

=8.5truein =11truein

Scalable Coordination For Sensor Networks In Challenging Environments

Sangmin Kim, VamsiKrishna Vasireddy, and Khaled Harfoush*

Department of Computer Science
North Carolina State University
Raleigh, NC 27695

{skim12, vvasire, harfoush}@cs.ncsu.edu

ABSTRACT

In sensor networks, the unpredictable dynamics caused by the disruption of communication in harsh environments, the depletion of battery resources, and node mobility make coordination between sensor nodes more challenging than coordination between nodes in typical wired and wireless networks. While the literature is rich in coordination protocols for sensor networks and routing protocols for ad-hoc wireless networks, they assume relatively static environments. Using these protocols in highly dynamic environments leads to increased coordination traffic, high energy-consumption, increased event loss, and excessive latency in reporting events.

In this paper we propose SWIFT, a coordination protocol that reacts to changes in nodes' connectivity with a local repair process, significantly reducing the coordination traffic. SWIFT eliminates routing loops through a combination of avoidance and detection mechanisms; and does not require GPS support, which cannot be used in closed environments and consumes significant energy resources. Experimental results reveal that SWIFT outperforms existing protocols and achieves significant energy efficiency, reduces event loss, and the latency in reporting events under various network dynamics.

Categories and Subject Descriptors

C.2.2 [COMPUTER-COMMUNICATION NETWORKS]: Network Protocols—Routing protocols

General Terms

Wireless Sensor Networks

Keywords

Sensor networks, Routing, Mobility

*This work was partially supported by NSF grant CAREER ANIR-0347226.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'07 March 11-15, 2007, Seoul, Korea

Copyright 2007 ACM 1-59593-480-4 /07/0003 ...\$5.00.

1. INTRODUCTION

Advances in hardware and wireless technologies have enabled the development of small and low cost sensor nodes that are capable of sensing, processing and communicating over short distances. The collaborative effort of such sensor nodes, inter-connected in an ad-hoc fashion to achieve a complex sensing task have revolutionized the field of sensing and information gathering in a range of application domains [2, 27]. However, the *coordination* between large colonies of sensors to report sensed information to specialized sensors (*sinks*) is challenging. The scarce energy resources of the sensors and their deployment in tough terrains while communicating over unreliable, short-range shared wireless medium contribute to the challenge. Furthermore, there are many sensing applications that require sensors and/or sink(s) to be deployed over mobile vehicles, which further complicates the coordination task. For example, in a battlefield, soldiers, military equipment like tanks and vehicles, and ammunitions can be equipped with sensors to monitor the equipment conditions and environmental factors like temperature, the presence of chemicals, etc. Mobile sensors form an ad-hoc network that reports to a central command, moving along with the equipment and soldiers. Similarly, sensor nodes can be used to track mobile equipment, doctors and patients in *smart* hospitals; or to monitor mobile parts and equipment during the manufacturing phases of a product.

In the aforementioned applications, sensor nodes need to self-organize and coordinate in order to efficiently and reliably route sensed information to appropriate sink nodes. The underlying coordination/routing protocol should orchestrate the interactions between sensor nodes to minimize messaging overhead, reduce contention over shared medium, and react to hot spots. It also has to adapt to disrupted communication in harsh environments, to the depletion of battery resources, and to node/sink mobility. While many ad-hoc routing protocols have been proposed for traditional ad-hoc networks [17, 7, 3], these are typically not well suited to the unique characteristics and requirements of sensor networks; and while many others have been proposed specifically for sensor networks [9, 10, 22, 23], they mainly assume relatively static environments, and thus incur high event loss and energy consumption in dynamic setups.

In this paper we introduce SWIFT, a coordination protocol for sensor networks in challenging, highly dynamic setups. SWIFT minimizes the coordination/routing overhead resulting from network dynamics by localizing the repairs to the underlying routing topology and eliminates routing loops

through a combination of avoidance and detection mechanisms. SWIFT does not require GPS support, which cannot be used in closed environments and consumes significant energy resources. Also, SWIFT incorporates battery power level of the sensor nodes in the routing path selection decision. This leads to a better distribution of the energy dissipated across the sensor nodes, thereby increasing the overall lifetime of the network. Our experimental results reveal that SWIFT outperforms existing protocols like [21, 9]¹ and achieves significant energy efficiency, reduces event loss, and the latency in reporting events under various network dynamics.

The rest of this paper is organized as follows. In Section 2, we survey routing protocols that may be applicable to sensor networks and to ad-hoc networks in general. In Section 3, we provide the details of the proposed SWIFT protocol and discuss its local repair process, how it avoids/detects routing loops, and how it incorporates energy levels in the routing decisions. In Section 4, we quantify through simulations the performance gains of SWIFT against AODV [21] and DD [9]. We conclude in Section 5.

