A Scalable Approach for Reliable Downstream Data Delivery in Wireless Sensor Networks *

Seung-Jong Park, Ramanuja Vedantham, Raghupathy Sivakumar and Ian F. Akyildiz
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
{sjpark,ramv,siva,ian}@ece.gatech.edu

ABSTRACT

There exist several applications of sensor networks where reliability of data delivery can be critical. While the redundancy inherent in a sensor network might increase the degree of reliability, it by no means can provide any guaranteed reliability semantics. In this paper, we consider the problem of reliable sink-to-sensors data delivery. We first identify several fundamental challenges that need to be addressed, and are unique to a wireless sensor network environment. We then propose a scalable framework for reliable downstream data delivery that is specifically designed to both address and leverage the characteristics of a wireless sensor network, while achieving the reliability in an efficient manner. Through ns2 based simulations, we evaluate the proposed framework.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols, Wireless Communications

General Terms

Algorithms, Design, Reliability, Performance

Keywords

Wireless Sensor Networks, Reliable Transport Protocols, Sink-to-Sensors Reliability, Energy Conservation

1. INTRODUCTION

In this paper, we consider the problem of *reliable downstream point-to-multipoint data delivery*, from the sink to the sensors, in wireless sensor networks (WSNs). The need (or lack thereof) for reliability in a sensor network is clearly dependent upon the specific application the sensor network is used for. Consider a security application where image sensors are required to detect and identify the presence of critical targets. Given the critical nature of the application, it can be argued that any message from the sink has to reach the sensors reliably.

Now, for this security application the sink may send one of the following three classes of messages, all of which have to delivered reliably to the sensors: (i) If the underlying network is composed of reconfigurable sensors[1], the sink may want to send a particular (say upgraded) image detection/processing software to the sensors. We refer to such messages as *control code*. (ii) Next, the sink may have to send a database of target images to the sensors, to help in the image recognition triggered by subsequent queries. We refer to such data as the *query-data*. (iii) Finally, the sink may send out one or more *queries* requesting information about the detection of a particular target. The sensors can then match targets detected with the pre-stored images, and respond accordingly.

The problem of reliable data delivery in multi-hop wireless networks is by itself not new, and has been addressed by several existing works in the context of wireless ad-hoc networks [13]. However, such approaches do not directly apply to a sensor environment because of three unique challenges imposed by the following considerations: (i) *Environment considerations:* The constraints imposed by a WSN environment is substantially different from those imposed by other types of multi-hop wireless networks. A few examples include the limited lifetime of network nodes, the scarcity of the bandwidth and energy, and the size of the network itself. (ii) *Message considerations:* While most approaches for group reliable transport over multi-hop wireless networks in related work consider large sized messages (spanning several packets), most messages in a sensor network might be small sized *queries*. This raises fundamental issues on what kind of loss recovery schemes can be employed. (iii)*Reliability considerations:* The notion of reliability that is traditionally prevalent is that of a simple 100% reliable data delivery. However, WSNs might require other notions of reliability ranging from reliable delivery to only a sub-region of the network to partial probabilistic reliability for scoped-resolution based querying.

In this paper, we address the above challenges and present an approach called GARUDA[2] that provides reliable point-to-multipoint data delivery from the sink to the sensors. GARUDA is scalable with respect to the network size, message characteristics, loss rate,

*This work was funded in part by the National Science Foundation under grant ECS-0225497, and by the Georgia Electronic Design Center.

MobiHoc'04, May 24–26, 2004, Roppongi, Japan.

[1]Sensors that can operate in one of several modes of operation.
[2]A mythological bird that reliably transported gods.

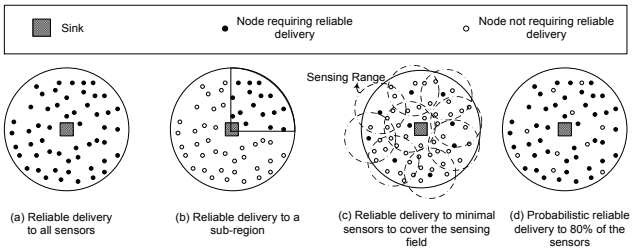


Figure 1: Types of reliability semantics

and reliability semantics, and consists of the following elements as the cornerstones of its design: (i) an efficient pulsing based solution for reliable short-message delivery; (ii) a virtual infrastructure called the *core* that approximates a near optimal assignment of local designated servers, which itself is instantaneously constructed during the course of a *single* packet flood; (iii) a two-stage NACK based recovery process that effectively minimizes the overheads of the retransmission process, and performs out-of-sequence forwarding to leverage the significant spatial re-use possible in a WSN; and (iv) a simple candidacy based solution to effectively support the different notions of reliability that might be required in a WSN. We show through both macroscopic and microscopic results that GARUDA shows great promise in terms of its performance.

The rest of the paper is organized as follows: Section 2 motivates the problem of downstream reliability, identifies the key goals, and discusses the challenges associated with realizing the goals. Section 3 describes the various design elements in GARUDA framework, while Section 4 presents the proposed framework approach for achieving downstream reliability to all sensors. Section 5 describes the framework for supporting the reliability variants. Section 6 compares the performance of the proposed framework with that of existing approaches. Section 7 discusses the related works for providing reliability in sensor networks and other related environments. Finally, Section 8 concludes the paper.

2. PROBLEM DEFINITION AND CHALLENGES

The problem addressed in this work is that of reliable sink-to-sensors downstream data delivery in wireless sensor networks (WSNs). We restrict the focus of the work to WSNs with a single sink and static sensors, and assume that the lack of communication reliability can be due to various reasons such as random wireless errors, congestion, or other failures. The problem scope includes tackling the diverse reliability semantics that might be required in WSNs. The goal is thus to achieve reliability while minimizing bandwidth usage, energy consumption, and delay, with the solution not only addressing the unique characteristics of WSNs, but also leveraging them where appropriate.

2.1 Challenges

We now present the fundamental challenges that need to be addressed for providing effective downstream reliability in WSNs:

2.1.1 Environment Constraints:

There are two primary limitations in a WSN that need to be tackled to provide effective downstream data reliability: (i) bandwidth and energy constraints, and (ii) frequent node failures.

The bandwidth and energy constraints may be tackled by minimizing the amount of *retransmission* overheads to ensure reliability. This in turn will reduce both bandwidth and energy consumption due to the reliability process. The proneness to node failures, on the other hand, should be tackled by not relying on statically constructed mechanisms (say, a broadcast tree) that do not account for the dynamics of the network. Note that "dynamic" mechanisms that periodically refresh any constructions are not desirable as the overheads due to the reliability process have to be minimized too.

Another characteristic of the target environment that needs to be accounted for is the scale of the network. WSNs can be expected to be of a large scale in terms of the number of nodes, and hence the diameter of the network. This in turn means that there is a tremendous amount of *spatial reuse* possible in the network, that should be tapped for achieving the best capacity, and hence delay. However, the specific loss recovery mechanism used may severely limit such spatial re-use as we elaborate in the next section.

2.1.2 ACK/NACK Paradox:

While the previous challenge was with regard to the constraints imposed by the environment, this challenge stems from the constraints imposed by typical message types that can be expected to use the downstream reliability. While the query-data and control code can be expected to be of non-trivial message size, queries pose a unique problem because of their short message sizes.

Negative acknowledgments (NACKs) are well established as an effective loss advertisement mechanism in multi-hop wireless networks in particular, and group communication in general as long as the loss probabilities are not inordinately high. However, NACKs cannot handle the unique case of all packets in a message being lost at a particular node in the network. Since the node is not aware that a message is expected, it cannot possibly advertise a NACK to request retransmissions.

If the message sizes are large, the probability of all packets in the message not arriving at a node will be negligible. But, for message types like queries, where it is very reasonable to expect messages to be merely a few packets long (if at all), the probability that a node does not receive any packet in a message is non-negligible, and hence has to be explicitly tackled.

While an ACK based recovery scheme would address the problems, its other deficiencies (in terms of ACK implosion) however clearly prohibit it from being used.

Finally, revisiting the issue of tapping spatial re-use, a NACK based loss recovery scheme will inherently require *in-sequence forwarding* of data by nodes in the network to prevent a NACK implosion [15]. This will clearly limit the spatial re-use achieved in the network.

