This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE SECON 2006 proceedings.

Solicitation-based Forwarding for Sensor Networks

Seoung-Bum Lee, Kyung Joon Kwak
Electrical Engineering, Columbia University,
New York, New York, USA
{sbl, kjkwak}@ee.columbia.edu

Andrew T. Campbell
Computer Science, Dartmouth College
Hanover, New Hampshire, USA
campbell@cs.dartmouth.edu

Abstract- Delivering sufficient fidelity to sensor network applications is challenging because unpredictable wireless links, network dynamics, and the presence of transitional regions in sensor networks impact the delivery of packets (i.e., fidelity) to the sink. One of the major reasons for this problem is due to the non-responsive nature of forwarding mechanisms commonly implemented in experimental sensor networks. Existing routing protocols implemented in these networks typically base their forwarding decision-making on some form of statistical observations regarding past communications or the quality of past beacon signals received by communicating nodes. This approach fails, however, to capture the link conditions at the exact time of forwarding packets across the wireless link, limiting the aggregate forwarding capability of the network. In this paper, we argue that the forwarding decision of a sensor device should be based not on historical information but on the instantaneous link conditions at the exact time of packet communications, and propose, *solicitation-based forwarding (SOFA)*, a highly-responsive hop-by-hop routing protocol that results in increased application fidelity. SOFA represents a cost-effective, on-demand scheme that makes use of simple solicitation-based handshakes between a sender and multiple potential receivers at each wireless hop to negotiate the best forwarding path to a target destination (i.e., sink) when events occur in the sensor field. We present the detailed design, implementation, and experimental evaluation of SOFA in a 36-node Mica2 testbed using TinyOS, and discuss its measured performance benefits in comparison to the TinyOS standard routing protocol widely used by the experimental sensor network community.

Keywords-sensor network; routing; responsive forwarding; fidelity

I. INTRODUCTION

Recent technological advances in wireless communications make it possible for low cost, low complexity sensor networks to monitor and to detect environmental and tactical events. Sensor devices are typically equipped with a low power communication transceiver and a limited processor to facilitate signal processing. Because a sensor network can be deployed anywhere, even in areas where accessibility is limited, it is suitable for many emerging applications. One class of widely deployed applications is event-driven applications that are used to detect and report important events that occur in a sensor field. This type of application offers minimal traffic load and spends most of its time in an idle state. When an event is detected, the network becomes active and generates temporally and spatially correlated information that needs to be delivered to the sink. Since an event may be short-lived, the burst of information the network generates/senses during this time is likely to be of most importance to the application. A sensor network is therefore tasked to deliver a sufficient amount of information within a bounded time, i.e., fidelity [1]. However, numerous technical challenges hamper the delivery of adequate fidelity at the sink points in sensor networks. One of technical barriers to supporting sufficient fidelity comes from network dynamics. Network dynamics appear in various forms, e.g., wireless error, node failure, or anything that unexpectedly impedes on-going communications. Even when conducting indoor experiments, we often observe that only a fraction of the generated events are delivered to the sink due to the observed network dynamics. The presence of transitional regions [2], packet collisions, the funneling effect [10], and congestion [10] further limits the performance of sensor networks. A transitional region comprises highly unpredictable links with intermittent and asymmetric connectivity, which present significant networking problems. Sensor networks often exhibit non-isotropic radio ranges [3] and comprise asymmetric and unidirectional links. These conditions impair support of adequate levels of fidelity because link-layer reliability (or goodness of the link) is typically perceived through signaling exchanges or overhearing between participating nodes.

Adequate fidelity requires that event flows are routed through the "good-conditioned" nodes that form paths to the sink. The term good-conditioned may represent the energy-reserve of a sensor node, congestion status, routing distance, or any characteristic that correlates positively with the ability to deliver information to the sink. Sensor networks need cost-effective mechanisms to exploit these better-conditioned nodes to deliver information. Responsive self-configurability is another key property for fidelity support in sensor networks. A sensor network should be able to configure itself quickly and facilitate information delivery as soon as it is deployed. Moreover, a sensor network should be able to quickly respond to changes in network topology. A sensor network should also be responsive to nodes that fail over time which typically alter the connectivity graph of the network. Therefore, the delivery path needs to quickly reflect any observed changes in the topology and quickly adapt its delivery path to sustain event flows of information to the sink. Similarly, when new sensors are added to existing networks, they should be quickly integrated into the network with minimal overhead. Many of the existing routing protocols implemented in experimental sensor networks are not responsive to these challenges. Rather they incur a large control overhead, and lack the agility to cope with network and link dynamics (i.e., node failure, packet loss, link loss, new nodes, etc.). As a result this significantly impacts the fidelity of the delivered signal to the sink and sensor applications. To address these issues, we propose a new routing algorithm called *solicitation-based forwarding (SOFA)*. Through expensive Mica2 mote testbed experiments, we show

that the on-demand nature of SOFA makes it cost effective, and responsive to network dynamics while supporting improved fidelity at the sink in comparison to existing experimental sensor network routing protocols [5] [8].

The structure of the paper is as follows. Section II presents networking problems that motivate our proposal. The related work is presented in Section III. This is followed by a detailed description of SOFA's operations in Section IV. Section V presents the experiemental evaluations of SOFA followed by concluding remarks.

II. FORWARDING PROBLEMS IN SENSOR NETWORKS

In what follows, we discuss a number of forwarding problems found in experimental sensor networks, which motivate the design of SOFA. We use results from a set of experiments conducted on an experimental 36 Mica2 [9] mote testbed arranged in a dense 6x6 grid topology to quantitatively study forwarding problems in experimental sensor networks. The testbed software comprises the standard release of TinyOS [8], the Surge application, the MultiHopRouter [5] routing protocol, which is based on link quality estimation, and B-MAC [5]. Link quality estimation requires nodes to periodically broadcast beacon signals to create and manage per-neighbor statistical records of past communications that are used when evaluating link quality and making forwarding decisions at sensor nodes. Although these proactive approaches generally provide good routing paths for a stable network, they also present a number of limitations. First, they are cost-ineffective because they require all nodes to periodically exchange broadcast messages regardless of the level of network activity. Any transmission/reception consumes energy and bandwidth. The smaller the amount of sensor and control traffic in the network, the less energy consumed and probabilistically less collision observed. Therefore, it is important to keep the control overhead to a minimum in energy-limited sensor networks. Link quality estimation requires periodic signaling (or beaconing) and continuously consumes energy even when the network is in an idle state. This is counterintuitive because maintenance of unutilized paths only wastes energy.