2. RELATED WORK

A significant body of research has been devoted to routing in ad-hoc networks. Many protocols have been proposed, which include AODV [21, 20], DSR [13, 14], DSDV [19], WRP [18], GSR [4], FSR [11], CGSR [5] and CBRP [12]; and can be classified as *reactive* protocols like AODV, DSR and CBRP, and *proactive* protocols like DSDV, WRP, GSR, FSR and CGSR. Proactive protocols maintain routes between every pair of nodes by periodically exchanging routing information with neighbors. Since sensor networks are densely deployed and the number of sensors is much larger than the number of nodes in a traditional ad-hoc network, proactive protocols cause significant and unnecessary routing overhead, which is a burden on the limited energy budget of the sensors. Reactive protocols create routes on-demand and thereby eliminate the unnecessary routing updates of proactive protocols, and are thus better suited to sensor networks.

Most recently much research has been done on routing in sensor networks, and quite a few protocols have been introduced. Those protocols have been categorized as *data-centric* [9, 24, 29], *cluster-based* [30, 16], and *location-based* [28, 31, 15]. Directed Diffusion (DD) [9] is the pioneer data-centric protocol, and the most popular. In DD, a message has attribute-value pairs, and each node receiving a query message from a sink maintains gradients (next hop information) towards all its neighbors. Detected Events are reported to the sink along these gradients. The sink *enforces* a gradient as a primary path for reporting events. Also, Energy-aware routing [24] was developed to improve the energy consumption by using sub-optimal paths occasionally. Path selection is carried according to a probability function that considers the lifetime of the sensor network. COUGAR [29] is another data-centric sensor protocol, which sees the network as a distributed database system. In COUGAR, nodes are clustered and a selected leader in each cluster aggregates data from its neighbors and forwards the result to the sink, saving energy and extending the lifetime of the network. LEACH [30] is one of the early routing protocols based on clustering. Only cluster heads are involved in routing to save energy. Cluster

¹Our choice of these two routing protocols is justified in Section 4.

heads are elected randomly and take turns to balance energy dissipation among nodes. PEGASIS [16] improves LEACH by forming routing chains within the network rather than building multiple clusters. Only one node is selected to communicate with the sink while the others are sending and receiving messages between their neighbor nodes. PEGASIS reduces energy cost, but excessive delay can be incurred due to congestion at some nodes. Among location based routing protocols, GAF [28] reduces energy cost by turning off unnecessary nodes in the network. GAF forms a virtual grid in the network and each node associates itself with a certain grid location following its GPS reading. Only nodes in the grid alongside the routing path keep its radio on, and others can sleep to save their battery power. GEAR [31] is another location-based routing protocol which saves energy by sending a query message only to the region where the event will be sensed, compared to the DD which floods the network to deliver a query message. Each node in GEAR keeps an estimated cost toward the sink, and the cost is a combination of residual energy and distance to the sink. GEAR improves earlier GPSR [15] in which a packet follows the perimeter of the planar graph to find their route.

Routing incurs significant data loss and energy consumption overhead especially when the network conditions are highly variable due to node and sink mobility. The main reason for the degradation in performance in these proposals is their inefficiency in repairing broken links to the sink. For example, in [26] and [25], Theoleyre et al. propose a cluster-based routing protocol in which a routing tree is constructed and a node propagates *break-messages* to all its tree descendants upon a disconnection of its parent. The propagation triggers the construction of a new branch of the routing tree including all node's descendants, which is expected to be expensive. SWIFT reacts to network disconnections with its *local* repair procedure that minimizes the number of propagated messages, minimizes the messaging contention between sensor nodes, and selects the nodes involved in routing messages based on their residual energy. SWIFT is reactive in order to eliminate the unnecessary routing updates of proactive protocols, and does not rely on GPS.

3. SWIFT PROTOCOL

3.1 Terminology and Overview

Let an *event* be a physical phenomenon with measurable characteristics by sensor nodes and let S be a *sink* node, which gathers information about sensed events from all sensors. Nodes begin their sensing activity *on demand*, when instructed by S through *query* messages, Q , which are flooded into the network. Upon sensing an event, a node forwards the event information in an *event* message, E , to S . In order to support the efficient delivery of E , nodes are organized in a *tree* hierarchy, T , with S being at the root of the tree.² We denote the set of children for a node x in T by $children(x)$ and thus $x \in children(parent(x))$. Each node x maintains the value of its depth $D(x)$ in T where $D(S) = 0$ and $D(x) = D(parent(x)) + 1$. Each node also maintains information about its perceived parent in T . An E message is delivered from x to $parent(x)$, then to $parent(parent(x))$ until it reaches

²To simplify our illustrations, we limit our discuss to the case of only *one* sink node. In general, many sinks may co-exist and the system will maintain one tree per sink.

S.

Node mobility and/or failure may lead to the distortion of \mathcal{T} and nodes may not be able to reach their parents. In this case, a *path repair* process is initiated in which affected nodes try to find alternate parents towards *S*. Our proposal, SWIFT, exploits the depth values a node's neighbors to localize the repair process while limiting its overhead. During the repair process, routing loops can be introduced resulting in messages indefinitely visiting a subset of the nodes and never reaching *S*. The details of the SWIFT protocol and the mechanisms to treat routing loops are explained in the following subsections.