2.1.3 Reliability Semantics:

Our final discussion is on constraints that are imposed by the *notion of reliability* that typical WSNs will require.

Two characteristics that are innate to a WSN environment are location dependency and redundancy in deployment. A query can be location dependent such as "Send temperature readings from rooms X, Y, and Z". At the same time, the redundant deployment of sensors in the field means that in order to get reliable sensing *information*, it is not necessary for all sensors in the field to reliably deliver their locally sensed data to the sink. Furthermore, a sink might also choose to reliably deliver a message only to a probabilistic fraction of the entire network, say as part of a sensing strategy that involves incrementally increasing resolution [5].

We thus define the reliability semantics that can be required in WSNs based on the above characteristics. We classify the reliability semantics into four categories: (i) *delivery to the entire field*,

which is the default semantics, (ii) *delivery to sensors in a sub-region of the field*, which is representative of location based delivery, (iii) *delivery to sensors such that the entire sensing field is covered*, which is representative of redundancy aware delivery, and (iv) *delivery to a probabilistic subset of sensors*, which corresponds to applications that perform resolution scoping.

Figures 1(a)-(d) illustrate categories (i) through (iv) respectively. Thus, any reliability solution should not only support the default reliability semantics, but also the other types of semantics that are unique to wireless sensor environments.

3. GARUDA DESIGN ELEMENTS

In this section, we present an overview of GARUDA's design that explicitly tackles the challenges identified in Section 2. The centerpiece of GARUDA's design is an instantaneously constructible loss recovery infrastructure called the *core*. The *core* is an approximation of the minimum dominating set (MDS) of the network sub-graph to which reliable message delivery is desired. While using the notion of a MDS to solve networking problems is not new ([11]), the contributions of this work lie in establishing the following for the specific target environment: *the relative optimality of the core for the loss recovery process*, *how the core is constructed*, *how the core is used for the loss recovery*, and *how the core is made to scalably support multiple reliable semantics*.

We present a *core construction* approach that constructs the *core* during the course of a single packet flood, and propose a *two-phase loss recovery* strategy that uses *out-of-sequence forwarding* and is tailored to satisfy our basic goals of minimizing retransmission overheads and minimizing delay. Finally, we show how a simple *candidacy* based approach for *core* construction can make the *core* scalably support multiple reliability semantics.

The second cornerstone of the GARUDA design is a pulsing based approach to deliver a single packet reliably to all the network nodes. Recall the trade-offs identified in Section 2 for reliable delivery of short-messages. Since GARUDA can ensure the reliable delivery of the first packet of messages of any size, it is no longer vulnerable to the *all packets lost problem* that straightforward NACK based schemes are susceptible to. This enables GARUDA to tap the advantages of NACK based schemes, but at the same time avoid any pitfalls that consequently arise.

In the rest of the section, we provide high level overviews of each of the above components. For the sake of clarity, we start with discussing the details about the *core* infrastructure in GARUDA. We assume that the first packet is reliably delivered for the initial discussions. Then, in Section 3.4, we present the details of how GARUDA achieves reliable first packet delivery.

3.1 Loss Recovery Servers: Core

GARUDA uses *local and designated loss recovery servers* in its loss recovery process. The motivation for localized recovery - reducing bottlenecks at the (otherwise) non-local servers, and reducing recovery time, and designated servers - preventing unnecessary redundant retransmissions by neighbors upon a retransmission request, have been well established in related work ([3]), and we do not delve further into the motivation in the interest of space.

The *core* in GARUDA thus forms the set of local designated loss recovery servers that help in the loss recovery process. The challenges that hence arise are (i) how should the core nodes be chosen in order to minimize retransmission overheads? and (ii) how can the *core* be constructed in a manner that is appropriate for the limiting characteristics (dynamic topology change due to node failures) of the target environment?

Ideally, the core designation should be done on a per-packet basis based on the loss pattern experienced during the packet delivery. Once the loss pattern is known, performing optimal[3] server designation reduces to the well known *minimum set-cover problem* (MSC) [6]. While the solution to the set-cover problem is ideal, it is obviously not a feasible one from the standpoint of performing such a core designation on a per-packet basis.

GARUDA, instead, performs core designation on a per-message basis[4], and independent of the loss patterns of the packets. It designates loss recovery servers by dynamically electing a subset of the nodes in the network as *core* nodes for each message delivery. While the *core* is thus not optimal for each packet loss pattern (does not approximate the minimum set cover for the loss pattern), it approximates the minimum dominating set (MDS) [2]. If we assume a uniform distribution for the placement of sensor nodes, it can be shown that the approximation of the minimum dominating set used in the construction of the *core* approaches a worst case ratio of $\frac{1+\ln(n)}{1+\ln(n)+\ln(p)}$ to a polynomial time approximation of the MSC structure for any loss pattern, where p is the loss probability ($0 < p \leq 1$), and n is the total number of nodes in the network.

3.1.1 Instantaneous Core Construction

In GARUDA, the *core* is constructed using the first packet delivery. The reliable delivery of the first packet determines the *hop_count* of the node in the network, which is the distance of the node from the sink. A node, which has a *hop_count* that is a multiple of three, elects itself as a core if it has not heard from any other core node. In this fashion, the core selection procedure approximates the MDS structure in a distributed fashion. The uniqueness of the *core* in GARUDA lies in the following characteristics: (i) the *core* is constructed using a single packet flood, more specifically during the flood of the first packet; and (ii) the structure of the sensor network topology (with sensors placed at fixed distances from the sink) is leveraged for more efficient, and fair *core* construction. Note that such an instantaneous construction of the core nodes during the first packet delivery of every new message addresses any vulnerability in the network in terms of node failures occurring at the granularity of a message. We defer the discussion of how our approach handles node failures occurring during a message transmission to Section 4.

3.2 Loss Recovery Process

3.2.1 Out-of-sequence forwarding with A-map propagation

In GARUDA, an out-of-order forwarding strategy is used while forwarding packets as opposed to an in-sequence forwarding scheme. A key drawback of the in-sequence forwarding strategy is that precious downstream network resources can be left under-utilized when forwarding of higher sequence number packets is suppressed in the event of a loss. An out-of-sequence forwarding on the other hand can overcome this problem as nodes that have lost a packet can continue to forward any higher (or lower) sequence number packets that they might have received. However, such an approach can potentially lead to unnecessary NACK implosion, where downstream nodes will issue a chain of NACK requests for holes detected in the sequence of packets received, even when the concerned packets are not available.

To inhibit such unnecessary retransmission requests, GARUDA uses a scalable *A-map* (Availability Map) exchange between core

[3]In terms of the number of retransmissions required.

[4]Performing designation at any larger time granularity will compromise the goal of addressing network dynamics and supporting different reliability semantics.

nodes that conveys meta-level information representing availability of packets with bits set. Any downstream core node initiates a request for a missing packet only if it receives an *A-map* from an upstream core node with the corresponding bit set. The *core* recovery phase in GARUDA is highly efficient as the core nodes initiate requests only when they are sure of an upstream core node having a particular packet. While the overhead associated with the A-map is an obvious concern, the performance results for GARUDA in Section 6 take into account the A-map overhead, and hence any improvements shown are after accounting for the A-map overhead.

3.2.2 Two-stage Loss Recovery

Once the *core* is constructed, the framework employs a *two-stage recovery process* that first involves the core nodes recovering from all lost packets, and then the recovery of lost packets at the non-core nodes. The reasons for using two-stage recovery are threefold: (i) the number of non core nodes will be a substantial portion of the total number of nodes in the network, and hence precluding any contention from them is desirable; (ii) when the core nodes perform retransmissions for other core nodes, holes corresponding to a single packet among a core node's neighbors would also be filled with a single retransmission; and (iii) when only the core nodes are performing retransmissions during the second phase, due to the nature of the *core* (ideally, no two core nodes are within two hops of each other), the chances for collisions between retransmissions from different core nodes are minimized.

