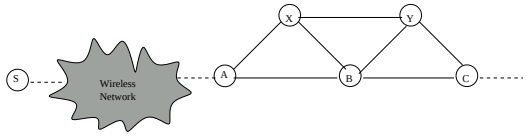

Fig. 4. An example topology in which nodes experience different types of dependencies between each other.

Applying the rules to Figure 3, we get
child$(A) = \{B, X\}$
parent$(B) = \{A\}$, sibling$(B) = \{X\}$, child$(B) = \{C\}$
parent$(X) = \{A\}$, sibling$(X) = \{B\}$
parent$(C) = \{B\}$

In view of more complex scenarios, we need another refinement. Consider a node receiving a packet from another node whose parent belongs to its set of siblings. This happens in Figure 4, 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. Therefore, in this context, node B adds Y to its list of children.

As a final refinement, on receiving a packet from a node whose parent is a child, the node is added to the list of children. In Figure 4, this results in node C being B's child even if C specifies Y as its parent.

To summarize, the protocol operation is as follows. Each node maintains four sets - *NeighborSet*, *ParentSet*, *SiblingSet*, *ChildSet*. During initialization, a node permanently inserts itself into *NeighborSet* and *SiblingSet*, while the other two sets are empty. Whenever the node receives a message from any node X with pid field set to some Y, it uses the following concise rules:

- Add X to *NeighborSet*
- If Y is not in *NeighborSet*, add X to *ParentSet*
- else if Y is in *ParentSet*, add X to *SiblingSet*
- else if Y is in *SiblingSet*, add X to *ChildSet*

Channel fluctuations, node failures, and other faults can impact the parent-child dependencies, and the dependencies are updated over time. More details on handling wireless losses and node failures are in Section IV.

D. Computing gossip probabilities

The application that utilizes smart gossip can specify 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. Note that if only 90% delivery is achieved at each hop, then over multiple hops the delivery ratio will be smaller than 90%. 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 solving 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 at least as large as τ_{rel}. In this case, $p_{required}$ in forwarded gossip messages is set to τ_{rel}. However, when a node Y has K parents, then it suffices for each parent to use a gossip probability ($p_{required}$) which ensures the probability that at least one of them transmits is greater than τ_{rel}. Therefore, the $p_{required}$ value announced by a node with K parents is estimated using the equation:

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

In this equation, $(1 - p_{required})$ is the (upper bound of) 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 is given by $(1-p_{required})^K$, which has to be less than $(1 - \tau_{rel})$ to meet the application reliability requirement.

E. Protocol properties

Smart gossip 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, which is expected to be small with respect to total nodes (N) in the network. Therefore, the storage requirements of smart gossip will be fairly small. Furthermore, with increasing memory available on newer generation sensor nodes, storage capabilities are not expected to be a constraint.

IV. Protocol extensions

A. Handling wireless losses

Empirical studies from the past have emphasized the need for considering wireless losses in sensor networks. 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. When the wireless loss rate is high or bursty, a pure probabilistic approach may not be sufficient. In view of this, we propose the following refinement that adds some *determinism to smart gossip*.

Every new gossip message initiated by a gossip originator includes a unique sequence number. Nodes in the network track the sequence number of received gossip packets. If a node X finds a missing sequence number(s), it explicitly requests one of its parents 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 piggy backs 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. Each node maintains 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.

B. Handling node failures

Wireless sensors are prone to failures. Node failures may arise out of battery draining out, harsh environmental conditions, etc. 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 reliability requirements. Smart gossip 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. 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, parent-child relationships are revised, and gossip probabilities are updated, if necessary. Thus, the smart gossip algorithm can adapt to changes in topology brought about by node failures.

V. Performance Evaluation

We evaluate the performance of the proposed "Smart Gossip" protocol, and compare it with "Static Gossip", "Adaptive Overheard", "Adaptive Neighbor" approaches (see Section II-B for details of each approach) using Qualnet simulator, version 3.7 [9]. Gossip packets are transmitted using MAC broadcasts. We use IEEE 802.11 MAC protocol model in Qualnet for ease of evaluation.