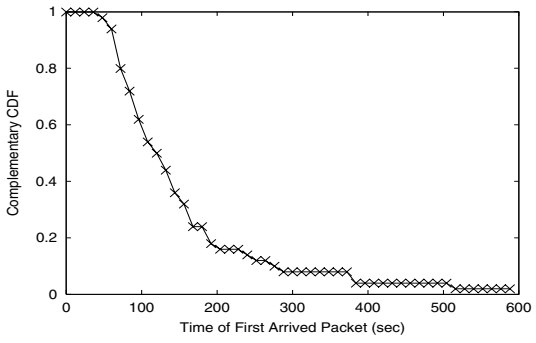

Figure 1. Path convergence of the proactive MultiHopRouter routing protocol

Creating a forwarding path based on a statistical record may be time-consuming because a relaying node (i.e., parent) has to be determined at each wireless hop and these piecewise decisions take time to converge. Consequently, path convergence between a sensor and a sink often takes a substantial amount of time, preventing sensor devices from immediately reporting on-going phenomena after deployment. In our testbed, when using the default MultiHopRouter [5] routing protocol distributed with the TinyOS release the path convergence often requires several minutes and scores of routing message exchanges. Figure 1 presents an example of the path convergence distribution observed for 50 different experiments. Each experiment lasts for 30 minutes where we record the time to deliver the first packet to the sink, representing the path convergence time. A source node begins transmitting its data as soon as it powers on. The transmitted data from most source nodes is lost for an arbitrary period of time along partially constructed paths because the path to the sink is not completely resolved. Due to slow path convergence time, only 6% of source nodes achieve their path convergence within 60 seconds, approximately 50% in 120 seconds and the remainder spans up to the 10th minute. Such path convergence behavior exhibited in sensor networks poses a serious technical barrier for many applications because information delivery is preceded by a long settling time after network deployment or network dynamics, such as, link or node failure. Similar problems are observed when new sensors are added to an operational network. Typically, large convergence times are experienced when integrating sensors into a network. Similar path convergence issues occur when node failure occurs (e.g., energy depleted node) on a forwarding path, where the impact may last for a long period of time because it requires multiple samples to detect the loss of a next hop node and even more samples to acquire a replacement next hop node. During this time data packets may be continuously sent only to be lost. From our testbed results, we observe that the impact of node failure typically lasts for 3~5 minutes, and in the worst case the forwarding path never recovers (see Section V B for a detailed discussion).

Another drawback of these proactive routing approaches is that they often fail to reflect link conditions at the exact time of the actual transmission. Events are rare in sensor networks and when an event occurs, a burst of information (i.e., an event train) is generated toward the sink node. However, estimation of link quality based on statistics from the recent exchange of periodic messages between nodes may not reflect the actual conditions when a burst of data traffic arrives at a link but is estimated when the burst of data is not present in the network. Therefore, it is likely that the link quality does not represent the actual condition when the data needs forwarding. We argue that forwarding decision should be made when the actual data is ready to traverse the wireless link. In other words, we argue that past measurements may have little relevance, particularly if they reflect past statistical states gathered under different conditions (e.g., idle state) from the actual data transmission.

The combination of path convergence and link quality estimate issues can substantially impact the overall performance of beacon-based proactive routing protocols [5]. Unless these forwarding problems are resolved, they limit the applicability of a sensor networks to a small number of simple low-fidelity applications (e.g., periodic reporting). Figure 2

shows a trace of a monitored event flow that encounters two route changes resulting from network dynamics (i.e., node failures in this example). As shown in Figure 2, the monitored event flow requires approximately 9 minutes for path convergence and the two re-routing conditions interrupt the event delivery for 4 and 12 minutes, respectively. Therefore, the event flow encounters an aggregate disruption duration of approximately 25 minutes. This constitutes about 40% of a testbed runtime (i.e., 60 minutes). The main reason for such poor performance is associated with the link quality update interval. At low data rate, with intermittent collisions, nodes often do not resolve a valid relaying node, resulting in lengthy disruption of information delivery. This shortcoming can be somewhat improved if the frequency of routing message is increased but only at the cost of substantially increased control overhead.

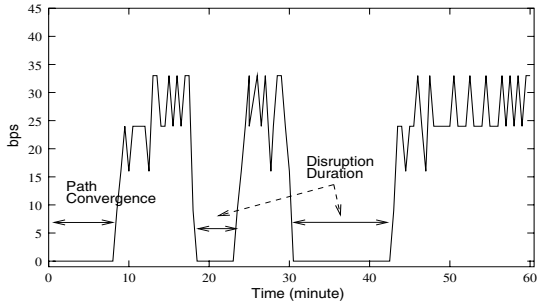

Figure 2. A monitored flow is traced at the sink to capture the impact due to slow path convergence time and node failures

III. Related Work

There are a number of routing protocols for sensor, mesh, MANET networks found in the literature. We first discuss the routing protocols released as part of the TinyOS software, and then discuss some relevant routing protocols for mesh and MANET networks. The TinyOS MultiHopRouter [8] protocol, which is widely used by the sensor network community, is based on the shortest-path algorithm that forms a spanning tree so that the path from any mote in the sensor field to the sink uses the least number of hops. Route control messages are periodically broadcast from each node in the network to estimate the routing cost and monitor link quality. The TinyOS Mintroute [5] protocol represents an adaptation of the simple MultiHopRouter protocol. MultiHopRouter uses the least number of hops as the primary metric with link quality as a tiebreaker, whereas, Mintroute uses the link quality with surrounding neighbors together with a cumulated route quality to the sink, ignoring the hop count in the route updates.

A number of researchers have revisited the design of routing protocols for mesh and sensor networks based on realistic wireless channel models founded on experimentation. In [13], the authors observe that the minimum hopcount without consideration of the channel characteristics shows poor performance. Destination sequenced distance vector (DSDV) [4] routing is a proactive routing algorithm that has influenced the implementation of TinyOS routing protocols. DSDV provides many-to-one routing to one destination at a time and can be used with either a hopcount metric or a quality metric. There are several geographical routing algorithms found in the literature that are applicable to sensor networks. Each node in greedy perimeter stateless routing (GPSR), for example, maintains a neighbor table that is updated via periodic beacon exchanges. However, these beacon messages constitute a large overhead for resource limited sensor networks. Other representative protocols in this class [15] [16] overcome the limitation of using periodic beaconing but still require some form of geographic coordinates provided by GPS for their operations. The geographic random forwarding protocol [17] represents one of the more sophisticated geographic routing solutions for sensor networks but its use of busy tones [6] makes it impractical to implement using standard sensor networking technology available today. There is a large body of work on MANET routing protocols [11] that has influenced our thinking, for example, the idea of "height" discussed in the next section is reminiscent of the TORA [12] routing protocol. However, these protocols are typically far too complex and costly in terms of control overhead to consider feasible for sensor network implementation.