- *Loss Recovery for Core Nodes*: The recovery process for the core nodes is performed in parallel with the underlying default message-forwarding. This is done in order to ensure that the core nodes receive all the packets in a message as quickly as possible. This parallel recovery process for the core nodes does not increase the contention in the network significantly because the fraction of core nodes is very small compared to the total number of nodes in the network, and all requests and retransmissions are performed as unicast transmissions to the nearest upstream core that has a copy of the lost packet.
- *Loss recovery for Non-core Nodes*: The second phase of the loss recovery starts only when a non-core node overhears an *A-map* from the core node indicating that the core node has received all the packets in a message. Hence, the second phase of the loss recovery does not overlap with that of the first phase in each local area, preventing any contention with the basic flooding mechanism, and with the first phase recovery.

While the two phase loss recovery can potentially increase latency, we show in Section 6 that the proposed framework incurs a latency which is in fact significantly smaller than competing approaches.

3.3 Multiple Reliability Semantics

In this section, we outline briefly how the *core* construction can be simply modified to account for the multiple reliability semantics identified in Section 2. We first assume, without loss of generality, that a given instance of reliability semantics will require reliable delivery to a subset G_S of the nodes in the underlying graph G. Consider the subset G_S to consist of K components, where each component is connected, but the components themselves are not connected with each other. The desired infrastructure for such a setting will entail the computation of the MDS for each component, and connecting the components back to the sink using a *travelling salesman path (TSP)*, if bandwidth costs were the optimization criterion [14].

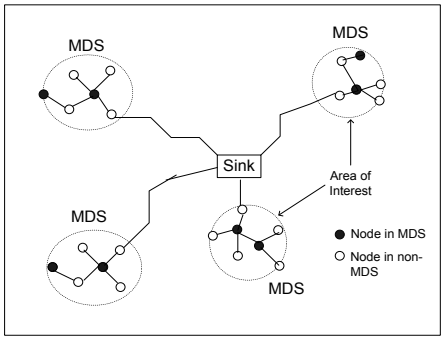


Figure 2: Core structure when target subgraph $G_S \subset G$

GARUDA uses a simpler, but reasonably effective, technique of computing the individual MDSs and connecting them back to the sink using an approximation of the *shortest path tree* (SPT). While this may incur additional bandwidth costs, note that it will have the benefit of better delay properties, in addition to being implicitly constructible as we describe in Section 5. Figure 2 shows GARUDA's solution that finds the minimum dominating set within each partition and approximates the SPT connecting all minimum dominating sets (MDS) to the sink.

The MDS within each component is constructed with minor changes to the *core* construction algorithm that merely involves nodes employing a *candidacy* check before participating in the *core* construction algorithm. The candidacy check is where nodes, upon receiving the first packet, determine whether or not they belong in the subset G_S. Nodes outside Gs but required for the construction of the SPT are inducted into the core structure through a *forced candidacy* mechanism.

3.4 Reliable Single/First Packet Delivery

Thus far, we have discussed the details of the *core* infrastructure in GARUDA, *assuming that the first packet is delivered reliably to all nodes in the network*. In the rest of the section, we outline how such first packet reliability is achieved.

Since NACK based request schemes do not suffice for a single packet delivery (or when all packets in a message are lost) without any support, we consider an ACK based scheme as an alternative just for the first packet[5]. However, such an approach will still incur the undesirable ACK implosion problem identified in Section 2.

GARUDA addresses the reliable delivery of the first packet using a *Wait-for-First-Packet (WFP)* pulse, which is a small finite series of short duration pulses, where the series is repeated periodically. The pulse has an amplitude that is much larger than that of a regular data transmission, and a period that is significantly smaller than that of a regular data transmission. The unique property of the pulse is that any receiving node, irrespective of whether it is currently idle or receiving a regular data packet, can sense the pulse due to the pulse's specific amplitude/period characteristics.

When a sink wants to send the first packet, the sink transmits the finite series of WFP pulse on a periodic basis. The sensor nodes within the transmission range of the sink, upon reception of the pulses, also start pulsing with the same periodicity between two series of pulses and this process is repeated until all the nodes start pulsing in anticipation of the reception of first data packet. The

[5]Note that as long as one of the packets is delivered to every node with sufficient information about the message (e.g. length), a NACK based scheme can be successfully used to provide guaranteed reliability.

sink after pulsing for a finite duration (so as to ensure that the pulses have propagated across multiple hops in the network), transmits the first packet as a regular data packet transmission and stops sending any further WFP pulses. Every sensor upon reception of the first packet also performs the same set of two actions.

Essentially, the WFP signal serves two purposes: (i) it allows the sink to inform the sensors about an impending message that has reliability requirements, and (ii) it enables sensors to request for retransmissions when they do not receive the first packet successfully. It might appear that resource constrained sensors can be overloaded, in terms of energy consumption, and cost, with the addition of the pulsing mechanism. However, we argue that the addition of the WFP signal alleviates several problems associated with reliable message delivery, and can in fact provide benefits that far outweigh the costs.

Briefly, (i) since the WFP pulse is just used to indicate the arrival of an impending new transmission, it requires a simpler modulation scheme than the default data transmissions and, is more robust to fading effects; (ii) the message advertisement scheme using WFP pulses is inherently robust to collisions, as the collisions of WFP pulse with other such pulses or data transmissions does not prevent sensors hearing the WFP pulses from inferring the impending message transmission (they still will sense that the WFP pulses are being sent) [4]; (iv) unlike in the ACK based scheme, where the ACK implosion can adversely impact the data transmissions as they do not scale well to increasing number of nodes in the network, the WFP pulse serves as an *implicit NACK* and (because of their small width) interferes to a very minimal extent with the regular data transmissions; and (v) the energy consumption of the WFP pulse is significantly smaller than that of a regular data transmission, thus rendering any additional energy consumption to be far less than the actual energy savings because of the other benefits[6].

4. GARUDA FRAMEWORK

The details of the GARUDA framework is presented in this section assuming a simple underlying flooding mechanism. However, GARUDA can as well be integrated with the flooding scheme itself. We assume that every incoming flooded packet is passed to GARUDA if it is part of a message that requires reliability.

The different components of GARUDA are explained in the chronological order that they occur when a reliable message is flooded. Hence, we first describe the details of GARUDA's pulse based single packet delivery mechanism, and then go on to describe the *core* construction and loss recovery procedures. Note that the reliable single packet delivery is leveraged for the instantaneous *core* construction.

4.1 Single/First Packet Delivery

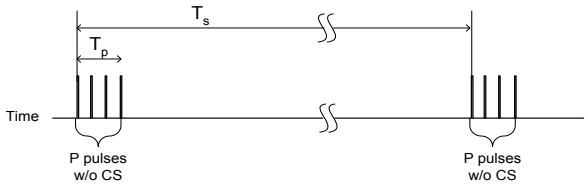


Figure 3: Transmission time of Wait-for-First-Packet pulse

[6]We profile the energy savings through the use of WFP pulses in Section 6.

4.1.1 WFP Pulse Transmission

Since a WFP Pulse can be regarded as a short period signal which does not include any information, the transmission period of the WFP pulse is significantly smaller when compared to the transmission time T_D required for a regular data packet. Also, twice the regular transmission power is used to transmit the pulses to achieve a relative amplitude of 3dB at the receiver (with respect to a default reception). The detection of a WFP pulse at a receiver is done based on a simple energy detection strategy that monitors changes in the amplitude of the energy of the incoming signal, and the duration of any such changes [4]. Note that the changes in energy can be detected even at receivers whose local channel is busy with an ongoing data transmission. The only nodes that cannot hear the WFP pulses are those that are not listening (either in transmit mode, or in a power-down mode).

To increase the robustness of the pulse detection, every set of pulse transmission includes p pulses transmitted consecutively within a period T_P ($T_P << T_D$). Figure 3 shows the transmission scheme for the WFP pulse. Hence, receivers infer an incoming WFP signal only after detecting p pulses.

The basic (and the only required) mechanism for WFP pulsing in GARUDA does not use any carrier sensing, and hence is referred to as *forced WFP pulsing*. This ensures that nodes that need to transmit the WFP (either as an advertisement or a NACK for the first packet) can do so without having to suffer from any MAC layer starvation problems. However, such transmissions clearly increase the chances for collisions with regular data packet transmissions, and hence are performed with a period T_s, where $T_s >> T_D$.

However, the forced pulsing in GARUDA is complimented with a carrier-sensing based WFP, and a data packet piggybacking based advertisement scheme that reduce the impact of the forced WFP[7].