We simulate wireless losses by randomly dropping gossip packets based on a 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 implement smart gossip in sensor nodes.

In most of this section, we present results for 100 randomly chosen topologies. Unless otherwise specified, each topology has 50 nodes. Transmission range of nodes is 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). 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 II. Unless otherwise specified, the target application-specified *average reception percentage* is set to 90%. One node in the network is randomly chosen to be the gossip originator. The gossip originator sends a total of 150 gossip messages.

A. Need for an adaptive gossip protocol

We first present results to motivate the use of adaptive gossip strategy. 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.

Topology-aware static gossip: Under this model 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 topology-aware static gossip 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.

Topology-unaware static gossip: Under this model, 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, smart gossip can also benefit by adapting to *density variations within a specific topology.*

Figure 5 plots the global gossip probability required to meet 99% and 90% application-specified reliability for topology-aware static gossip, over 100 random topologies. As we can see from the figure, different topologies require widely different gossip probabilities

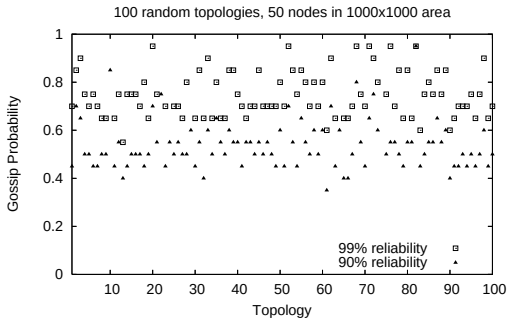

Fig. 5. Average reception percentage for topology-aware static gossip.

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*.

Under topology-unaware static gossip, we will have to choose a single gossip probability for all possible topologies. We can see from Figure 5 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 topology-unaware 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 6 plots the overhead, measured as the *average forwarding percentage*, for different global gossip probabilities, for a randomly selected set of 5 topologies (for clearer presentation, we use only 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.

B. Evaluation of proposed smart gossip protocol

1) Comparison with other adaptive protocols: We compare the smart gossip with two different adaptive gossip approaches proposed in the literature - "Adaptive Neighbor" and "Adaptive Overheard" (see Section II-B for a description of these approaches). Figure 7

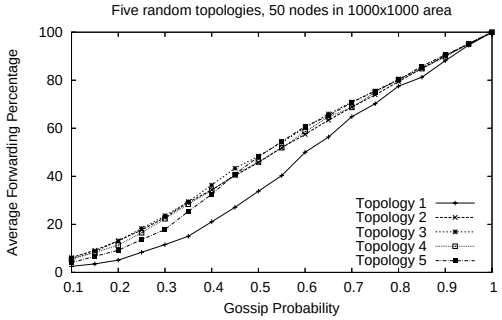

Fig. 6. Average forwarding percentage for static gossip.

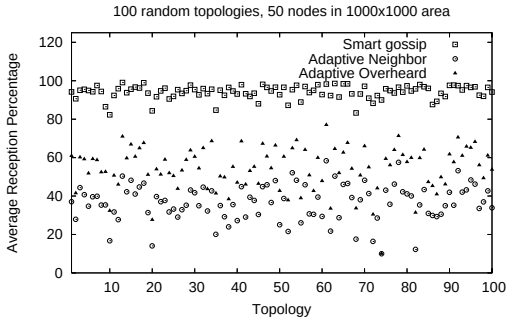

Fig. 7. Average reception percentage for different gossip strategies.

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, smart gossip 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 a self-tuning adaptive protocol.

2) Adapting to application-specified requirements: We evaluate the ability of smart gossip to adapt the gossip probabilities based on the application-specified reliability requirements. We compare the overheads of smart gossip with static gossip.