IV. SOFA Design

In what follows, we present the detailed design of the SOFA protocol.

A. Protocol Overview

SOFA establishes a path from a sensor to the sink based on hop-by-bop forwarding decisions by selecting appropriate relaying nodes at each wireless hop. A chain of relaying nodes composes the path to a sink. Each forwarding decision uses solicitation-based handshakes between a sender sensor and potential acceptors (i.e., next hop relaying nodes), where preference is given to the "best-conditioned" nodes as a relaying node at the time of packet communications. SOFA comprises four protocol phases; there are, *solicitation*, *acceptance*, *data-send*, and *passive acknowledgement*. In the solicitation phase, a solicitor seeks out a relaying node among its neighboring nodes by broadcasting a solicitation message called *solicit-to-forward (STF)*. A neighboring node that is nearer to the sink receiving the STF accepts the solicitation by generating an *accept-to-forward (ATF)* message as long as it hears no other node has already responded to the STF. Once the solicitor node finds an acceptor, the accepting node becomes the *designated next hop (DNH)* for the solicitor node and solicitor node can send data to its DNH. Note, that the DNH is established on an on-demand basis and reflects the best link toward the sink at the time of data transfer across the link. In this sense the link is only assessed at the time of transmission and not continuously/periodically, which is the case for the link estimation schemes discussed in Section II. The maintenance of the DNH is based on soft-state where the timer is associated with the event flow time-scale; that is, the DNH is kept active for a period of time to allow events to drain to the sink. After the soft-state timeout period the solicitor node would need to determine its DNH again. The thinking behind this is that the link quality may change after a certain period of time and the solicitor needs to determine its new DNH. This is triggered on an on-demand basis when the next event/data

packet needs forwarding and is not assessed during the period when there is no data to transmit to the sink.

SOFA uses a passive acknowledgement mechanism, which means when a solicitor node overhears the forwarding of its data packet by its DNH it assumes reliable delivery has taken place. In the case it does not overhear the forwarding operation it can retransmit the original data packet if the application requires such a level of reliability. All transmitting nodes require a DNH and once a DNH is selected, data is unicast in a "distance-decreasing" direction toward a sink through a chain of DNH nodes. This is analogous to water flowing from higher to lower ground if we consider distance as height. We use the term *height* to represent the distance of a sensor node to the sink.

B. Height Initialization

During the height initialization phase, each node learns its height through sink-generated sink advertisement messages traversing the network. All messages (i.e., advertisements or any application query messages) that originated from a sink have a height of zero. As these messages propagate through the network their height information is incremented by one at each hop to reflect relative distance from the sink. Note, that height information of the sender is piggybacked in each message header. The sink node also has options to rebroadcast messages to update height information and re-advertise its existence. Any remote node or newly joined node that fails to receive advertisements would acquire its height through the height acquisition procedure described in Section IV E.

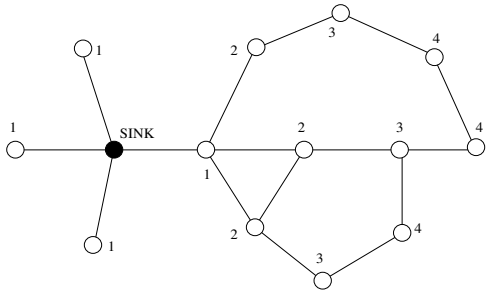

Figure 3. Height Initialization: sink advertisement floods the network and each node learns their distance to the sink.

C. Solicitation based Handshake

When a height-aware sensor node has data to transmit, it first checks whether it has a DNH. If it has a DNH, data can be immediately transmitted via its DNH, otherwise, the sender needs to acquire a DNH through solicitation-based handshake. A solicitation-based handshake starts when a DNH seeking node (i.e., solicitor) broadcasts an STF. When a node receives an STF, it first compares the height advertised in the STF with its local height and proceeds with an ATF response only if its height is less than that of the soliciting node (i.e., STF sender) to guarantee that the DNH is closer to the sink. In this context, neighboring nodes with small heights are termed next hop candidates.

Figure 4 shows an example of the solicitation-based handshake procedure. In Figure 4(a), node A has data to send but does not have a DNH, thus, node A broadcasts an STF to solicit a designated next hop. In this example, node B, node C, and node D are next hop candidates. As shown in Figure 4(b), node B is the first node to respond to the STF with an ATF. Note, that node D overhears the ATF response from node B and discard its pending ATF transmission. In other words, only one node within a common radio range responds to an STF. This mechanism provides a means to limit the number of ATF responses by permitting only a subset of next hop candidates to respond to single STF.

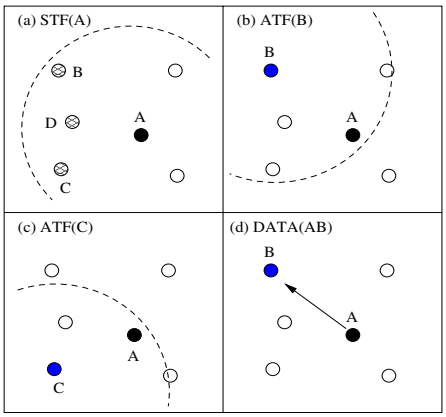

Figure 4. STF, ATF and DATA exchanges

In Figure 4, node C is outside the radio range of node B and unaware of the ATF response from node B. Therefore, node C also transmits an ATF to node A and as a consequence node A receives two ATF responses. In this example, node B is selected as the DNH of node A (DNH_A) because the ATF from node B reaches node A first. Having acquired a DNH, node A begins to unicast data toward the sink. The solicitation-based handshake completes when node A overhears node B (DNH_A) transmitting the data. Once the solicitation-based handshake completes, subsequent data messages from node A are unicast to node B without STF-ATF exchanges. Node A maintains its DNH unless an erroneous condition is detected. SOFA considers a number of packet losses, energy depletion, or any form of forwarding failure as erroneous conditions. SOFA conservatively assumes that a packet is lost when it is not passively acknowledged. For example, node A assumes that transmitted data is not received by DNH_A (i.e., node B of Figure 4) when it does not hear DNH_A relay its data packet. When γ consecutive passive acknowledgements fail, SOFA executes a new solicitation to acquire a new DNH. In such a case, the old DNH is blacklisted for a temporary period until a new DNH is selected. Note, that a blacklisted node can be reselected as a DNH if no other next hop candidates exist (i.e., no other ATF). If the solicitation process fails consecutively ϕ times, this may indicate the height of the soliciting node may be a local minimum (i.e., the node has no next hop candidates). This condition may arise when its DNH node expires or the network topology changes. SOFA resolves this situation through a height healing algorithm. Note, that both γ and ϕ can be tuned to make SOFA conservative or more aggressive.

Details on height-healing and other height maintenance algorithms are discussed in Section IV.E.