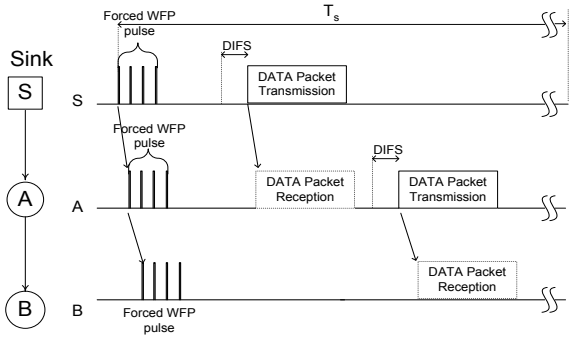


Figure 4: Example for Single/First packet delivery

4.1.2 First Packet Delivery in GARUDA

The delivery procedure for the single/first packet consists of three modes: (1) the advertisement which notifies the ensuing single/first packet to all nodes with the forced WFP pulses; (2) the delivery which sends the single/first packet through simple forwarding; and (3) the recovery which sends NACKs using WFP pulses to request for retransmission of the single/first packet.

Figure 4 shows the basic procedure of the single or the first packet delivery with a simple topology. When a sink wants to initiate a reliable single/first packet delivery, it sends a set of forced WFP pulses without sensing the wireless channel. When neighboring sensors hear WFP pulses, they send a set of forced WFP

[7]But note that the *guarantee* of reliable first packet delivery is provided only by the forced WFP.

pulses immediately. After a deterministic period that is set based on the diameter of the network, the sink transmits the single/first data packet subject to the medium access scheme (e.g. CSMA).

If the node A receives the single/first packet, it changes its operation from the advertisement mode to the delivery mode by halting the WFP pulses, and by sending the single/first data packet after carrier-sensing. However, if the single/first packet is lost, nodes will continue to transmit the WFP pulses, which in turn trigger retransmissions. Figure 5 shows the case of retransmission.

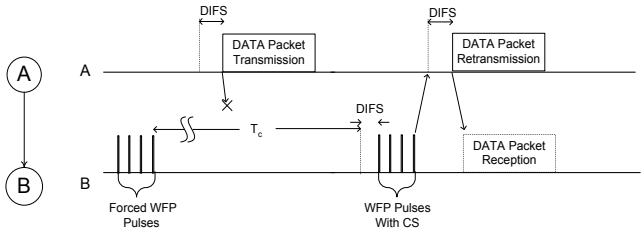


Figure 5: Example for loss detection and recovery

Since the forced WFP pulses sent every T_s period play the role of a NACK signal, node B will wait for a duration of at least T_s to send next set of forced WFP pulses. Therefore, the latency for the single/first packet delivery is directly dependent upon T_s.

To reduce the latency, GARUDA uses another kind of WFP pulse which a node sends after a regular carrier sensing operation. Node B sends p number of WFP pulses after carrier-sensing (WFP_{cs}) opportunistically (unless it has received the single/first packet) with a period T_c which is smaller than T_s. The period T_c should be proportional to the hop distance of the node B from the sink because a node should wait until the upstream nodes between the node and the sink receives the single/first packet. T_c is heuristically set to the following value in GARUDA:

$$T_c = i \times \Delta \times T_D, \quad (1)$$

where i is the hop distance from a sink to a node, and Δ is the maximum node degree.

Since a node senses the state of channel before transmitting WFP$_{cs}$ pulses, the WFP$_{cs}$ pulses have a lesser probability of colliding with data packets than WFP pulses. When a node gets to transmit WFP$_{cs}$ pulses, it resets the timer corresponding to the T_s time period for forced WFP pulses.

A further opportunistic optimization that GARUDA uses is the piggybacking of the NACK information on the regular data packet transmissions. The NACK is merely the sequence number of the last message ID the node has received thus far. Any neighbor that is aware of a greater message ID and has the corresponding first packet then retransmits. We refer to this as an implicit NACK mechanism.

4.2 Instantaneous Core Construction

4.2.1 The Core

In this section we present the details of the instantaneous *core* construction assuming a simple 100% network-wide reliable flood. We revisit the case of other reliability semantics in Section 5.

Assuming a network organization with the sink at the center of a sensor field, the first packet delivery establishes *band-ids* for nodes based on the hop distance that they perceive from the sink[8]. This is shown in Figure 6 stage 1. We consider all nodes with the same band-id as forming a "band" with a certain id. The bands can thus be viewed as concentric circles around the sink.

4.2.2 Procedure

The *core* construction algorithm works as follows:

Sink:

- When the sink sends the first packet, it stamps the packet with a "band-id" (bId) of 0[9]. When a sensor receives the first packet successfully, it increments its bId by one, and sets the resulting value as its own band-id. The band-id is representative of the approximate number of hops from the sink to the sensor[10].

Nodes in $3i$ bands:

- Only sensors with band-ids of the form $3i$, where i is a positive integer, are allowed to elect themselves as core nodes.
- When a sensor S_0 with a band-id of the form $3i$ forwards the packet (after a random waiting delay from the time it received the packet), it chooses itself as a core node if it had not heard (or snooped) from any other core node in the same band. Once a node chooses itself as a core node, all packet transmissions (including the first) carry information indicating the same.
- If any node in the core band that has not selected itself to be a core receives a core solicitation message explicitly, it chooses itself as a core node at that stage.
- Every core node S_3 in the $3(i+1)$ band should also know of at least one core in the $3i$ band. If it receives the first packet through a core in the $3i$ band, it can determine this information implicitly as every packet carries the previously visited core node's identifier, bId, and *A-map*. However, to tackle a condition where this does not happen, S_3 maintains information about the node (S_2) it received the first packet from, and the S_2 node maintains information from the node (S_1) it received the first packet from. After a duration equal to the core election timer, S_3 sends an explicit *upstream core solicitation* message to S_2, which in turn forwards the message to S_1. Note that by this time, S_1 will already have chosen a core node, and hence it responds with the relevant information.

Nodes in $3i+1$ bands:

- When a sensor S_1 with a band-id of the form $3i+1$ receives the first packet, it checks to see if the packet arrived from a core node or from a non-core node. If the source S_0 was a core node, S_1 sets its core node as S_0. Otherwise, it sets S_0 as a candidate core node, and starts a core election timer[11]. If S_1 hears from a core node S_0' before the core election timer expires, it sets its core node to S_0'. However, if the core election timer expires before hearing from any other core node, it sets S_0 as its core node, and sends a unicast message to S_0 informing it of the decision.

[8]Note that this view of the network is purely to ensure description clarity, and has no bearing on the correctness of the approach.

[9]To balance the load of core and non-core nodes, the sink can choose the band-id among 0, 1, and 2. Therefore $3i$ bands (core bands) can be changed dynamically

[10]Note that due to the availability of multiple paths from a sink to sensors, it is possible that the computed band-id is either greater than the minimum number of hops from the sink to the sensors.

[11]The timer is set to a value larger than the retransmission timer for the first packet delivery.