3.2 Query Propagation and Routing Tree Construction

In order to trigger the sensing activity and to construct the event delivery tree, \mathcal{T} , a sink *S* floods a *Q* message into the network. The *Q* message is initially broadcasted by *S* to its neighbors³ and includes in addition to the application-specific information, the depth of *S* in the targeted \mathcal{T} , $D(S) = 0$. A node, *x*, receiving *Q* for the first time from some node, *y*, takes *y* as its parent in \mathcal{T} and adjusts its own depth information relative to the depth of *y* such that $D(x) = D(y) + 1$. Then *x* waits for a random time interval before broadcasting *Q* to its own neighbors, after inserting $D(x)$ in the message. The process is repeated recursively until each node has a depth value and is aware of its parent in \mathcal{T} . Duplicate copies received for *Q* from different nodes are discarded. The reason for waiting a random interval before broadcasting *Q* is to reduce collisions between broadcasted *Q* messages from neighboring nodes.

3.3 Local Path Repair

Dynamics caused by node failures, node mobility and/or sink mobility will disturb the structure of \mathcal{T} : A mobile node may have to pick a new parent, a mobile parent may be out of the communication range of its children which will have to pick new parents, and a mobile sink may implicate many nodes which have to adapt their parents and depth values to tune \mathcal{T} towards the new location of the sink. Since many of these scenarios may happen concurrently, the underlying coordination/routing protocol will have to adapt very efficiently to absorb these dynamics and maintain a usable delivery tree. Inaccuracies will definitely happen in a very dynamic environment but the protocol should recover and the more efficient the recovery, the better the performance. Our proposed SWIFT protocol confronts network dynamics with three types of messages:

- *Orphan (O)*,
- *Adopt (A)*,
- *Fix (F)*

Recall that each node *x* maintains a reference to its parent, $parent(x)$. If *x* cannot communicate with $parent(x)$ due to its failure or its transition outside *x*'s communication range, the following repair procedure is triggered:

³Neighbors of a node *x* are nodes within the communication range of *x*; that is, nodes that can overhear *x*'s transmissions.

3.3.1 O Messages

First, *x* broadcasts an *O* message to all nodes in its radio range informing them that it has become an *orphan* and needs another parent. *O* messages include a field with $D(x)$ value. Neighbors with a depth that is *smaller* than $D(x)$, especially those with large energy levels, are good candidates to be *x*'s parent. On the other hand, nodes that are members of the set $children(x)$ receiving an *O* message from *x*, initiate a timer T_O that is intended to allow *x* some time to find a parent. If T_O expires without hearing back from *x* (through an *F* message, as we describe below, informing them that *x* found a parent), then each node in $children(x)$ realizes that its parent, *x*, failed to find alternative parent. In this case, nodes in $children(x)$ broadcast their own *O* messages to their neighbors. This process is repeated recursively until all nodes are adopted.

3.3.2 A Messages

A node *y* upon receiving an *O* message from an orphan, checks the $D(x)$ field in the message, compares it to its own $D(y)$, and replies with an *A* message to the orphan only if $D(y) \leq D(x)$. However, since many potential parents may reply, contention over the shared medium may result. SWIFT thus requires that node *y* sends *A* after a random time interval T_A proportional to *y*'s depth, $D(y)$, and inversely proportional to *y*'s energy level, $E(y)$.

$$T_A \propto \left(\alpha_1 D(y) + \frac{\alpha_2}{E(y)} \right), \quad (1)$$

where α_1 and α_2 are constants used to give preference between low depth and high residual energy or lack thereof. As a result, nodes with lower depth and with higher residual energy will respond first. Note that if there is still a chance that nodes contend over the shared wireless medium. In this case, the underlying MAC protocol will resolve the contention. An *A* message from node *y* includes a field with a $D(y)$ value. Receiving an *A* response from *y*, node *x* updates its depth value to be $D(y) + 1$ and uses *y* as its parent. An orphan receiving multiple *A* messages from multiple nodes will make use of the first message and suppress the remaining ones.

3.3.3 F Messages

After being adopted by *y*, node *x* broadcasts an *F* message to $children(x)$, informing them of its new depth value and asking them to fix their depth values accordingly. Nodes that are members of $children(x)$ fix their depth values to be $D(x) + 1$ and broadcast *F* messages signaling their new depth values to their own children and the process is repeated recursively. *F* messages are simply *absorbed* by a node, and not re-broadcasted, if this node does not need to change its depth value.

3.4 Adoption Strategies

The main problem that can arise from adoption is the potential for *looping*. A loop occurs when the adopting node turns out to be a descendant of the orphan. Proposition 1 provides the key idea behind the SWIFT approach to avoiding loops.