D. ATF Response and Defer-Time

One of the important features of SOFA is that ATF responses reflect current node conditions such that a DNH is less likely to be selected from problem-prone nodes. This mechanism is realized by coupling node conditions to the *defer-time* where defer-time is an additional waiting time that precedes an ATF response. Each next hop candidate receiving an STF delays its ATF response for the duration of its local defer-time. A node with a problematic condition would have a non-zero defer-time while a node in a better-condition would introduce no defer-time. This allows the STF sender to receive an ATF response from a better-conditioned node first. Equation (1) describes the defer-time of SOFA.

$$\text{defer\_time} = \{\text{random}\% \, CW_{SOFA}\} \cdot \text{slot\_time} \quad (1)$$

Note, that CW_{SOFA} (SOFA's Contention Window) can be used to represent various node conditions. For example, Equation (2) reflects a node's energy reserve status and congestion status.

$$CW_{SOFA} = (1-\beta) \cdot \text{energy}_{SLOT} + \beta \cdot \text{congestion}_{SLOT} \quad (2)$$

Note, that it requires congestion or energy concerns to have a nonzero CW_{SOFA}. For example, when the energy reserves (i.e., $E_{CURRENT}$) of a node is below some predefined threshold value (i.e., E_{THRESH}), an additional energy_{SLOT} is added to CW_{SOFA}. These tunable system parameters are dependent on the hardware specifications and applications. Similarly, when congestion is detected at a node, the congestion_{SLOT} is added to CW_{SOFA}. The weight factor β is a system parameter to control the sensitivity of these metrics on the node's CW_{SOFA}. For example, when $\beta = 1$, CW_{SOFA} only reflects congestion condition whereas when $\beta = 0$, only energy.

Figure 5 captures the impact of the defer-time in DNH selection. We construct a 9-node network using the *ns-2* simulator [7] and observe DNH selection using (2). For simplicity, we set $\beta = 0.5$ and the congestion_{SLOT} to either 0 (i.e., no congestion) or 100 (i.e., congestion). The energy_{SLOT} is assigned with respect to a node's energy reserve. The simulation comprises one sink, one source node with height of 2, and seven intermediate nodes with height of 1. The next hop candidates (i.e., intermediate nodes) are assigned with diverse initial energy (i.e., node 7 = 10 J, node 3 = node 5 = 8 J, and the rest with 4 joules). The E_{THRESH} is set at 10 joules so that a non-zero defer-time always precedes an ATF response. Congestion is introduced randomly into the network in the first 50 seconds of the simulation run. With the source node transmitting 5 STF/second, we plot the corresponding DNH selections in Figure 5.

The y-axis represents the node number of the selected DNH and the x-axis represents simulation time. Clearly, the three energy abundant nodes (i.e., node 3, 5, and 7) are predominantly selected as a DNH. Figure 5 also show that the existence of congestion in the first 50 seconds also has an impact on the DNH selection (i.e., DNHs are more distributed). In the first 70 seconds on the trace, node 7 is mostly selected as a DNH because it has the most energy but as energy reserve of node 7 decrease, node 5 and node 3 start to be selected as DNH. This result shows that the defer-time can effectively expose node conditions (e.g., energy, congestion, etc.) to the instantaneous forwarding decisions. Note, that in real testbed experiments discussed later in the paper, only congestion condition is utilized because the energy-reserves are not accessible when using the Mica2 motes.

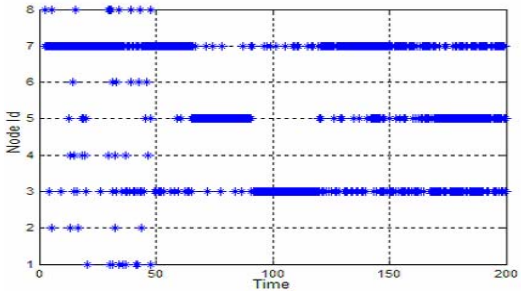

Figure 5. Impact of defer-time on DNH selections

E. Height Maintenance

SOFA implements three simple height maintenance algorithms: height healing, height rollback, and height acquisition. Height healing resolves deadlock conditions, height rollback optimizes the height information, and height acquisition provides height information for a null-height node. In what follows, we discuss the algorithms.

1) Height Healing Algorithm

Height healing is executed when the height of a node becomes a local minimum and finds no next hop candidates. Such a node is dubbed a "sinkhole". A sinkhole can become a DNH but never finds its own DNH. Although this condition rarely occurs, its impact is significant because all traversing packets are discarded at the sinkhole. Absence of a DNH would prompt re-solicitations but a sinkhole has no next hop candidates to acquire a DNH unless this anomaly is resolved. The only way to correct the condition is to increase the sinkholes height by one. This procedure is termed height healing. In general, height healing is preceded by multiple re-solicitation failures after a DNH is lost (i.e., unreachable). From our experimental results, most height healings is successful after one or two iterations.

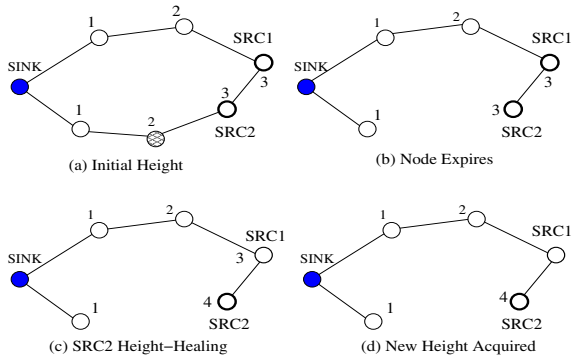

Figure 6. Height Healing Illustration

Figure 6 illustrates an example of the height healing algorithm. As illustrated in Figure 6(a), SRC1 and SRC2 are sending data to the sink. The initial heights of SRC1 and SRC2 are 3 in the example. As shown in Figure 6(b), SRC2 loses its DNH (e.g., due to node expiration, node failure) and data forwarding is suddenly disrupted. This condition is detected by SRC2 through the continuous lack of passive acknowledgements; therefore, the soliciting node assumes that its DNH is no longer reachable. SRC2 re-solicits for a new DNH but fails to acquire a DNH because its height is a local minimum. When the solicitation attempts continuously fail, SOFA identifies SRC2 as a sinkhole and performs height healing to resolve the anomaly. As illustrated in Figure 6(c), SRC2 increases its height by one and retries solicitation with the new height of 4. After a successful handshake of STF-ATF, SRC1 becomes the new DNH of SRC2 and data packets can now be forwarded toward the sink.