Figure 8 plots the *average reception percentage* achieved for different application-specified reliability requirements over 100 random topologies. As we can

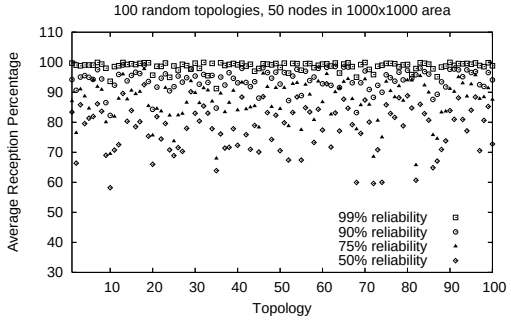

Fig. 8. Average reception percentage with different application-specified reliability requirements.

see from the figure, smart gossip 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 a conservative approach to ensure reliability requirements are met at all nodes 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.

We can see from Table I that smart gossip has overheads comparable to topology-aware static gossip. Since the topology-aware 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 decentralized algorithm). Only when the application-specified reliability is small, smart gossip has higher overheads than topology-aware static gossip. 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.

Smart gossip has significantly lower overhead than topology-unaware static gossip (up to 20% in many cases). In general, smart gossip can significantly improve performance by adapting to the network topology.

Application-specified Reliability	Smart Gossip	Topology-aware Static Gossip	Topology-unaware Static Gossip
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

AVERAGE OVERHEAD IN 50 NODE TOPOLOGIES.

Application-specified Reliability	Smart Gossip	Topology-aware Static Gossip	Topology-unaware static gossip
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

AVERAGE OVERHEAD IN 1000 NODE TOPOLOGIES.

We have also evaluated the performance of smart gossip in larger networks having 1000 nodes. Our evaluation shows that even in large networks, smart gossip can meet the application-specified reliability requirement (figure omitted for lack of space).

Table II compares the average overhead of smart gossip and static gossip with different application-specified reliability requirements for 1000 node networks. 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.

In addition, smart gossip has significantly lower overhead than both topology-aware and topology-unaware static gossip 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 increases. As a result, static gossip (by choosing a common global gossip probability) performs significantly worse than smart gossip.

C. Impact of wireless losses

In this section, we evaluate the ability of smart gossip to recover from wireless losses, by using the proposed retransmission strategy. We vary the degree of wireless losses by varying the packet loss probability from 1% to 20%. We set the buffer size at each node for supporting retransmissions to 5 packets. Figure 9 plots the *average reception percentage* for different packet loss probabilities. 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 a few scenarios, smart gossip 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. Our simulations also show that the increase in overheads because of using a retransmissions is minimal (less than 4% even with 20% wireless losses). For lack of space we refer the reader to [10] for more details.

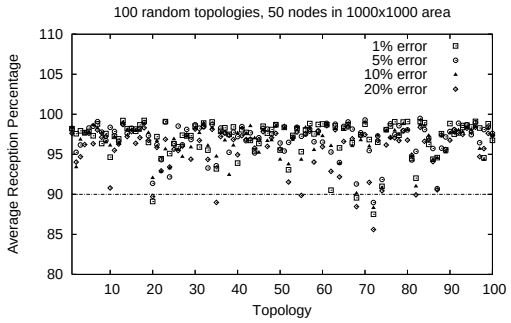

Fig. 9. Average reception percentage under link errors.

D. Impact of node failures

In this section, we evaluate the ability of the smart gossip to adapt to changing node densities on account of node failures. 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).

Figure 10 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.

More detailed simulation results are in an extended technical report [10].

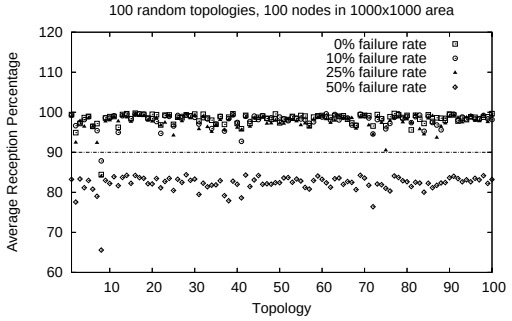

Fig. 10. Average reception percentage under node failures.