PROPOSITION 1. *A necessary and sufficient condition to avoid routing loops is that a node *y* extends an adoption invitation to an orphan *x* if and only if *y* is not a descendant of *x*.*

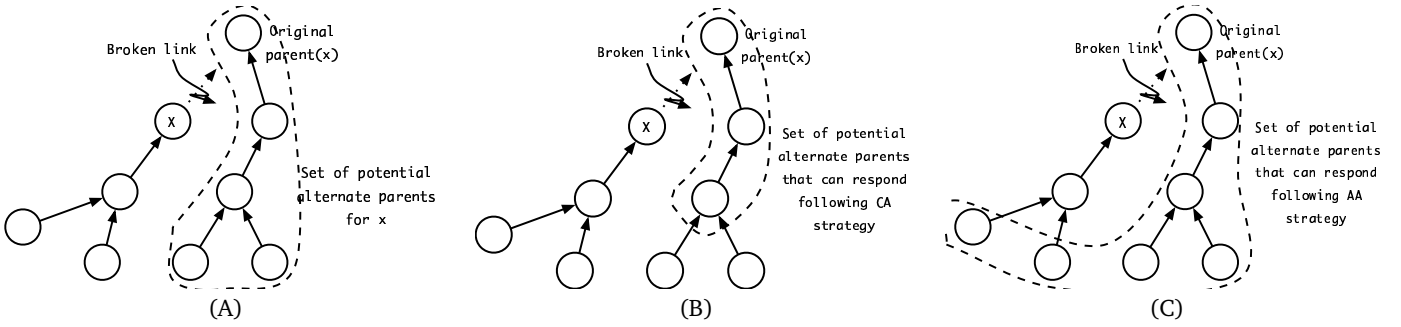


Figure 1: (A) Nodes that can adopt x without leading to loops, (B) Nodes that can extend an adoption offer to x using the conservative adoption strategy, and (C) using an aggressive adoption strategy.

Figure 1(A) shows a scenario where node x lost its parent, and the potential nodes that are allowed to extend adoption invitations to x while avoiding loops. Precautions need to be taken so that descendants of x do not respond to its O message. However, in order to minimize the overhead of the tree management, the SWIFT protocol requires that a node maintains only information about its parent and is not aware of its children in the tree. As a result, while x 's children which are supposed to be aware of their parent x can avoid responding to x 's O message, grandchildren of x are not aware of x and thus may respond. In order to avoid an adoption from a grandchild of x , SWIFT follows the following conservative adoption strategy.

DEFINITION 1. Conservative Adoption (CA). A node y extends an adoption invitation to an orphan x only if $D(y) \leq D(x)$ or if $y \in \text{children}(w)$ where $D(w) = D(x)$ and $w \neq x$.

In other words, following CA, y extends an adoption invitation to x if y is at least as close to the sink as x (has the same depth or smaller than $D(x)$) or if y is a child of some node with the same depth as $D(x)$. Figure 1(B) shows the nodes that are allowed to extend adoption invitation to x under the CA strategy. Note that some nodes that could adopt x without leading to loops will not respond if the CA strategy is followed. One can think of a more aggressive adoption strategy like the one defined below.

DEFINITION 2. Aggressive Adoption (AA). A node y extends an adoption invitation to an orphan x if it is not a descendent of x even if $D(y) > D(x)$.

Figure 1(C) shows the nodes that are allowed to extend adoption invitation to x under the AA strategy. Note that some nodes (grandchildren of x) may try to adopt x and result in routing loops. Thus, as expected, aggressive adoption may lead to more adoption options, which may be useful especially in sparse networks, but comes at the risk of leading to loops. To reduce this risk, SWIFT follows the CA strategy. Still, under high network dynamics, loops may exist. While the CA strategy guarantees a loop-free network when the tree T is *depth-consistent*, this is not guaranteed when it is not. T is deemed depth-consistent if the following definition applies.

DEFINITION 3. A tree T is *depth-consistent* if and only if for any two nodes a and b with depth values $D(a)$ and $D(b)$ such that $D(a) < D(b)$, node a can reach the root of T (the sink) following its upstream nodes (parents) in T using less hops than node b can reach the root.

Since depth-consistency is not guaranteed, SWIFT complements the CA strategy with a loop detection mechanism. Messages are identified by unique identifiers (sequence numbers or random numbers). Messages that follow a routing loop are detected by the first nodes that observes the same message twice; that is, the message with the same identifier in a short period of time. If a routing loop is detected, then it breaks the loop by triggering the path repair process again through an O message.

Now, what happens if an orphan, x , does not receive an adoption invitation from any other node? In this case, x waits for a random period of time and re-broadcasts its O message. Also, the children of node x will start sending their own O messages. Once some of these nodes are adopted, they start extending adoption invitations to their parents which were not adopted. That is, the adoption process will react to temporary network disconnections caused by rough conditions and/or high mobility and will seek any available route to the sink.

4. PERFORMANCE EVALUATION

In this section, we evaluate through simulations the performance of SWIFT under various degrees of network dynamics, and compare its performance to that of a traditional ad-hoc routing protocol, Ad-hoc On-demand Distance Vector Routing (AODV) and to a data-centric routing protocol, which is specifically designed for sensor networks, Directed Diffusion (DD).⁴ Like SWIFT, AODV is a reactive routing protocol designed for mobile setups. We provide the simulation setup in Section 4.1, the performance metrics in Section 4.2, and the simulation results in Section 4.3.