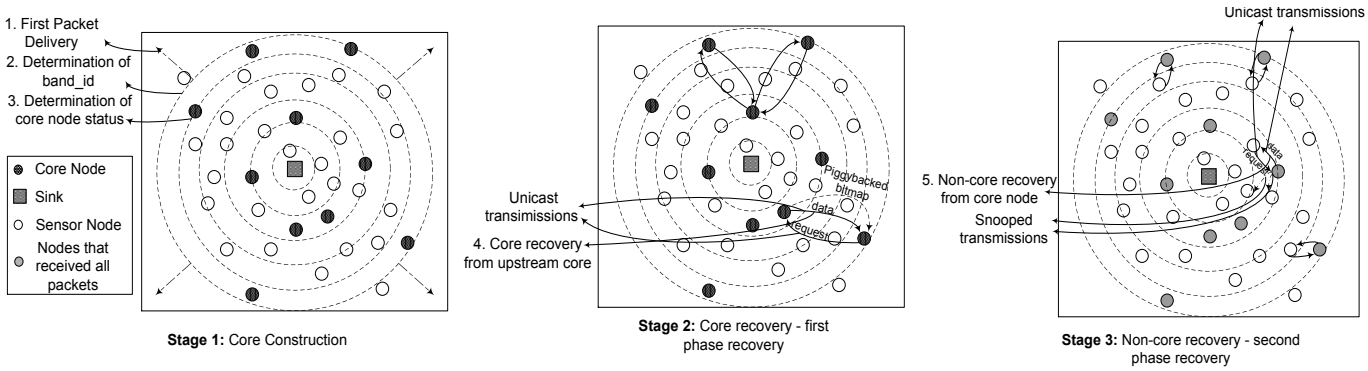


Figure 6: Instantaneous $core$ construction and two-stage loss recovery in GARUDA

Nodes in $3i + 2$ bands:

- When a sensor S_2 with a band-id of the form $3i + 2$ receives the first packet, it cannot (at that point) know of any $3(i + 1)$ sensor. Hence, it forwards the packet without choosing its core node, but starts its core election timer. If it hears from a core node in the $3(i + 1)$ band before the timer expires, it chooses the node as its core node. Otherwise, it arbitrarily picks any of the sensors that it heard from in the $3(i + 1)$ band as its core node and informs the node of its decision through a unicast message. If it so happens that S_2 does not hear from any of the nodes in the $3(i + 1)$ band (possible, but unlikely), it sends an anycast *core solicitation* message with only the target band-id set to $3(i + 1)$. Any node in the $3(i+1)$ band that receives the anycast message is allowed to respond after a random waiting delay. The delay is set to a smaller value for core nodes to facilitate re-use of an already elected core node.
- A boundary condition that arises when a sensor with a band-id of $3i + 2$ is right at the edge of the network, is handled by making the band act just as a candidate core band ($3i$). Such a condition can be detected when nodes in that band do not receive any response for the anycast core solicitation message.

Thus, at the end of the first packet delivery phase, each node knows its bId, whether it is a core node or not, and in the latter case its core node information. In addition, every core node in the $3(i + 1)$ band knows of at least one core node in the $3i$ band.

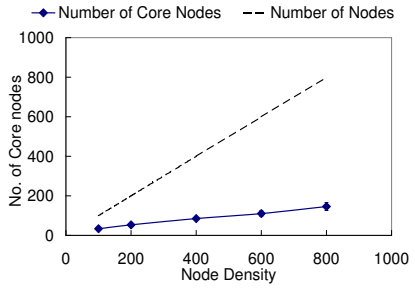


Figure 7: Number of core nodes vs. total number of nodes

4.2.3 *Optimality of the Core*

Since the core nodes approximate a minimum dominating set, an obvious question is how is the *core* construction set up in a way to minimize the number of core nodes. Ideally, for any given core node, there should not be any other core node in its 2-hop neighborhood. The proposed framework attempts to achieve this condition using a two-pronged approach: (i) only nodes in 3i bands (core bands) are allowed to contend to become a core node; and (ii) of the nodes that belong to the core bands, only nodes that have not heard from any other core node from its band are allowed to choose themselves as core nodes. Figure 7 shows the number of core nodes as the node density is increased from 100 to 800. As we can see, the number of core nodes decreases from 30% when the node density is 100 to about 13% when the node density is 800.

4.3 Two Phase Loss Recovery

4.3.1 *Loss Recovery for Core Nodes*

4.3.1.1 *Loss Detection.*

When a core node receives an out-of-sequence packet, the core node infers a loss. A core node sends a request to an upstream core node only if it is notified through an *A-map* that the missing packet is available at the upstream core node.

4.3.1.2 *Loss Recovery.*

When a core node receives a unicast request from a downstream core node, it performs a unicast retransmission for the request. Figure 6 stage 2 shows the loss detection and the loss recovery between core nodes at $3i$ band and core nodes at $3(i + 1)$ band. If any of the non-core nodes on the path of the unicast request has the requested packet, it intercepts the request and retransmits the requested packet.

The use of the *A-map* is central to the core recovery process. For the sake of brevity, we assume that the *A-map* is capable of representing all packets of a message irrespective of the message size. The core recovery process works as follows:

Upstream Core Nodes:

- A core node, when it forwards a packet, stamps on the packet the following meta information: (C_{id},*A-map*, $bId, vFlag$), which consists of the core node's identifier, bit map, band-id, and valid flag respectively. The valid flag is used by a recipient core node to determine whether the path in the meta information is valid or not.
- When a core node receives a retransmission request, it responds with unicast retransmissions of the available packets.

Intermediate Non-core Nodes:

- Any non-core node NC_{id} that forwards a packet leaves the *A-map* information untouched, but adds its own identifier as

follows: ($C_{id} + NC_{id}$,*A-map*, bId). If the number of the identifiers in the incoming information is equal to three, the non-core node does not add its identifier and sets the $vFlag$ to NULL.

Downstream Core Nodes:

- Thus, when a core node receives the meta information, it not only knows of what packets the source core node has, but also the path it can use to request for a retransmission. If the $vFlag$ is NULL, the core node still uses the *A-map* information, but falls back on any earlier cached path to the relevant core node for issuing the request.

- Each core node maintains two *A-map*s locally: $myBM$ which represents the successfully received packets, and $totBM$ which represents both the received and the requested packets.

- When a core node receives an incoming *A-map* ($inBM$), it checks to see if the *A-map* is from a valid source. If the source is valid, it then checks to see if the A-map conveys availability of a packet that has neither been received nor been requested. If at least one such packet is available, the node creates a request *A-map*, updates its $totBM$, and sends the request. It also starts an expiry timer for the request.

- For a successful packet reception, the core node updates its $totBM$ and $myBM$. Also, when a timer expiry occurs for a request, $totBM$ is updated accordingly.

- When a core node does not hear an *A-map* from any of its upstream core nodes for a specified duration (*core presence timer*[12]), it explicitly issues a request to the default upstream core node to which the upstream core node responds with its current *A-map*.

4.3.2 Loss Recovery for non-Core Nodes

A non-core node snoops all (re)transmissions from its core node. Once it observes an *A-map* from its core node with all the bits set, it enters the non-core recovery phase by initiating retransmission requests to the core node. Alternatively, if it does not hear from its core node for the period *core presence timer*, it sends an explicit request to the core node to which the core node responds with its current *A-map*. Figure 6 stage 3 presents the loss detection and recovery between non-core nodes and a core node. Since all retransmissions from the core nodes are snooped by the non-core nodes, redundant retransmissions for the same loss are removed.

5. SUPPORTING OTHER RELIABILITY SEMANTICS

In Section 4, GARUDA was described in the context of single and multiple packet delivery, while assuming the simplest form of reliability semantics along the other dimensions (all nodes, 100% reliability). In this section, we revisit the GARUDA design and show how it can accommodate the other reliability semantics. Specifically, we discuss three variants in terms of the reliability semantics: (i) reliable delivery to all nodes within a sub-region of the network; (ii) reliable delivery to minimal number of sensors required to cover entire sensing area; and (iii) reliable delivery to p% of the nodes in the network.

The fundamental difference between the context in Section 4, and in the above variants is that only a subset of the nodes in the network require reliable delivery. The variants differ in *which subset of nodes* receive the message delivery. We refer to the problem of determining the subset as the *candidacy* problem. Also, in all of the solutions discussed, the first packet is always delivered to all nodes in the network. All subsequent packets are delivered based on the candidacy.

Generically, the solutions to the three variants use three common elements to tackle the other reliability semantics:

- The first packet carries information to identify the eligibility for candidate nodes that should receive the entire message reliably. For example, in the case of reliability within a sub-region, the first packet may carry a coordinate based description of the sub-region.

- Participation in the *core* construction is limited to only those nodes that have chosen themselves as candidates. Note that the other aspects of the *core*-construction still remain the same (nodes only in the $3i$ bands can select themselves as core nodes, etc.). At the end of the *core* construction, each independent component of the candidate sub-graph G_S thus has its own core.

- The last element in GARUDA is that of *forced candidacy* to enable the *core* of the different components to be connected back to the sink. Thus non-candidate nodes in the $3i$ bands on the path from each component to the sink are forced to participate as candidate core nodes to ensure connectivity. The forced candidacy is actually achievable in GARUDA with very minimal changes to its original design (as described in Section 4). Essentially, non-candidate nodes in core bands, if they would have otherwise chosen themselves as core nodes identify themselves as non-candidate core nodes when the first packet is forwarded. A downstream candidate core node that has not heard from any other candidate upstream core node explicitly requests the upstream non-candidate core node to become a candidate. Through this process, a structure that is an approximation of independent MDSs (within each component of G_S) connected through an SPT is achieved.

