Journal of Network and Systems Management, vol. 14, no.3, September 2006, pp. 359-380

A Self-selection Technique for Flooding and Routing in Wireless Ad-hoc Networks

Gilbert G. Chen, Joel W. Branch, and Boleslaw K. Szymanski
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY

Abstract

There is a fundamental difference between wireless and wired networks, since the latter employ point-to-point communication while the former use broadcast transmission as the communication primitive. In this paper, we describe an algorithm, called *self-selection*, which takes advantage of broadcast communication to efficiently implement the basic operation of selecting a node possessing some desired properties among all the neighbors of the requestor. Self-selection employs a prioritized transmission back-off delay scheme in which each node's delay of transmitting a signal is dependent on the probability of the node's ability to best perform a pertinent task and in turn, enables the node to autonomously select itself for the task. We demonstrate the benefits of self-selection in two basic wireless ad hoc network communication algorithms: flooding and routing. By relating back-off delay to the signal strength of a received packet, we design an efficient variant of conventional flooding named *Signal Strength Aware Flooding*. By using distance-to-destination to derive back-off delay, we design a novel and fault-tolerant wireless ad hoc network routing protocol named *Self-Selective Routing*.

Keywords – Wireless ad hoc networks, wireless networks, ad hoc routing, ad hoc flooding, leader election

1. INTRODUCTION

Wireless ad hoc networks (WANETs) [1] are used to provide on-demand network infrastructures. They are usually composed of small portable clients (or *nodes*), but they lack consistent topologies. They range in size from small Bluetooth piconets [2] used for device synchronization to large-scale networks used for supporting military and emergency-response applications. Perhaps the most popular WANET applications recently have been wireless sensor networks [3], which are most often used for the pervasive and remote monitoring of environmental or spatial phenomena.

To date, a plethora of research efforts have addressed some significant challenges introduced by various aspects of WANET operation, perhaps the most fundamental being fault-tolerant and energy-efficient communication. Since communication is typically the most costly operation in WANETs because of the overhead of transceiver operation, innovative and efficient approaches to basic communication algorithms such as flooding and routing are in strong demand. Fortunately, the communication primitive in WANETs, *broadcast*, offers some interesting possibilities.

As an instructive example, let us consider a lecture hall with a lecturer and student audience. If the lecturer wants to identify a student with a certain property (e.g., the least number of credits taken at the university) in a lecture hall with n students, it seems that n communications (verbal in this case) would be needed. However, if the lecturer and students use the broadcast nature of verbal communication, they can accomplish this task in just two communications. In the first broadcast, the lecturer asks all students to announce their total number of completed credits after a delay (in seconds, starting from the time that the lecturer stops talking) that is equal to their answer (where 1 credit = 1 second). Upon hearing the first answer, students may suppress their response. Assuming that the answer is unique, in a few seconds, the student with the desired property will identify himself, and the problem will be solved in just two broadcasts. The principles of this "lecture hall" algorithm form the basis of our work.

In WANETs, we consider the problem of identifying a node with pertinent desired properties among all neighbors within the transmission range of the requestor. Such a selection problem is called *local leader election,* to distinguish it from more common global leader election problems typically found in distributed computing [4,5]. This paper starts with a description of a new local leader election technique for WANETs, called *self-selection*. Then, new solutions based on self-selection are presented are applied to flooding and routing, resulting in efficient algorithms that address fault-tolerance and energy-efficiency in WANET communication.

The remainder of this paper is organized as follows. Section 2 presents the basic principles and operation of the self-selection algorithm. Section 3 describes self-selection's application to WANET flooding. Section 4 describes WANET routing based on self-selection and evaluates the fault-tolerance and efficiency of this solution. Section 5 covers related works. Section 6 summarizes the contributions made by this paper.

2. THE SELF-SELECTION ALGORITHM

The majority of leader election algorithms require a synchronization mechanism to enable all participating nodes to determine the start and end of any given election round. Unfortunately, continuous synchronization turns out to be quite a difficult problem for WANETs because of hardware clock skew and other factors [6-9]. However, in many situations, such as in the case of a packet transmission, which exemplifies the occurrence

of a commonly observable event, a common signal can be observed by many nearby nodes. If each node records the time at which the signal is detected, then all nodes that have received the same signal are implicitly synchronized in an on-demand fashion. We refer to these time markings as *implicit synchronization points* (ISPs). Here, we assume that the propagation delay of the signal to each node is negligible. Hence, the nodes are synchronized (within the precision of the signal propagation delay) without the use of any explicit and potentially costly synchronization protocols.