4.1 Simulation Setup

We implemented SWIFT using the *ns-2* network simulator [1]. In all the simulations, we deploy 100 sensor nodes in a square field. Nodes use IEEE 802.11 Medium Access Control protocol for wireless LAN's with RTS-CTS mechanism [6]. In order to provide a realistic emulation of sensor nodes' capabilities, we modified *ns-2* default energy model: We fixed the receive power dissipation of a node to 395 mW, the transmit power dissipation to 660 mW, and the idle time power dissipation to 35 mW [9]. The transmission range of the radio antenna of each node is configured as 40 m. Each node is as-

⁴Our choice of AODV and DD is also motivated in part by the fact that the implementations of most other proposed routing protocols are not publicly available.

Table 1: Values of Simulation Parameters

Parameter	Default Value	Investigated Values	Figures
Total Number of Nodes	100 nodes	100 nodes	-
Pause Time	50 sec	[0,200] sec	Figure 2
Velocity	10 m/sec	[5,25] m/sec	Figure 3
Field Size	225 m \times 225 m	[50 m \times 50 m, 300 m \times 300 m]	Figure 4
Mobility Model	Random Waypoint	Random Waypoint and RPGM	Figure 5
Number of Source Nodes	10 nodes	[4,20] nodes	Figure 6
Event rate per Source	1 event/sec	1 event/sec	-

signed a *velocity*, and a *pause time*. Throughout the simulation time, which we set to 200 sec, a node switches between *static* and *mobile* states. The amount of time that a node spends in the static state is determined by the assigned pause time parameter; and the amount of time in the mobile state is always 1 sec. In the mobile state, a node moves towards a random point in the simulation field at a speed determined by the assigned velocity parameter. By increasing/decreasing the pause time, nodes spend less/more time moving. For example, if the pause time is equal to the simulation time, 200 sec, then the network is stable and nodes do not move for the entire simulation time. When the pause time is set to 0, then nodes are mobile all the simulation time. *Note that in our simulations all nodes may be moving including the sink nodes.*

In our default simulation setup, we use the default values listed in Table 1 (the second column). We also use the default settings for AODV and DD, which are already implemented in ns-2. It should be noted that the AODV and DD parameters can be tuned to tradeoff energy for performance. This however does not change our conclusions, which we detail in Section 4.3. In order to evaluate the performance of SWIFT under different conditions, we varied the value of each parameter separately while keeping all the other parameters at their default values. The investigated values of each parameter are reported in Table 1 (the third column), and references to the figures plotting the corresponding results are given in the fourth column of Table 1.

4.2 Performance Metrics

The metrics used in the performance evaluation are:

- **Event throughput (ET)**: is defined as the percentage of events that were successfully delivered to the sink. The larger the value of ET the better.
- **Average Event Latency (EL)**: is defined as the average end-to-end delay from the time an event is transmitted from a source node to the time it is intercepted by the sink. EL is measured in sec. The smaller the value of EL the better.
- **Average dissipated energy per event received per node (EE)**: is defined as the ratio of the average dissipated energy by each node during an experiment to the number of events that were successfully delivered to the sink node. EE is measured in Joules/node/event. The smaller the value of EE the better.

In each of the reported figures in the next section, each point corresponds to the average of a performance metric from 30 different experiments, using a different random seed for each experiment.

4.3 Experimental Results

In Figure 2, we plot EE, ET, and EL as we vary the pause time of the default setup between 0 and 200 sec. The figure shows that in general, for all values of the pause time, SWIFT incurs less EE, higher ET and less EL compared to both AODV and DD.

AODV is an on-demand routing protocol. As a result, the overhead in establishing new routes is relatively low. However, establishing new routes will incur extra delay. This is reflected in Figures 2 (A) and (C). However, AODV can adapt to mobility and its ET is relatively high. Still, SWIFT with its local repair procedure outperforms AODV in EE, ET, and EL under all values of pause time values.

On the other hand, in DD, the node at which the path to the sink is broken has to wait until the next periodic exploratory event for an alternate path to be established. As a result, DD incurs high routing delay and overhead to find new paths, when existing paths fail due to node mobility (or node failures). This high overhead even leads to high fraction of event losses, which is reflected on the high EE and low ET in Figures 2 (A) and (B). Notice that using DD, ET almost reaches zero when the pause time is zero (very high mobility). In Figure 2 (C), SWIFT outperforms DD in EL for low mobility scenarios but has higher EL for high mobility scenarios. This is an artifact of DD's low ET in the latter case. One can conclude that, SWIFT with its local repair procedure outperforms DD in EE, ET, and EL under all values of pause time values.

In Figure 3, we plot EE, ET, and EL as we vary the velocity of the default setup between 5 and 25 m/sec. The figure shows that for all values of the velocity, SWIFT incurs less EE, higher ET and less EL compared to both AODV and DD. The trends are similar to the those observed in Figure 2 and the explanation is also similar. Note that the differences between Figures 2 and 3 is the result of the fact that decreasing the pause time disrupts more communication paths than increasing the velocity.