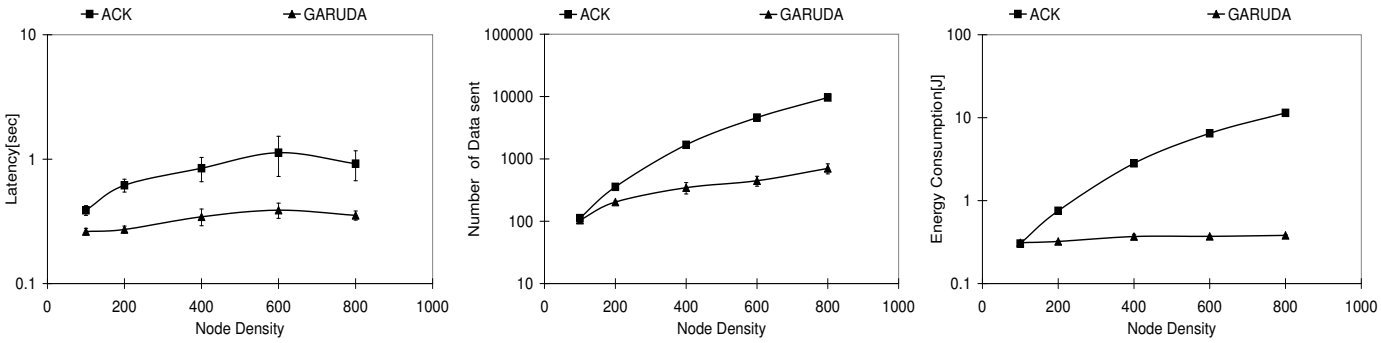
In the rest of the section, we elaborate on how the candidacy for the three variants are established in GARUDA.

5.1 Reliable Delivery within a Sub-region

As we motivate in Section 2, it is quite likely that the sink requires reliable delivery of a query or a message only to sensors within a specific sub-region of the network area. We assume that the specifications of the sub-region are available in the form of coordinates. Without loss of generality, we also assume that the sub-region is rectangular in shape (although the GARUDA design by itself does not have any such limitations). The sub-region can either be contiguous or non-contiguous with the region occupied by the sink.

The desired sub-region coordinates is piggybacked on the first packet sent by the sink. Each sensor in the network that receives the first packet can thus determine locally whether it is a candidate or not, based on its own location and the desired sub-region. Once the candidacy is determined, the behavior of sensors is exactly the same as described in Section 4, except if the sensor were to be on a core band. Whereas in the default operation, a sensor does not choose itself as a core node only if it hears from another core node before it transmits, under this variant, a sensor does not choose itself as a core node if it is not a candidate irrespective of the other conditions. Note that this does not mean that such a sensor can later be forced to become a core node, as we elaborate next.

[12]The timer is set to a value larger than three-hop round trip time.



(a) Latency (b) Number of data packet sent (c) Energy consumption per node

Figure 8: Performance evaluation of GARUDA: single packet delivery

5.2 Reliable Delivery to Cover Sensing Field

This variant requires reliable delivery while remaining aware of the inherent redundancy in the sensor network deployment. Specifically, under this variant, reliable delivery needs to be performed only to a minimal subset of the sensors in the network such that the entire sensing field is covered. For purposes of this discussion, we assume that the sensing range S is always less than or equal to the transmission range R.

Unlike in the previous variant where the candidacy of each node is determined locally, in this variant coordination between nodes is required in order to eliminate sensors, which are covering a region already covered by other sensors, from the candidacy. In GARUDA, the core nodes are best equipped to perform such coordination as they are immediately adjacent to all non-core nodes that depend on them, and under ideal conditions are at least a distance of $2R$ away from the nearest core node (which gives a core node a virtual "ownership" of at least the sensing region defined by its transmission range). Thus, non-core nodes under this variant seek permission from their respective core nodes to become candidates. Each core node keeps track of the coverage of the region defined by the square[13] of side $2(S + T)$ (with itself at the center). It provides permission to a seeking non-core node only when the node can cover an area not already covered inside the square. Note that given our assumptions about S and T, no non-core node within a core node's transmission range can have a sensing coverage area that even lies partially outside the above defined square.

All core nodes implicitly become candidates. This is reasonable even without any coordination with other nearby core nodes as under ideal conditions, the distance between a core node and its nearby core nodes will be $2R$, which in turn means that a core node can choose itself as a candidate without concern of overlapping with a nearby core node's sensing region.

5.3 Reliable Delivery to Probabilistic Subset

This variant involves support for reliable message delivery to say p% of the network sensors. Such semantics might be useful when the sink intends to perform *scoped* sensing. In other words, the sink can at the outset decide to sense only 25% of the field, with the intent of increasing the sensed region only upon some triggers detected during the preliminary sensing.

Just as in the case of delivery within a sub-region, determining candidacy in this variant is purely a local process. When a sensor receives the first packet, it chooses itself as a candidate with a probability of p. If the sensor is on a core band, and decides not to be a candidate, it does not choose itself as a core node irrespective of the other conditions.

[13]As an approximation of a circle for simplicity.

6. PERFORMANCE EVALUATION

This section evaluates the performance of the proposed framework for 100% reliability to all sensors. For single packet reliable delivery, we compare the performance of the proposed framework to that of an ACK-based scheme, which uses ACK feedback for packet delivery along with a retransmission timeout. As we had identified in Section 2, a NACK based approach will not be able to recover single packet losses. For multiple packet delivery, we compare GARUDA with both in-sequence delivery and out-of-sequence delivery mechanisms that use NACKs.

6.1 Simulation Environment

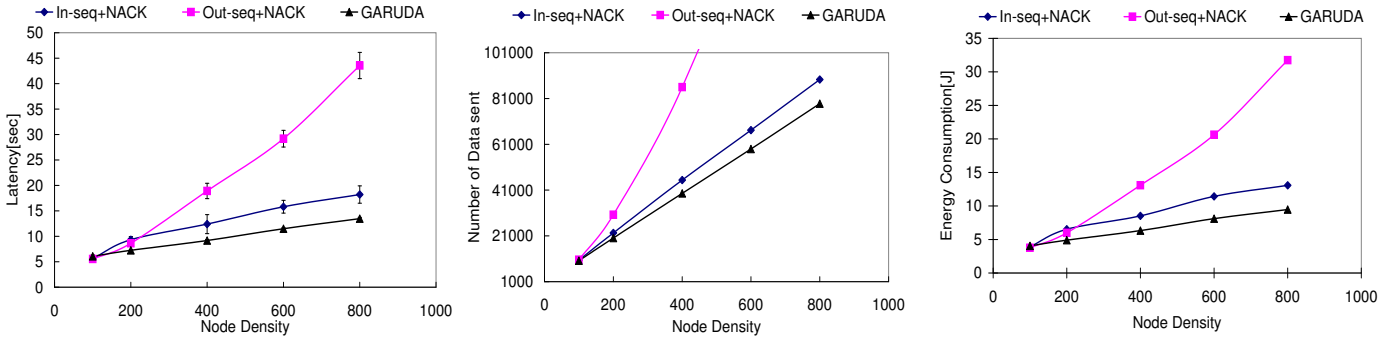
For all *NS2* based experiments: (a) the first 100 nodes are placed in a grid fashion within a 650m x 650m square area to ensure connectivity, while the remaining nodes are randomly deployed within that area, and the sink node is located at the center of one of the edges of the square, (b) transmission range of each node is 67m [10], (c) channel capacity is 1 Mbps, and (d) each message consists of 100 packets transmitted at the rate of 25 packets per second (except for the single packet delivery part), and the size of packet is 1 KB. CSMA/CA is used as the MAC protocol. We use basic flooding as the routing protocol. All the simulation results are shown after averaging the metrics over 20 randomly generated topologies and calculating 95% confidence intervals.