To take advantage of ISPs, every node must be instructed to wait a different amount time before taking further actions. This delay, referred to as the *back-off delay*, has been widely used in CSMA (Carrier Sense Multiple Access) protocols to avoid packet collisions resulting from multiple simultaneous transmissions. The origin of using the back-off delay for this purpose can be traced back to Ethernet [10]. It is also used in the IEEE 802.11 protocol [11].

Regarding our research, the most fundamental observation, with an important consequence, is that back-off delays do not only avoid packet collision, but they also provide a precious opportunity to prioritize the status of different nodes, enabling a simple solution to the local leader election problem. Employing a common ISP followed by different back-off delays assigns different probabilities of becoming a leader to all participating nodes. Each node, after observing the ISP, calculates a back-off delay based on a certain criterion and then sets a back-off timer accordingly. When the back-off timer expires, the node can start transmitting an *announcement* packet. However, if the node receives an announcement packet from another node before its own back-off timer expires, it cancels the back-off timer. Thus, in most cases, only the node with the smallest back-off delay will succeed in transmitting the announcement packet, and will naturally become a local leader. Upon receiving the announcement packet, other nodes passively learn that a new local leader has been elected.

This simple algorithm does not guarantee to always produce at least one local leader. Multiple nodes may choose almost identical back-off delays, leading to a collision. It cannot guarantee *only* one local leader either, since the announcement packet sent by a node may be out of the listening range of some nodes, and these nodes may continue to broadcast new announcement packets. However, both cases can be easily addressed by the algorithm. If there is no local leader elected at the first attempt, some upper layer protocol, such as one in the transport layer, may invoke the procedure repeatedly until there is a local leader. Also, multiple local leaders, as mentioned earlier, may be welcomed for redundancy.

If the reliability of the outcome is desired, then an *arbiter* node can be chosen to broadcast an *acknowledgement* packet when it hears an announcement packet. The arbiter node may or may not be the same as the node that triggered the ISP. However, it must be chosen so that every node involved in the local leader election is within its transmission range. If the arbiter node does not receive any announcement packets within a predefined interval, it will trigger the ISP again by sending out the original synchronization packet. However, if it does receive an announcement packet, it will immediately rebroadcast the acknowledgement packet, upon the receipt of which other nodes will cancel their back-off timers, even if they have not received any announcement packet. Eventually there will be at least one local leader elected.

The heart of the solution is how to derive the back-off delay based on a metric, or a combination thereof, so that the most desirable node would have the greatest probability of being elected a leader. In CSMA protocols the back-off delay is usually randomly generated. Since the back-off delay actually represents the priority assigned to each node, a fully random choice wastes the precious opportunity to prioritize different nodes as they compete for the local leadership. As we will discuss in the next two sections, a wide variety of metrics can be used to derive the back-off delay, some of which may lead to effective solutions to some WANET communication problems.

The use of the back-off delay as a priority value is not completely new. For example, back-off delay has been used to give nodes with more connectivity and more energy higher priority to become the coordinators in the Span protocol [12]. However, the identification of the ISPs, and the use of the arbiter node, as well as the generalization of the local leader election problem described here, to the best of our knowledge, have not been attempted before[1]. We leave a more detailed discussion regarding related works in Section 5.

3. SIGNAL STRENGTH AWARE FLOODING

Routing is a necessity in WANET communication. The simplest routing algorithm is flooding, which is primarily used when there is no existing knowledge about the network's topology. In the most basic form of flooding, every incoming packet is forwarded to every receiver's neighbor, except the one from which the packet was received.

In WANETs, a packet broadcast by one node can be received by many neighboring nodes. As this process repeats, many transmissions will occur, creating an overall energy-intensive operation. A common practice reducing the number of packet transmissions restricts forwarding to only those packets that were not previously received. To remain distinguishable from other packets, every packet must contain a unique sequence number. Every node must also keep a list of sequence numbers of received packets, and whenever a packet is received, its content is checked against this list. The packet will be rebroadcast only when its sequence number is new. This approach is used in *counter-1 flooding* [15].

A packet cannot be rebroadcast immediately for fear of collision, so a back-off delay scheme is usually applied. The node chosen to forward the packet being broadcast can be identified as a local leader, so the solution to the local leader election problem naturally applies here. The end of the current packet transmission is an ISP commonly known to all nodes that received the packet. Using properly selected back-off delays, nodes that are most appropriate to forward the packet can be given higher probabilities of leadership, characterizing the right to rebroadcast the packet. An example of an algorithm that uses this approach is location-based flooding [15], in which the nodes furthest from the previous sender of the packet rebroadcast the packet earliest. However, location information is costly to obtain in WANETs, as it requires the use of additional GPS-related hardware or trilateration systems.