2) Height Acquisition Algorithm

Another important feature of SOFA is that it transparently integrates new nodes into existing operational networks. Newly joining sensor nodes do not have height information nor have knowledge of the sink. New nodes are considered to have null heights. There are two ways to acquire a valid height in SOFA. A new node can learn its height when it overhears any transmission from its height-aware neighbors. Its initial height becomes $\{h_{FIRST}+1\}$, where h_{FIRST} is the height of first overheard packet. The initial height is optimized through height maintenance algorithms. On the other hand, when a null height node wants to send data to the sink, it executes a height acquisition routine called "rippling". Rippling involves transmission of an STF with null height (i.e., STF_{NULL}). When a height-aware node receives an STF_{NULL}, it generates a normal ATF with a height. Upon receiving ATFs from its neighboring nodes, the null-height node selects the minimum-height node as its DNH and sets its height to $h_{DNH}+1$ where h_{DNH} is the height of its DNH. However, when an STF_{NULL} is received by another null-height node (e.g., observed when cluster of new sensors are deployed) no ATF reply is sent; instead null-height nodes rebroadcast STF_{NULL} until it reaches a height-aware node. Each rippling node sets its ripple_flag and ATF response from the height-aware node backtracks to the STF_{NULL} originator through the path with the ripple_flag set. When the rippling phase completes, all associated null-height nodes becomes aware of their heights. Figure 5 illustrates the rippling case where node A, B and SRC2 represents newly joined nodes. SRC2 needs to report a detected event but lacks height information because it has not heard any transmissions from its height-aware neighbors. This condition triggers SRC2 to broadcast an STF_{NULL} and the broadcast message is received by node A and B. However, node A and node B cannot respond to the STF because they are also null-heighted nodes. In this case, node A and node B rebroadcast the STF_{NULL} (i.e., rippling). The process is repeated until an STF_{NULL} is received by a height-aware node. In this example, SRC1 and the sink respond to the rippled STF_{NULL} with ATFs. All ATFs piggyback the height information of the transmitter, as illustrated in Figure 7. Upon reception of ATFs, node A and node B learns their height. Since node A and node B have their ripple_flag set, they relay the ATF with their newly acquired height information. When SRC2 receives an ATF, the rippling routine completes and SRC2 acquires height.

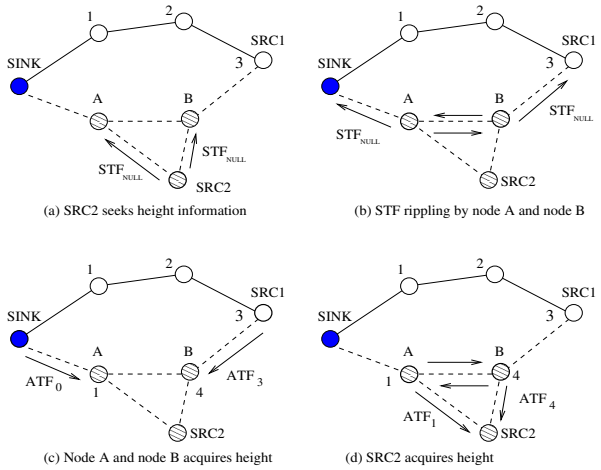

Figure 7. Height Acquisition Example

3) Height Rollback Algorithm

Height rollback is the counterpart of height healing algorithm that adjusts height to a smaller value. Height rollback is used to correct transient height sub-optimality in a sensor network. Typically, height sub-optimality is observed when new sensors are added into the existing network, altering the network topology. In such event, a node sometimes finds a new DNH with smaller height indicating that the new path is shorter in distance than the previous path. Soliciting nodes with height $h_{SOLICITOR}$ learns this condition when the height difference with its DNH node is greater than one (i.e., $h_{SOLICITOR} - h_{DNH} > 1$). This condition triggers the soliciting node to adjust its height to $\{h_{DNH} + 1\}$. From Figure 7, when node B seeks a DNH it sends out an STF (i.e., $h_B = 4$). When node B receives an ATF from node A, it detects that the height difference is more than 1 and adjusts its height to 2 (i.e., $h_B = h_A + 1$). Note, that height rollback does not incur any additional control load.

V. Experimental Testbed Evaluation

A. Mote Testbed Setup

In this section, we discuss the implementation of SOFA on a real sensor network using TinyOS [8] on Mica2 motes [9]. The testbed comprises 36 Mica2 motes arranged in a dense 6x6 grid topology. Two additional motes are strategically placed as snooper nodes to monitor communication details not captured by the sink node. Node spacing and transmission power are set such that one-hop neighbors can deliver more than 80% transmitted packets, while two-hop neighbors deliver less than 20%. The data packet size is 36 bytes. We assume this experimental setting unless specified otherwise. We report detailed performance results of SOFA using B-MAC and compare it to MultiHopRouter[5] using B-MAC. MultiHopRouter is a routing protocol included in TinyOS for mote-based sensor networks where route control messages are periodically broadcasted from each node to estimate the routing cost and monitor link quality. We refer to the network

running MultiHopRouter as the "baseline system" in the remainder of the paper. In the testbed, we set γ (see Section IV.C) to be 3 and the solicitation limits to be 3 (i.e., $\phi = 3$). However, the γ value changes to 7 when the link-layer retransmission option is enabled. These values reflect the link-layer retransmission limits [7][11] and routing failure notification limits [7] [11] commonly adopted by the MANET [11] community.

B. Path Convergence Analysis

In this section, we compare path convergence of the SOFA system to that of the baseline system. We define path convergence of a flow as 'the time required for a packet to reach the sink for the first time'. In other words, the path convergence is, $t_{PATH\text{-}CONVER} = t_{FIRST\text{-}PKT} - t_{INIT}$, where t_{INIT} is the time when a source node generates a data packet for the first time and $t_{FIRST\text{-}PKT}$ is the time when a packet from the source reaches the sink for the first time.

Figure 8 shows the path convergence distribution of 50 different experimental cases in our 36-mote network. The x-axis of Figure 8 represents $t_{PATH\text{-}CONVER}$ and the y-axis represents the complementary CDF (cumulative distribution function). Figure 8 clearly shows that there is a significant difference in path convergences between the baseline and SOFA systems. In particular, 96% of SOFA's path convergences are accomplished within first 60 seconds while only 6% of path convergences are accomplished with the baseline system in the same window of time. In fact, the baseline system requires 372 seconds to achieve 96% of path convergences. The main reason for slow path convergence lies in its link quality update interval (i.e., periodic routing messages). At a low rate, with intermittent packet losses, nodes often fail to determine a valid relaying node. This can be somewhat improved if the routing update frequency is increase but only at the expense of substantially increased control overhead. For example, when the routing update frequency is doubled (i.e., update interval decreases from 20 seconds to 10 seconds), the average path convergence times improve by 37% for one of our baseline experiment but the corresponding control overhead increased by 200%. Moreover, increasing the update frequency may have a significant impact on fidelity because the increase in control load inevitably increases the collision probability and impairs the information delivery at the sink.