In Figure 4, we plot EE, ET, and EL as we vary the node density by varying the field size between 50 m \times 50 m (annotated on the x -axis by 50) and 300 m \times 300 m (annotated on the x -axis by 300). As the field size becomes smaller, nodes are close to each other and it is easier to find alternate paths through many nodes, which is not the case as the field gets larger. Also DD incurs high energy cost since periodic exploratory events are forwarded by each node to all possible gradients (neighbors), which are more in the dense than in the sparse setup. Figure 4 shows that SWIFT still outperforms AODV and DD even in the sparse 300 m \times 300 m setup. The trends in the EE, ET, and EL figures are similar to the previous figures and the reasons for these trends are again as explained for Figure 2.

We next change the random-waypoint model, in which nodes

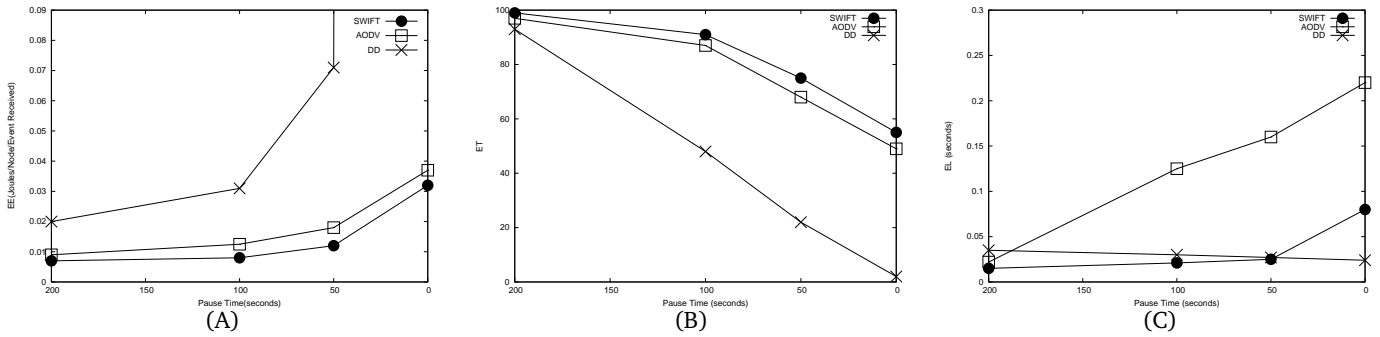


Figure 2: Protocol performance when varying the pause time.

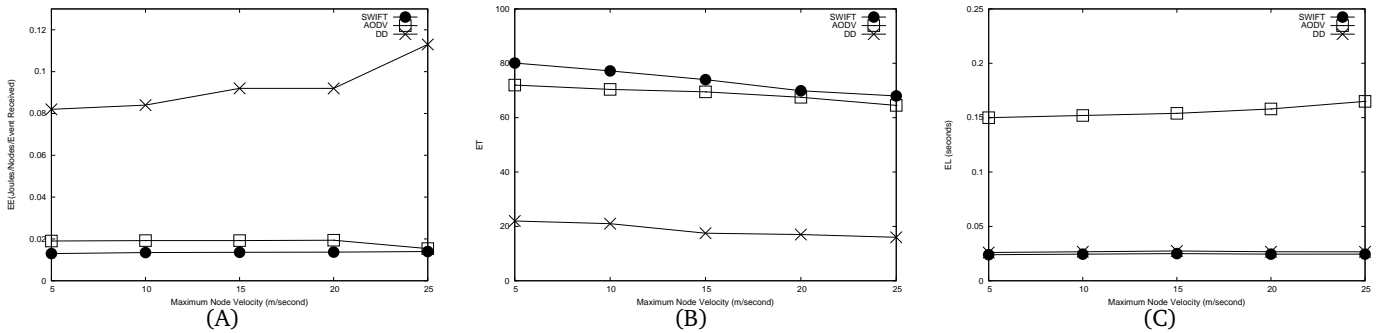


Figure 3: Protocol performance when varying the velocity.

move to random points in the simulation field, to the RPGM model [8] in order to study the performance of SWIFT under group mobility scenarios, in which nodes move in groups. This could be useful in a case where sensors are installed on a group of people moving in close vicinity doing say search and rescue operation. In Figure 5, we plot EE, ET, and EL as we vary the pause time, and when using the RPGM model. It is still clearly the case that SWIFT outperforms AODV and DD. Again, the trends are similar and the explanation is similar.

Finally, we vary the event load on the system by varying the number of source nodes. Still each source node generates 1 event/sec. In Figure 6, we plot EE, ET, and EL as we vary the number of source nodes between 4 and 20. It can be seen that SWIFT still outperforms AOV and DD, and achieves approximately same ET under various event loads. Figure 6 (A) reveals a counterintuitive trend. Typically with the increase in the event load on the network, one would expect EE to either remain constant or increase. But Figure 6 (A) shows that EE slightly decreases. This is justified by our EE definition in Section 4.2: While the average energy dissipated for sending an event from a source node to the sink remains approximately the same independent of the load on the system, the average energy dissipated for the routing overhead per event decreases as it is distributed over a higher number of events and hence the average energy dissipated per event received is slightly lower at higher workloads than at lower workloads. This explains the small downtrend in Figure 6 (A).