As described in Section 2, losses can occur due to wireless channel errors, or collisions among transmissions. To emulate the two types of losses, we choose a fixed packet loss rate of 5% for wireless channel error, and vary the number of nodes in the network (and hence the network density) which in turn increases the degree of contention in the network.

6.2 Evaluation of Single Packet Delivery

6.2.1 Latency

The latency involved in receiving a single packet reliably with increasing number of sensors is presented in Figure 8 (a) for both GARUDA and the ACK based scheme. The latency of the proposed scheme is significantly smaller because of the WFP pulse, which is essentially an implicit NACK, thus not increasing the load in the network. We also see that the latency scales well with the increase in the number of nodes because of the same reason. However, in the ACK based scheme, the latency is appreciably higher because



(a) Latency (b) Number of data packets sent (d) Energy consumption per node

Figure 9: Performance evaluation of GARUDA: multiple packet delivery

there is explicit ACK feedback to the sender thus increasing the traffic and thereby the collisions in the network.

6.2.2 *Data sent*

Figure 8(b) shows the number of data sent by GARUDA and the ACK based scheme. It is interesting to note that in the proposed framework, the number of data sent increases more or less linearly (with a slope of 1 approximately) as the number of nodes is increased. The reasons can be attributed to the implicit NACK scheme, which alleviates congestion related losses, and the inherent redundancy and the broadcast nature of the flooding process ensures that the packet is received successfully without any need for retransmission even in the presence of losses. For the ACK based scheme, the number of data packets sent is appreciably higher and shows a non-linear increasing trend with increasing number of nodes in the network. This is again because of the increased load in the network due to the presence of ACK transmissions thus increasing the losses in the network.

6.2.3 *Energy Consumption*

The energy consumed per node in joules for each scheme is shown in Figure 8(c). The energy consumed per node is significantly smaller for the proposed framework than the ACK based scheme even though it uses a WFP pulse. This is because of two reasons. Firstly, the duration of WFP pulse is insignificant compared to that of data packet transmissions. In fact, the duration of these WFP pulses can be as low as 15-20μs in order to recognize them [4]. Secondly, WFP pulses themselves do not suffer from any implosion while they address the ACK implosion problem. In fact, the energy consumed shows a linear increase with increasing number of nodes.However, the ACK based scheme suffers from NACK implosion problem because of which energy consumption per node increases with increasing node density.

6.3 Evaluation of Multiple Packet Delivery

To compare the performance of GARUDA for multiple packet delivery, we have implemented two simple reliable transport protocols that allow in-sequence and out-of-sequence forwarding respectively, coupled with NACK based error detection and non-designated local recovery servers.

6.3.1 *Latency*

Figure9 (a) shows the latency for 100% delivery as function of increasing number of nodes in the network. The proposed framework has significantly lower latencies compared to the other two schemes when the node density is increased. The reasons for reduced latencies are two-fold: the advantage gained by having a local designated server as opposed to a non-designated which reduces the amount of data sent and the advantage gained by using out-of sequence forwarding but without the NACK implosion problem, which increases the spatial reuse in the network. The latency of the out-of-sequence with NACK scheme is significantly higher at higher node densities and increases at a much faster rate than the other two schemes because of the NACK implosion problem. Although, our *core* construction scheme uses out-of-sequence delivery, we piggyback the *A-map* of the core node along with the transmission of each packet which allows the other dependent nodes to wait for the core to recover from all losses prior to any retransmission requests thus eliminating the NACK implosion problem.

6.3.2 *Number of Data Sent*

The number of data sent for all three schemes are presented in Figure 9 (b). Among the three schemes, GARUDA performs the best followed by the in-sequence with NACK and the out-of-sequence with NACK schemes. The number of packets sent in GARUDA is about 10% lower than that of in-sequence with NACK scheme for node density of 400, 600 and 800 and 55% to 80% lower when compared with out-of-sequence with NACK scheme. The reason for significantly better performance of GARUDA is again mainly due to the improvement gained by having a designated recovery server as opposed to a non-designated server and the *A-map* structure propagation.

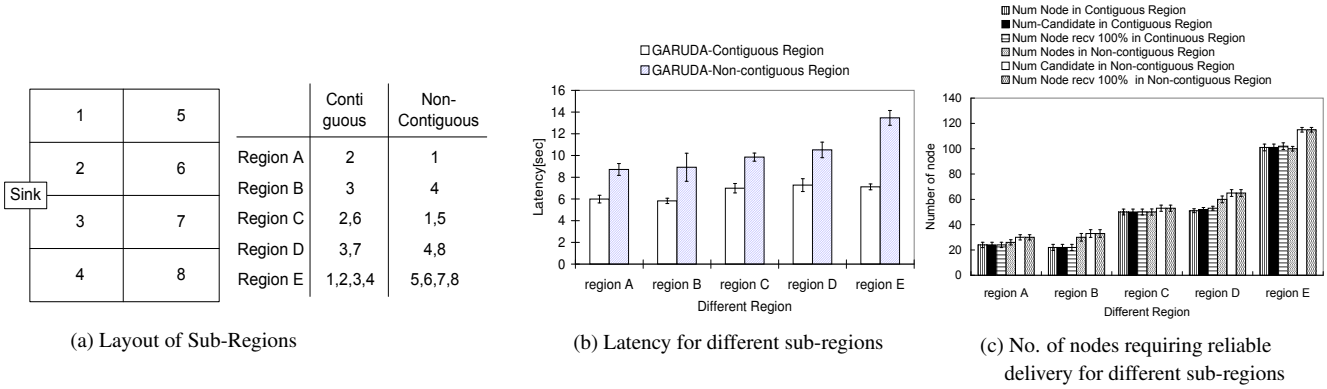
6.3.3 *Energy Consumption per Node*

The average energy consumed per node is significantly smaller for GARUDA when compared to the other two cases (Figure 9 (d)). The average energy consumed for all three cases is directly proportional to the number of transmissions, which is the sum of the number of requests sent and the number of data sent per node. Since the sum of the number of requests and data sent is the least for GARUDA, the energy consumed per node is also significantly less. In fact, results indicate that the energy consumed per node is about 30% less compared to the in-sequence case and about 80% less compared to the out-of-sequence scheme for 800 node scenario.

6.4 Evaluation of Variants

6.4.1 *Reliable Delivery within a Sub-region*

Figures 10(a)-(c) present performance results for the first variant for a 200 node, 650mx650m network with a transmission range of 67m per node. Figure 10(a) shows the partitioning of the network grid into sub-regions. Figure 10(b) shows the latency in-



(a) Layout of Sub-Regions

(b) Latency for different sub-regions

(c) No. of nodes requiring reliable delivery for different sub-regions

Figure 10: Performance evaluation of GARUDA: reliable delivery to all sensors in a sub-region

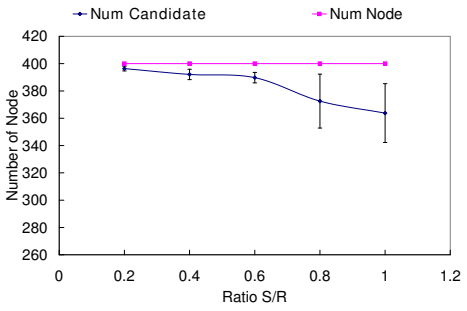


Figure 11: Performance evaluation of GARUDA: reliable delivery to minimal number of sensors in a region

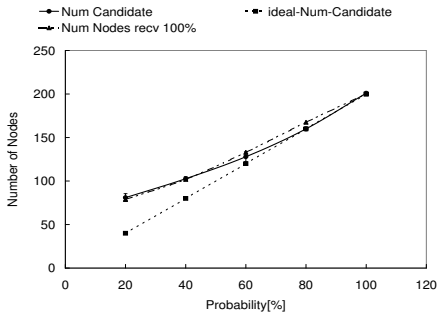


Figure 12: Performance evaluation of GARUDA: probabilistic reliable delivery

curred with increasing number of regions for both contiguous and non-contiguous regions respectively. While it is obvious that the latency increases with increasing number of regions, an interesting observation is that they latency for the non-contiguous regions scenario is always more. Recall that this is due to the latency involved in non-candidates being forced into candidacy. Figure 10(c) shows the number of data packets transmitted for the same scenarios. For the contiguous regions scenario, the achieved number of candidates is typically very close to the ideal number of candidates. However, for the non-contiguous regions, the achieved numbers are typically higher due to the forced candidacy of nodes to achieve connectivity.