VI. Related Work

Probabilistic techniques have been employed in building robust and scalable distributed systems. Demers et al. [11] were among the first to demonstrate the benefits of gossip-based protocols for distributed systems. More recently, gossip has been used in wired networks [12], peer-to-peer networks [13], mobile ad hoc networks [14], [6], [15] and sensor networks [16], [8], [7].

There are many proposals for wired networks that use network information for adapting gossiping (c.f., [17], [18], [19]), but those techniques are not applicable to wireless networks. Some wireless network specific techniques for reducing gossip overhead include the "Adaptive Neighbor" approach [5] and the "Adaptive Overheard" approach [8] which we have shown to be insufficient in sensor networks. Other adaptive gossip work [20] is suited only for uniformly deployed networks. Adaptive techniques have also been used to restrict the scope of gossip to a desired region of the network [21]. This scoped gossiping technique can be used in conjunction with our proposal. Static gossip protocols have been previously analyzed by applying percolation theory (e.g., [5], [7]). However, these analysis techniques are not applicable for adaptive gossip.

VII. Conclusion

This paper presents *smart gossip*, a protocol designed to offer reliable broadcast services to resource-constrained sensor networks. Smart gossip is designed to automatically and dynamically adapts itself to any network topology. Smart gossip also adapts to application-specified reliability requirements. Our results indicate that smart gossip successfully achieves application reliability requirements in all topologies, and significantly reduces overheads when lower degree of reliability is sufficient. Smart gossip is also resilient to wireless losses and node failures that may be quite prevalent in wireless sensor networks.

References

[1] S-Y Ni, Y-C Tseng, Y-S Chen, and J-P Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Mobicom*, 1999.

[2] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks," in *Hawaii International Conference on System Sciences (HICSS02)*, 2001.

[3] A. Qayyum, L. Viennot, , and A. Laouiti, "Multipoint relaying for flooding broadcast messages in mobile wireless networks," in *Hawaii International Conference on System Sciences (HICSS02)*, 2002.

[4] I. Chlamtac and S. Kutten, "Tree-based broadcasting in multihop radio networks," *IEEE Trans. Comput.*, 1987.

[5] Z. J. Haas, J. Y. Halpern, and L. Li, "Gossip-Based Ad Hoc Routing," in *Infocom*, 2002.

[6] R. Chandra, V. Ramasubramanian, and K. P. Birman, "Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks," in *ICDCS*, 2001.

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

[8] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," in *NSDI*, 2004.

[9] Scalable Network Technologies, "Qualnet simulator 3.7," .

[10] P. Kyasanur, R. Roy Choudhury, and I. Gupta, "Smart Gossip: Infusing Adaptivity into Gossiping Protocols for Sensor Networks," Tech. Rep., UIUC, April 2006.

[11] A. Demers, D. Greene, C. Hauser, W. Irish, and J. Larson, "Epidemic Algorithms for Replicated Database Maintenance," in *ACM PODC*, 1987.

[12] 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.

[13] I. Gupta, A. Kermannec, and A. Ganesh, "Efficient Epidemic-style Protocols for Reliable and Scalable Multicast," in *21st Symposium on Reliable and Distributed Systems (SRDS)*, 2002.

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

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

[16] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-Based Computation of Aggregate Information," in *IEEE Symposium on Foundations of Computer Science (FOCS'03)*, 2003.

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

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

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

[20] D. J. Scott and A. Yasinsac, "Dynamic Probabilistic Rebroadcast in Ad hoc Networks," in *International Conference on Wireless Networks*, 2004.

[21] X.-Y Li, K. Moaveninejad, and O. Frieder, "Regional Gossip Routing for Wireless Ad Hoc Networks," in *IEEE International Conference on Local Computer Networks (LCN03)*, 2003.