Our solution to the above challenges is to associate the back-off delay with the strength of the received signal, leading to *Signal Strength Aware Flooding (SSAF)*. In general, the farther the receiving node is from the sending node, the weaker the signal is.

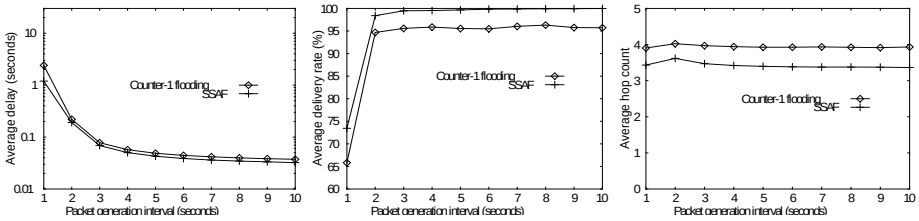
[1] Initial research in self-selection, on which this paper is based, was previously described in [13, 14].

This is true for large-scale wireless propagation models such as the free space and two ray models [16]. In small-scale propagation models such as the Rayleigh model [16] and in practice [17], the signal strength may vary dramatically at the given radius for different directions because of obstacles. However, even in these cases, the weakening of the signal along the specific direction as the distance increases still holds. SSAF does not intend to precisely select the farthest node every time, but to choose nodes that are highly likely to be far away from the sender. Overall, this creates a more efficient flooding algorithm (reducing the number of retransmissions) without relying on the use of additional hardware.

For evaluation purposes, we used the SENSE [18] simulation framework to compare the performance of SSAF with that of counter-1 flooding. A WANET consisting of 100 nodes randomly distributed in a 1000 x 1000m^2 terrain is simulated. 50 connections were created between randomly chosen sources and destinations. In all simulations, the free space propagation model was used. Figure 1 illustrates the comparison between SSAF and counter-1 flooding with respect to the following performance factors:

- Average delay: The average time required for a packet to travel from a source to a destination.
- Average hop count: The average number of nodes a packet must traverse to travel from a source to a destination.
- Average delivery rate: The average percentage of packets sent by sources that are successfully received by destinations.

All results were obtained as a function of the packet generation interval, which ranged from 1 to 10 seconds. To obtain average values, all experiments were repeated four times using different randomly generated seed values.



(a) (b) (c)

Figure 1. Performance comparison between SSF and counter-1 flooding.

As shown by the simulation results, SSAF consistently outperforms the counter-1 flooding with respect to all three performance factors. In most cases, SSAF obtains a slightly shorter end-to-end delay, but for smaller packet generation intervals, the gap becomes much more significant. This is because with more intense traffic activity, the queue between the network layer and the MAC layer tends to become more crowded. In SSAF, a priority queue favors those packets with shorter back-off delays; hence, prioritization takes effect not only among packets in different nodes, but also among packets in the same node. The priority queue has no effect on counter-1 flooding. Another interesting property of SSAF is its decreased average hop counts and increased delivery rates. The average hop count is smaller in SSAF than in the counter-1 flooding because nodes farther from the current hop are more likely to forward the packet. The delivery rate is also improved, since the rebroadcast of a packet tends to cover a larger

area that has not seen the packet before and therefore fewer rebroadcasts are needed, causing fewer collisions. In conclusion, SSAF is likely to establish shorter routes than counter-1 flooding when deployed to find paths between the sources and destinations, which is the primary use of flooding in other, more complicated routing protocols.

4. SELF-SELECTIVE ROUTING

Flooding, in spite of its simplicity, is not a practical solution for large-scale WANETs because of the potential for excessive communication overhead. This is why it is usually employed only as an initial step in establishing bidirectional routes between sources and destinations in WANETs. There is a substantial body of work proposing WANET routing protocols [19], with most being categorized as either *proactive* or *reactive*. Proactive protocols (e.g., DSDV [20]) attempt to continuously maintain routes to all nodes in a network while reactive protocols (e.g., AODV [21] and DSR [22]) only seek to establish routes to specific nodes on an on-demand basis. A growing number of routing protocols have also been developed specifically for wireless sensor networks, in which more emphasis is typically placed on energy-efficiency (e.g., LEACH [23]) and data-centric query models (e.g., SPIN [24] and Directed Diffusion [25]).