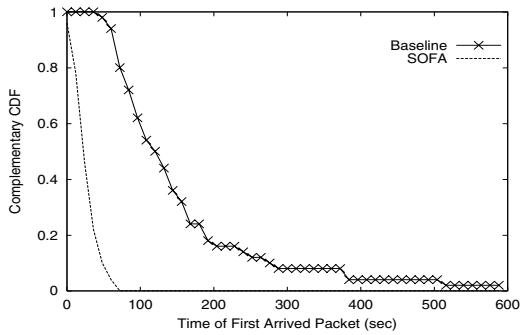

Figure 8. Path Convergence Time

C. Network Dynamics Analysis

In what follows, we discuss the impact of network dynamics on information delivery at the sink (i.e., the fidelity). Since network dynamics cannot be controlled with Mica2 motes, we create artificial node failures that arbitrarily discard to-be-forwarded packets. In this set of experiments, only one source is present in the 36-node network. We carefully introduce node failures on the forwarding path at t = 18 minutes and t = 30 minutes, and observed the event flow for 60 minutes. The same sets of experiments are repeated for the baseline and SOFA systems. Figure 9 captures throughput of two systems, as measured at the sink. Note, that the baseline system takes approximately 9 minutes for path convergence (i.e., (1) in Figure 9). With the first artificial node failure at t = 18 minute, information delivery is interrupted for 4 minutes (i.e., (2) in Figure 9) until a new forwarding node is selected. Similarly, the disruption of information due to the second node failure at t = 30 minute lasted for approximately 12 minutes (i.e., (3) in Figure 9). The baseline system is slow in recovering a path because its link quality estimation mechanism requires multiple routing packets to realize and resolve the node failure problem. Note, that the baseline system uses the default settings for MultiHopRouter where each node broadcasts a non-propagating link-quality update message every 20 seconds. Thus, the link quality estimation of a particular link has an evaluation resolution of 20 seconds. This implies that path changes can only be executed in the multiples of 20 seconds (i.e., routing update rate). In contrast, the path convergence of SOFA complete in 10 seconds and the source node immediately starts its information delivery to the sink. More importantly, the impact of node failure on the SOFA system is minimal. When node failure is detected (i.e., loss of 3 consecutive passive acknowledgements), SOFA re-solicits acquires a new DNH in a single handshake that involved one STF and 2 ATFs. However, SOFA also incurs four additional re-solicitations before the second node failure and six more after second node failure. Lack of passive acknowledgements is misinterpreted as a node failure which triggers re-solicitation. This implies that SOFA entails additional control overhead without actual node failure when faced with packet loss. In fact, SOFA can entail substantial control overhead in a lossy environment. We discuss this issue later in Section V.E.

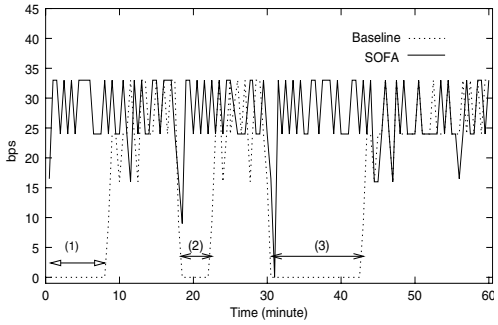

Figure 9. Comparison of a monitored flow

When the network is in a stable condition with minimal problems, both systems perform well delivering about 80% of generated information. Under these conditions, SOFA's advantages over the baseline system are limited to faster path

convergence, reduced overhead, and corresponding reduction in energy consumption due to reduced overhead. However, with the presence of any network dynamics, a greater disparity in performance begins to emerge. From Figure 9, the slow path convergence and two node failures constitute approximately 29% of total disconnected duration (i.e., no information is delivered during this period). In contrast, SOFA immediately achieves path convergence and the two node failures have virtually no impact on SOFA.

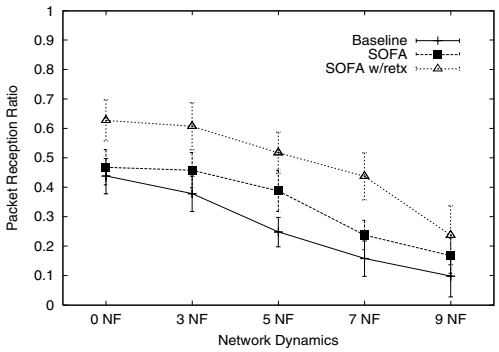

Figure 10. Comparison of Packet Reception Ratio

Figure 10 plots the packet reception ratio. Each point is an average of 10 experiments with 95% confidence interval. Note, that each testbed run lasts for one hour. The packet reception ratio (PRR) represents the ratio of number of packets received at the sink to the number of packets generated by source nodes. In this experiment, 6 nodes are randomly selected as source nodes. Their average height is 3 and their data rate is fixed at 1-packet/4-second. The x-axis represents the various degrees of network dynamics (as modeled in Section V.C). The x-axis ranges from 0 NF (i.e., no node failures) to 9 NF (i.e., 9 node failures). Note, that node failures are only introduced on the 29 non-source nodes (i.e., not the 6 source nodes and sink node). We do not show results beyond 9 node failures since many of the experiments fail to deliver any data packets when there is more than 10 node failures, due to lack of connectivity in our testbed.

With zero node failures, SOFA shows a PRR gain of 7% over the baseline system. The main reason for this improvement lies in the path convergence where the baseline system has an average path convergence time of 8 minutes. With SOFA, all path convergences complete within 3 minutes. The impact of node failures on the two systems is clearly shown in Figure 10. As more network dynamics are introduced, the packet reception ratio of two systems degrades accordingly. However, SOFA provides improvements over the baseline system under all conditions. When there are 5 node failures, the baseline system can only support a PRR of 25% indicating that more than 4000 packets are lost in the network. In contrast, the corresponding PRR for SOFA is 40%, an improvement of 60% over the baseline system is achieved. In general, the SOFA system performs much better in the face of network dynamics. Average disruption duration for SOFA system is 28 seconds whereas the average disruption duration of the baseline system is 240 seconds.

Figure 10 also shows the results of SOFA with link-layer retransmissions. As noted before, SOFA has an option to enable link-layer retransmissions. Consecutive retransmission failure triggers re-solicitation and when re-solicitations continuously fail SOFA-retx (i.e., SOFA with retransmission option enabled) assumes the node is a locally minimum in height and executes the height healing algorithm. As shown in Figure 10, the link-layer retransmissions for SOFA generally provide additional improvements in PRR. For example, with 3 NFs the PRR increases from 0.46 to 0.62, an improvement of more than 34 %. One interesting observation is that more than 60% of packet losses are observed in the vicinity of sink (i.e., nodes with h=1 and h=2) due to funneling effect [10] exhibited in sensor networks. We believe implementation of link-layer retransmissions in the vicinity of sink is a cost-effective way to improve overall fidelity.