5. CONCLUSIONS

A sensor network routing protocol needs to operate well under all levels of network dynamics.

In this paper, we proposed the (SWIFT) routing protocol for sensor networks that can tolerate various degrees of network dynamics due to node failure and node mobility. It provides good performance even if both the sink and the nodes are mobile. SWIFT is based on constructing a tree of sensor nodes on demand, with the sink node as root. Perturbations in the tree structure due to network dynamics are efficiently repaired through localized path repair process, which reduces the routing overhead and latency in finding alternate paths. SWIFT also detects and reacts to routing loops.

ns-2 simulation experiments reveal the performance edge of SWIFT in energy efficiency, event throughput and event reporting latency when compared with AODV and DD in various dynamic setups. Results also indicate that SWIFT can tolerate high network workloads.

6. REFERENCES

- [1] The network simulator - ns-2.
- [2] I. Akyildiz, W. Su, Y. Sanakarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks Journal*, 38(4):393–422, 2002.
- [3] J. Broch, D. Maltz, D. Johnson, Y.-CH, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of ACM/IEEE MobiCom 1998*, pages 85–97, 1998.
- [4] T.-W. Chen and M. Gerla. Global state routing: A new routing scheme for ad-hoc wireless networks. In *Proceedings of IEEE ICC'98*, 1998.
- [5] C.-C. Chiang. Routing in clustered multihop, mobile wireless networks with fading channel. In *Proceedings of . IEEE SICON'97*, pages 197–211, April 1997.

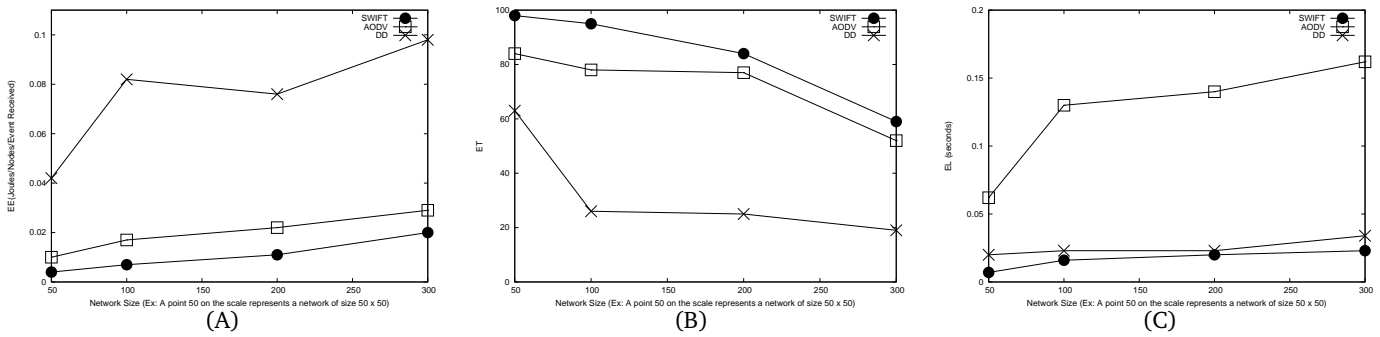


Figure 4: Protocol performance when varying node density by varying the simulation field size.

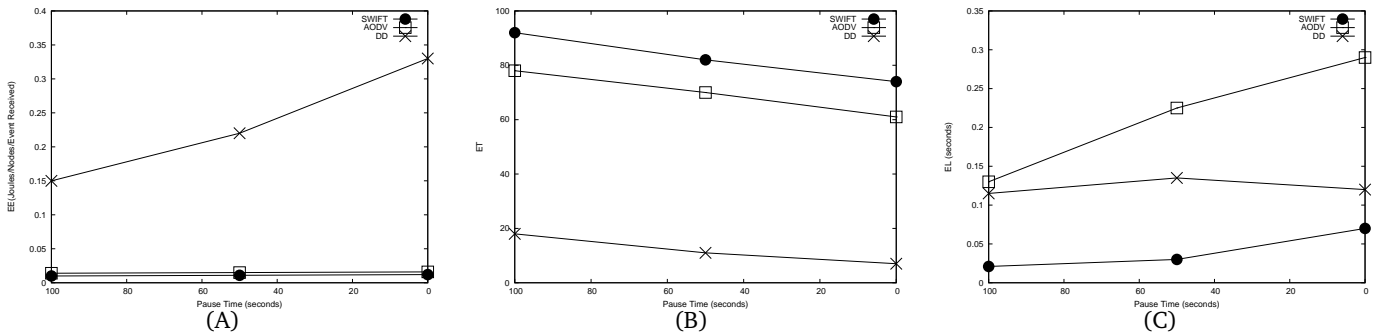


Figure 5: Protocol performance using RPGM group mobility.