6.4.2 Reliable Delivery to Minimal Set of Sensors

Figure 11 shows the number of nodes selected as candidates for the second variant. It can be observed that the number of nodes chosen decreases with increasing ratio $\frac{S}{R}$. The decrease is not much for the smaller values of $\frac{S}{R}$ because for the scenario considered (400 nodes in a 650mx650m grid with a transmission range of 67m), the minimum value for $\frac{S}{R}$ required to cover the entire area is approximately 0.5. As the ratio of $\frac{S}{R}$ increases beyond 0.6, we see a more pronounced decrease in the number of candidate nodes. This is because the overlap area among nodes become more pronounced as the sensing range approaches the transmission range.

6.4.3 Reliable Delivery to Probabilistic Subset

Figure 12 presents simulation results for the third variant. The scenario considered is 200 nodes in a 650mx650m grid, with nodes having a transmission range of 67m. The number of candidate nodes chosen with increasing probability is shown. It can be seen that at lower probabilities, the achieved number of candidates is larger than that of the expected number due to the forced candidacy of nodes to achieve connectivity. However, for larger probabilities ($\geq 50\%$), the achieved number of candidate nodes closely approximates the ideal values.

7. RELATED WORKS

To provide reliability, researchers have proposed several approaches at the different protocol layers including: (i) physical/link layer approaches, such as Forward Error Correction (FEC) [1], (ii) MAC layer approaches, such as reliable MAC [13], and (iii) transport layer approaches, such as reliable multicast [7] and reliable transport protocol [15, 12].

- [7] are reliable multicast approaches specifically designed for wired or multi-hop wireless environments which assume an address-centric routing layer and global unique node identification. Since wireless sensor networks require a data-centric routing layer without global identification, such approaches cannot be applied directly to wireless sensor networks.

- FEC has been an appealing approach to prevent feedback implosion that can happen when performing a large scale reliable multicast [1]. However, [8] evaluates the utility of FEC for reliable multicast and compares the effectiveness of FEC

with that of a subcasting[14] enabled multicast. [8] argues that FEC provides little benefit for an efficient reliable multicast protocol like [7] that uses subcasting. Since wireless sensor networks inherently support local subcasting because of the shared nature of the wireless channel, the gain of FEC in wireless sensor networks can be argued to be minimal. Currently the effect of FEC in wireless sensor networks is being evaluated.

- Several works have been proposed to perform efficient flooding in multi-hop wireless networks [16]. [16] classifies some of these approaches as probability-based, area-based, and neighbor-knowledge based schemes. While such approaches improve the successful delivery rate of messages, they still cannot guarantee any strict reliability semantics that the proposed framework supports. Such approaches in fact can be used in tandem with the proposed framework.

- PSFQ [15] is a transport layer protocol that addresses the issue of reliability in sensor networks. The key idea in the design of PSFQ is to distribute the data from a source node by transmitting data at a relatively slow speed, but allowing nodes that experience losses to recover missing data packets from immediate neighbors aggressively. However, PSFQ does not provide any reliability for single packet messages as it uses a pure NACK based scheme. Also, it uses in-sequence forwarding for message delivery to accomplish the pump slowly operation. This results in the wastage of precious bandwidth as shown in Section 6.

- Approaches such as [12, 9] that focus on upstream reliability as opposed to downstream reliability are clearly orthogonal to the focus of this work.

8. CONCLUSIONS

In this paper, we have proposed a new framework for providing sink-to-sensors reliability in wireless sensor networks. We have identified several challenges to provide sink-to-sensors reliability and addressed the challenges by proposing key elements: (1)Wait-for-First-Packet (WFP) pulse, (2) *core* structure approximating the minimum dominating set, (3) instantaneously constructible optimal *core* structure, (4) availability bitmap, and (5) two-stage recovery process. Note that, although we have proposed an effective way to realize the WFP pulse in-band, it is equally possible to use out-of-band signaling in scenarios where a pilot radio is available. We have also identified three new types of reliability semantics unique to downstream sensor environment and elaborated how our proposed framework can provide reliability to such variants. We have shown through ns2-based simulations that the proposed framework performs significantly better than the basic schemes proposed thus far in terms of latency and energy consumption. We have also profiled the A-map overhead in GARUDA and observed it to be minimal. We have also studied how the mechanisms in GARUDA can handle node failures. However, due to lack of space, we do not present these studies. Our future directions of work include extending the proposed framework to environments with mobility and in the presence of multiple sinks.

[14]Subcasting is a functionality that involves multicasting of a retransmitted packet by a loss recovery server over the entire subtree rooted at the server. Hence, all instances of that lost packet within the subtree are recovered by the single subcast.

9. REFERENCES

[1] BYERS, J. W., LUBY, M., MITZENMACHER, M., AND REGE, A. A digital fountain approach to reliable distribution of bulk data. In *Proceedings of the Special Interest Group on Data Communications (ACM SIGCOMM)* (Oct. 1998), pp. 56–67.

[2] F.HARARY. *Graph Theory*. Addison Wesley Publishing Co., Oct. 1969.

[3] FLOYD, S., JACOBSON, V., LIU, C., MCCANNE, S., AND ZHANG, L. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transaction on Networking 5*, 6 (Dec. 1997), 784–803.

[4] IEEE STANDARDS BOARD. 802 Part 11: Wireless LAN Medium Access Control (MAC) and physical layer (PHY) specifications, Mar. 1999.

[5] INTANAGONWIWAT, C., GOVINDAN, R., AND ESTRIN, D. Directed Diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the international conference on Mobile Computing and Networking (ACM MOBICOM)* (Aug. 2000), pp. 56–67.

[6] JOHNSON, D. S. Approximation algorithms for combinatorial problems. In *Journal of Computer and System Sciences 9* (1974), pp. 256–278.

[7] LI, D., AND CHERITON, D. R. OTERS (On-Tree Efficient Recovery using Subcasting): A reliable multicast protocol. In *Proceedings of the International Conference on Network Protocols (ICNP)* (Oct. 1998), pp. 237–245.

[8] LI, D., AND CHERITON, D. R. Evaluating the utility of FEC with reliable multicast. In *Proceedings of the International Conference on Network Protocols (ICNP)* (Nov. 1999), pp. 97–105.

[9] SANKARASUBRAMANIAM, Y., AKAN, O. B., AND AKYILIDIZ, I. F. ESRT: Event-to-Sink Reliable Transport in wireless sensor networks. In *Proceedings of the international symposium on Mobile Ad Hoc Networking and Computing (ACM MOBIHOC)* (June 2003), pp. 177–188.

[10] SAVVIDES, A., AND SRIVASTAVA, M. B. A distributed computation platform for wireless embedded sensing. In *Proceedings of the International Conference on Computer Design (ICCD)* (Sept. 2002), pp. 220–225.

[11] SIVAKUMAR, R., SINHA, P., AND BHARGHAVAN, V. CEDAR: A Core-Extraction Distributed Ad hoc Routing algorithm. *IEEE Journal on Selected Areas in Communications (Special Issue on Ad-hoc Routing) 17*, 8 (Aug. 1999), 1454–1465.

[12] STANN, F., AND HEIDEMANN, J. RMST: Reliable data transport in sensor networks. In *Proceedings of the international workshop on Sensor Net Protocols and Applications (SNPA)* (Apr. 2003), pp. 102–112.

[13] TANG, K., AND GERLA, M. Mac reliable broadcast in ad hoc networks. In *Proceedings of the conference on Military Communications (MILCOM)* (Aug. 2001), pp. 1008–1013.

[14] VAZIRANI, V. V. *Approximation Algorithms*. Springer, May 2001.

[15] WAN, C.-Y., CAMPBELL, A., AND KRISHNAMURTHY, L. PSFQ: A reliable transport protocol for wireless sensor networks. In *Proceedings of the international Workshop on Sensor Networks and Arch. (WSNA)* (Sept. 2002), pp. 1–11.

[16] WILLIAMS, B., AND CAMP, T. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the international symposium on Mobile Ad Hoc Networking and Computing (ACM MOBIHOC)* (June 2002), pp. 194–205.