D. Joining Nodes Analysis

To evaluate the integration of new nodes in the operation testbed, we conduct five sets of identical experiments on both systems. Each experiment starts with 24 active nodes with one source node. We let the network settle for 10 minutes (i.e., for path convergence of baseline system). In the 11th minute, we add a new source node every 2 minutes. In the 20th minute, we add cluster of 7 nodes (with one source node among them) simultaneously and we observed the integration behavior. Each experiment entails six integration instances using 12 sensor devices.

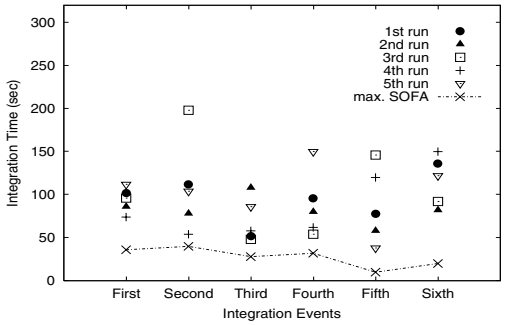

Figure 11. Integration Observation

As observed in Figure 11, the baseline system performs poorly and often requires long integration times. The average integration time for the baseline system is 94 seconds while the longest integration is 198 seconds. In contrast, SOFA completes all of its integrations within 60 seconds. Among the 30 integration opportunities, two rippling events are observed where STF traversed two null-height nodes for height acquisition. The integration experiments also entail 12 height rollbacks where interims heights are resolved during the DNH reacquisition processes. Figure 11 also plots the maximum integration times of SOFA.

E. Overhead Analysis

In this section, we discuss the overhead associated with baseline and SOFA systems. We consider all control and signaling messages to be overhead. Note, that the overhead of the baseline system has a constant rate because it broadcasts routing messages at a fixed interval. Therefore, the overhead of the baseline system is proportional to the network size (i.e.,

number of nodes) and operational duration. With the default settings of the baseline protocol, each node in the 36-node network generate routing message every 20 seconds. Therefore, the overhead of the baseline system has a constant rate of 6480 messages/hour, regardless of network activity. In contrast, the overhead of SOFA varies with the degree of activity in the network and is driven on an on-demand basis. When the network is in an idle state, SOFA does not produce any control overhead. Control overhead is associated only when an active sensor has data packets to forward. Active nodes are either source nodes or DNH nodes. In the 36-mote testbed, there are 6 source nodes with heights of {5, 4, 4, 3, 2, 2}. So, there are at most 20 nodes participating (because some node overlap) in solicitation-based handshakes. Each active node generates an STF message triggering a maximum of 3 ATF responses (see Figure 13). Thus, the control load for the SOFA system is at most 20×4=80 messages. As mentioned in Section IV, SOFA entails initial sink advertisements that flood the network. In the current SOFA implementation, the first sink advertisement broadcasts five consecutive advertisements at a 1-second interval. Therefore, with the worst-case assumption that all non-sink nodes rebroadcast advertisements without error, there would be at most 175 (i.e., 5×35nodes=175) additional control messages in the network. Even with the consideration of optional sink re-advertisements at 5 minute intervals, the final overhead for our experimental testbed is 80+175+(12×35) = 675 msgs/hour. This clearly outperforms the 6480 msgs/hour of the baseline system. The worst case for SOFA is when all nodes in the network are sources. Though we did not conduct experiments for this case, if all 35 non-sink nodes are sources then the number of STF/ATF control messages is 35 x 4 = 140, and the final overhead is 140+175+(12x35) = 735 messages/hour. Even in this worst case, the SOFA overhead is an order of magnitude lower than that of the baseline system.

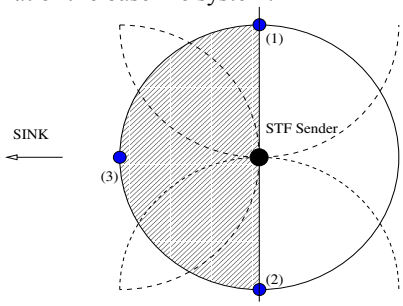

Figure 12. Possible positions of ATF senders when three ATF responses are sent (i.e., maximum bound). Note, that next hop candidates can reside only in the shaded semicircle. Only one ATF can be generated in a common radio range.

Therefore, the SOFA system consumes significantly less control overhead but still offers PRR improvements, faster path convergence, and less service disruption. Reduction in control overhead correspondingly saves energy. In the event of forwarding failures, SOFA's overhead increases due to the re-solicitation process. Therefore, the more dynamic the network, the more control load SOFA generates. This behavior is shown in Figure 14 where the control load of both the baseline and SOFA systems is compared against the degree of network dynamics. The x-axis represents degrees of network dynamics in the form of node failures. The y-axis represents the corresponding control load produced for 60-minute testbed experiments. The control overhead of the baseline system is only dependent on the network size, regardless of the level of data traffic or packet loss. The control overhead for the baseline system shows a constant value of 6480 packets in all cases. In contrast, the overhead for SOFA varies with the number of sensed events and the degree of network dynamics. Under ideal conditions, SOFA generates approximately 700 control packets in the 60-minute testbed run, as described previously, but as observed in Figure 13 the resulting control overhead ranges from 1700 to 3300 packets. This is due to frequent loss of DNHs and lack of passive acknowledgment in the network. The control overhead of SOFA monotonically increases with network dynamics but the curve flatten out beyond 7 NF (i.e., 7 node failures). This is because the network contains less traffic due to lack of connectivity. Packets are simply not forwarded toward the sink and as a consequence less traffic exists in the network such that even with increase in network dynamics, fewer re-solicitations and height management procedures are triggered. We observe that the network often becomes disconnected when we introduce more than 10 node failures. In all cases, SOFA generates less control load than the baseline system.