- [6] I. C. S. L. M. S. Committee. Wireless lan medium access control (mac) and physical layer (phy) specifications, 1997. IEEE 802.11 draft standard for wireless LAN, MAC and physical layers.
- [7] I. E. T. F. M. W. Group. Mobile ad hoc networks (manet) charter.
- [8] X. HONG, M. GERLA, G. PEI, and C.-C. CHIANG. A group mobility model for ad hoc wireless networks. University of California.
- [9] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of 6th Annu. ACM/IEEE Int. Conf. Mobile Computing and Networking (Mobicom'2000)*, pages 55–67, Boston, MA, August 2000.
- [10] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11:2–16, February 2003.
- [11] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen. Scalable routing strategies for ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks*, pages 1369–1379, August 1999.
- [12] M. Jiang, J. Li, and Y. Tay. Cluster based routing protocol, August 1999. IETF Draft.
- [13] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks, 1996. In *Mobile Computing*, edited by T. Imielinski and H. Korth, chapter 5, Kluwer Academic Publishers.
- [14] D. B. Johnson, D. A. Maltz, and Y.-C. Hu. The dynamic source routing protocol for mobile ad hoc networks, April 2003. IETF Internet draft, draft-ietf-manet-dsr-09.txt.
- [15] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Mobile Computing and Networking*, pages 243–254, 2000.
- [16] S. Lindsey and C. S. Raghavendra. Pegasus: Power-efficient gathering in sensor information systems, March 2002.
- [17] P. Misra. Routing protocols for ad hoc mobile wireless networks.
- [18] S. Murthy and J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Networks and App. J., Special Issue on Routing in Mobile Communication Networks*, pages 183–197, October 1996.
- [19] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance vector routing (dsv) for mobile computers. *ACM SIGCOMM: Computer Communications Review*, 24(4):234–244, October 1994.
- [20] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc on demand distance vector (aodv) routing, February 2003. IETF Internet draft, draft-ietf-manet-aodv-13.txt.
- [21] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, February 1999.

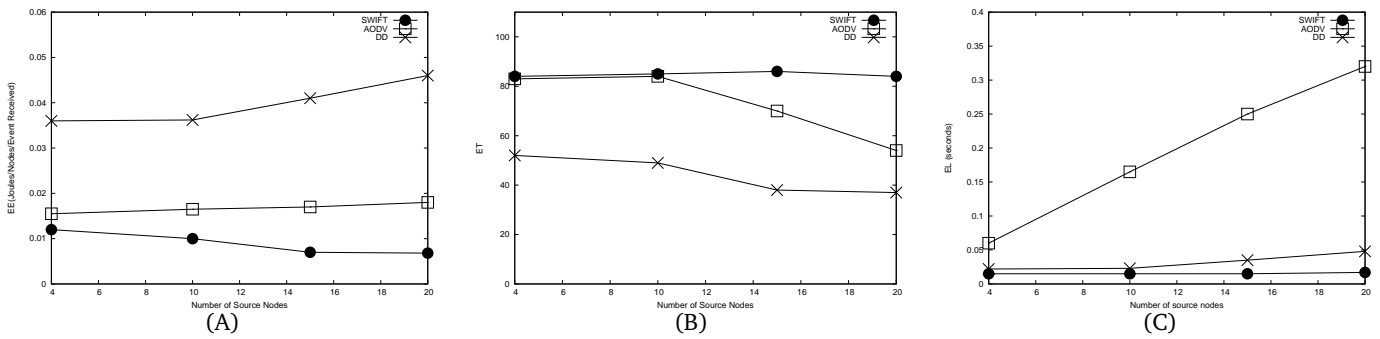


Figure 6: Protocol performance when varying the event load on the system.

- [22] S. Ratnasamy, D. Estrin, R. Govindan, B. Karp, L. Yin, S. Shenker, and F. Yu. Data-centric storage in sensornets. In *Proceedings of ACM First Workshop on Hot Topics in Networks*, 2001.
- [23] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. Ght - a geographic hash-table for data centric storage. In *Proceedings of First ACM International Workshop on Wireless Sensor Networks and their Applications*, 2002.
- [24] R. Shah and J. Rabaey. Energy aware routing for low energy ad hoc sensor networks, 2002.
- [25] F. Theoleyre and F. Valois. A virtual structure for hybrid network. In *IEEE Wireless Communications and Networking Conference (WCNC 2004)*, Atlanta, USA, March 2004. IEEE.
- [26] F. Theoleyre and F. Valois. Virtual structure routing in ad hoc networks. In *International Conference on*

Communication (ICC'2005), Seoul, Korea, May 2005. IEEE.

- [27] Tilak, N. Abu-Ghazaleh, and W. Heinzelman. A taxonomy of wireless microsensor network models. *ACM Mobile Computing and Communications Review*, 2002.
- [28] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. Rome, Italy, July 2001.
- [29] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, 31(3):9–18, 2002.
- [30] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks, 2002.
- [31] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks, 2001.