Regardless of the application, a salient problem facing WANET routing is the presence of dynamic topologies resulting from presence of transient nodes and inconsistent (and frequently unpredictable) wireless link quality. Even though most WANET routing protocols have mechanisms for maintaining routes in the context of topological changes, they are usually costly and inefficient in nature. Frequently, after a route has been established, a node on a route must constantly monitor the status of the designated next hop on the route. This may be adequate for wired links, however, wireless link failures are difficult to quickly detect, and the links can also temporarily fail as a result of burst interference. Also, it is often the case that a considerable amount of time elapses before a node notices the unavailability of the next hop. When this happens, the node has to either report the route error (to a higher protocol layer) or actively repair it by establishing a new route or selecting an alternative route through other available candidates. The necessity of maintaining explicit routes also implies that nodes on the route cannot autonomously hibernate to conserve energy; it is their responsibility to notify other nodes if they want to do so.

The above problems can be addressed by reducing WANET routing to the local leader problem, as deciding how the next hop will be selected is equivalent to selecting a suitable local leader. To apply local leader election requires just deciding how to compute the back-off delay in transmitting a packet. Naturally, the back-off delay should be assigned in such a way that the closer a node is to the target node (source or destination), the more likely it will be selected as the next hop to forward the packet. Therefore, the distance to the target node, measured in the number of hops, becomes an appropriate metric to calculate the back-off delay. This metric naturally gives rise to a new WANET routing protocol called *Self-Selective Routing (SSR)*, which, in addition to using back-off delay, is distinguished from many others by never maintaining explicit routes. Rather, the actual route is constructed in such a way that the next hop is always determined *after* the packet leaves the current hop.

In the discussion that follows, we assume that wireless links between neighboring nodes are mostly bidirectional. The existence of unidirectional (or asymmetric) links may negatively affect the efficiency, but not the correctness of the protocol. Furthermore, each

node must have a hardware clock with a sufficiently small resolution to differentiate between various back-off delays. These hardware clocks do not need to be very accurate nor do they need to be continually synchronized.

4.1. The SSR protocol

4.1.1. Path discovery process

The data structure used by SSR is fairly simple. Each node maintains a target node cost table. Table entries consist of (i) the identity of a target node (which is either a source or a destination), (ii) the sequence number of the last packet observed from the target node, and (iii) the hop distance from the target to the current node.

The path discovery process consists of two phases described below.

1) Destination request phase: When a source node wants to send DATA packets to a destination node for which there is no cost table entry, it transmits a destination request (DREQ) packet via a flooding protocol and then increases its own sequence number by 1.

Each DREQ packet contains the identity of the source node, a sequence number to distinguish the packet from the other DREQ packets originating from the same source, and the identity of the destination node. In addition, the DREQ packet has an actual hop count field that records the number of hops that this packet traveled from the source to the current receiving node.

If an intermediate node receives a DREQ packet from a source for which there is no entry in its target node cost table, this node creates a new entry with the source ID, sequence number, and actual hop count fields. Otherwise, if an entry for the source already exists, the entry is updated either (i) when the DREQ packet's sequence number is higher than that in the table, or, in case of equality, (ii) when the actual hop count is lower than the stored hop distance. In either case, the hop count is updated with the value in the DREQ packet. If the first case occurs, the sequence number in the table is also updated. The intermediate node then attempts to forward the packet. It will first set a random back-off timer upon the expiration of which the packet will be forwarded. This is to avoid collision with other nodes that received the same packet. If the node receives another packet with the same source ID and sequence number before the back-off timer expires, it simply cancels the timer and does not relay the packet.

2) Destination reply phase: The destination node, upon receiving a new DREQ packet, will reply with a destination reply (DREP) packet. The header of this packet contains the same fields as those of the DREQ packet, as well as an expected hop count field indicating the expected number of hops needed for the packet to travel to reach the target node (in this case, the source). As in the destination request phase, the sequence number of the destination node is increased after broadcasting the DREP packet. Unlike the DREQ packet, the DREP packet does not rely on flooding to find its return path back to the source. Neither does it use an existing path determined during the traversal of the DREQ packet.

The destination node simply broadcasts the DREP packet, without specifying the next hop. It obtains the hop count to the source from its target node cost table, then subtracts 1 from it, and puts the result into the expected hop count field in the DREP packet.

Every node that detects the arrival of a DREP packet will first inspect its expected hop count field. Deciding the next hop then becomes a self-selection problem, and the algorithm presented in Section 2 can be readily applied.