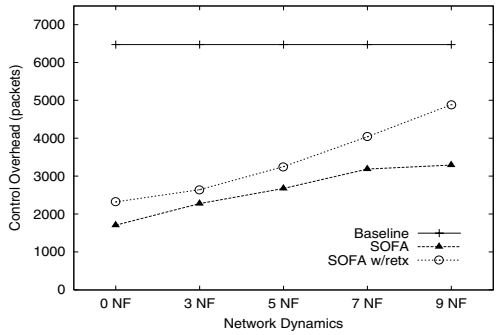

Figure 13. Overhead Comparison

F. *Network Efficiency Analysis*

In this section, we quantify the efficiency of the SOFA system through a parameter called the network efficiency. The network efficiency describes how efficiently a packet is delivered to the sink node with respect to total packet generation and it is defined by (3).

$$\eta_i = \frac{R^i_{SINK}}{(S^i_{SRC} + O^i_{NET})} \quad (3)$$

The parameter *i* of Eq. (3) denotes the type of the network (i.e., baseline or SOFA systems), R_{SINK} represents the total number of packet received by the sink, S_{SRC} denotes the total number of packets originated by source nodes, and O_{NET} represents the volume of control overhead in the network. Therefore, the upper bound for network efficiency is 1, which exists only if there is no packet loss and no control overhead in the network. If a network has substantial overhead and poor packet delivery, the network efficiency would be $\eta_i << 1$. Since a senor network inevitably entails packet loss and some control

overhead, the network efficiency is typically well below the upper bound value of 1.

Figure 14 plots the network efficiency of the baseline, SOFA, and SOFA-retx systems against network dynamics. Each point represents an average of ten experiments with 95% confidence interval. As observed in Figure 14, the efficiency of the baseline system (η_{BASE}) is 0.2 when there is no network dynamics present. As network dynamics increase the η_{BASE} begins to decrease correspondingly. For example, the average values of {R_{BASE} O_{BASE}, S_{BASE}} are recorded {2376, 6480, 5400} when no network dynamics are present but as the network dynamics increases to 5 NF, the R_{BASE} decreases to 1355 and consequently the η_{BASE} decreases to 0.11. The worst η_{BASE} of 0.046 is observed when the network dynamics is at 9 NF.

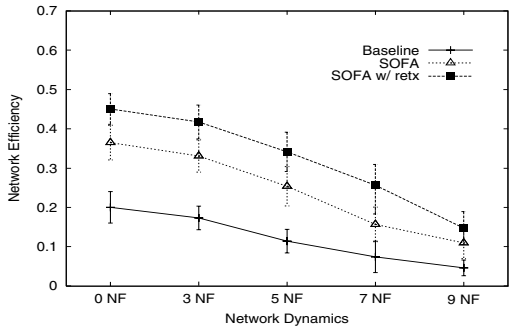

Figure14. Impact of network dynamics on network efficiency

As shown in Figure 14, the SOFA system provides better network efficiency than the baseline system under all tested conditions. With no network dynamics, SOFA has η_{SOFA} of 0.364 while with 9NF η_{SOFA} decreases to 0.109. These results correspond to improvements of 82% and 137 % respectively when compared with the baseline system. The improvement over the baseline system is mainly due to the combination of control overhead reduction and R^{SINK} improvement. When the network faces little network dynamics, the dominant factor for the improvement is overhead reduction while with network dynamics present, the dominant factor is the R^{SINK} improvement. Figure 14 also plots the network efficiency of SOFA with retransmissions ($\eta_{SOFA\text{-}RETX}$). Enabling the link-layer retransmissions option increases the control overhead in all cases but also provides substantial improvement in R_{SINK}. In fact, the R_{SINK} improvement provided by the retransmission outweighs the impact of corresponding increase in control overhead. This condition is clearly shown in Figure 14 because $\eta_{SOFA\text{-}RETX}$ always outperforms η_{SOFA}.

VI. CONCLUSION

The topology of sensor networks continuously change and thus information delivery has to efficiently adapt to these changes while sustaining on-going communications with low overhead. In this paper, we presented the design, implementation, and evaluation of SOFA, an on-demand solicitation-based forwarding protocol for sensor networks. SOFA represents a very simple and scalable solution for routing in experimental sensor networks. Our experimental testbed results confirm that SOFA provides excellent path convergence times and is responsive to various network dynamics experienced in sensor networks. We show through extensive experimentation that on-demand approaches such as SOFA are very applicable to event-driven sensor applications, and that SOFA outperforms the commonly used link estimation-based routing schemes implemented in TinyOS sensor networks.

ACKNOWLEDGMENT

The authors would like to thank Shane Eisenman for his input on this paper. This work is supported by the Army Research Office (ARO) under Award W900NF-04-1-0311.

REFERENCES

[1] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. ACM Mobicom 2000, Boston, MA, Aug. 2000.

[2] M. Zuniga, B. Krishnamachari, Analyzing the Transitional Region in Low Power Wireless Links, IEEE International Conference on Sensor and Ad hoc Communications and Networks, Santa Clara, CA, Oct. 2004

[3] G. Zhou, T. He, S. Krishnamurthy, and J. Stankovic, Impact of Radio Irregularity on Wireless Sensor Networks, MobiSys, MA, June 2004

[4] C. E. Perkins and P. Bhagwat, Highly Dynamic Destination Sequenced Distance Vector Routing for Mobile Computers, ACM Sigcomm, Sept. 1994.

[5] A. Woo and D. Culler, Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks, ACM Sensys, CA, Nov 2003

[6] Eugene Shih et al.., Physical layer driven Protocol and Algorithm Design for Energy-efficient Wireless Sensor Networks. In Proc. of ACM Mobicom, Italy, 2001

[7] ns-2 simulator, http://www.isi.edu/nsnam

[8] TinyOS Package,available from http://www.tinyos.net

[9] Mica2 datasheet, www.xbow.com/products/product_pdf_file/wireless_ pdf/mica2_datasheet.pdf

[10] C. Wan, S. Eisenman, and A. Campbell, CODA: Congestion Detection and Avoidance in Sensor Networks, ACM Sensys, CA, Nov. 2003

[11] MANET Charter, http://www.ietf.org/html.charters/manet-charter.html

[12] V. Park and M. S. Corson, A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks, IEEE Infocom, Japan. 1997

[13] D. S. J. De Couto, D. Aguayo, J. Bicket and R. Morris. High-Throughput Path Metric for Multi-Hop Wireless Routing, ACM MobiCom, Sept. 2003.

[14] B. Karp and H.T. Kung, GPSR: Greedy perimeter stateless routing for wireless networks, in Proc. Mobicom, Boston, August, 2000

[15] M. Heissenbuttel, T. Braun, T. Bernoulli, and M. Walchli, BLR: Beacon-Less Routing Algorithm for Mobile Ad Hoc Networks. Elsevier's Computer Comm. Journal, 27(11):1076-1086, July 2004

[16] Matthias Witt and Volker Turau, BGR: Blind Geographic Routing for Sensor Networks. In Proceedings of the 3rd Workshop on Intelligent Solutions in Embedded Systems (WISES'05). Hamburg, Germany, May 2005

[17] M. Zorzi and R. R. Rao, Geographic Random Forwarding for Ad Hoc and Sensor Networks: Multihop Performance, IEEE Trans. On Mobile Computing, vol. 2, pp. 337-348, Oct-Dec 2003