The central idea of the SSR protocol is to derive the back-off delay based on the

known distance, measured by the number of intermediate hops from the target node. This idea is based on the rationale that the node closer to the target should be given higher priority to forward the packet than the node father from the target. However, by passively listening to all packets and looking into the actual hop count field, an intermediate node only knows the distance from the target node to itself, not the opposite. This is why the assumption of bidirectional links is needed, as an asymmetric link in the forward path may result in a longer return path.

Having received a new DREP packet, indicating an ISP, a node determines the back-off delay, $d_{back\text{-}off}$, according to the following equation[2]:

$$d_{backoff} = \begin{cases} \lambda \cdot ((h_{table} - h_{expected}) \cdot U(0,1) + 1) & if\ h_{table} > h_{expected} \\ \dfrac{\lambda}{h_{expected} - h_{table} + 1} \cdot U(0,1) & if\ h_{table} \le h_{expected} \end{cases} \quad (4.1)$$

In Eq. (4.1), h_{table} is the known number of hops to the target node (available from the current node's cost table), $h_{expected}$ is the number of expected hops indicated in the DREP packet, and *U()* is a random number generator producing numbers uniformly distributed over the range defined by its arguments. λ is a tuning parameter that must be carefully chosen. If λ is too small, the difference between $d_{back\text{-}off}$ calculated by various nodes will be too small to avoid collisions. However, a large λ would increase the end-to-end packet delivery delay. As indicated by Eq. (4.1), the formula assigns a back-off delay larger than λ to nodes with a hop count larger than $h_{expected}$. The smaller h_{table} is, the smaller $d_{back\text{-}off}$ will be, and the more likely the node will succeed in transmitting the packet.

After calculating $d_{back\text{-}off}$, the node sets its back-off timer accordingly. A node may cancel its back-off timer and DREP packet transmission if (i) it overhears the same DREP packet being broadcast again, represented by an implicit ACK, indicating that another node self-selected itself to transmit the packet first, or (ii) it receives the explicit ACK packet from the destination node. Regarding the latter case, the destination node acts as an arbiter and continues to listen on the wireless medium after it has transmitted the original DREP packet. If it captures the rebroadcast of the same DREP packet by another node, the destination node immediately transmits an explicit ACK packet that contains the source ID and the sequence number of the DREP packet to notify nodes out of range of the original rebroadcast that the DREP packet has been relayed.

If the back-off timer legitimately expires, the node will immediately transmit the DREP packet. This entire process continues with each self-selected node acting as an arbiter until the source is reached.

In Figure 2, node A wants to send a DREP (or DATA) packet to node D. It simply broadcasts this packet, which is then received by nodes B, C, and F. Assuming that node B ends up with the smallest back-off delay, it broadcasts the same DREP packet. Node C can receive the rebroadcast by B and hence cancels its back-off timer. However, node F is so far away from B that it is entirely unaware that B has become the winner. To prevent node F from reaching the end of its back-off delay and subsequently transmitting the same DREP packet, once node A receives the rebroadcast of node B, it broadcasts an explicit ACK packet containing the same source ID and sequence number as that of the

[2] This formulation is efficient since the expected winner of self selection should have h_{table} - $h_{expected}$ value equal to zero. In more general applications of SSR, logarithm of this difference may be more appropriate, limiting the time the winner will wait for its back-off timer to expire, or even limited number of integer delays can be chosen (e.g., 0 for nodes with the hop count smaller than expected, 1-2 for those with hop count equal to expected and 3 for all the others).

previously transmitted DREP packet. Upon receiving this ACK packet, node F will know that the DREP packet originally broadcast by node A has been forwarded by another node, even though it does not know which node it was.

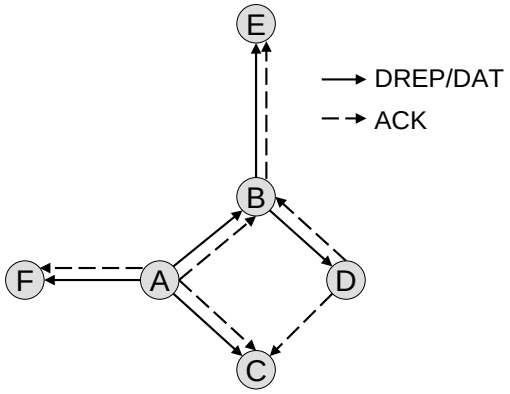


Figure 2. ACK packets suppress multiple copies of the same DREP/DATA packet.[3]

Continuing, when the DREP reaches the source, the source creates a new target node cost table entry using the packet's source ID and actual hop count. It then sends an explicit ACK packet to indicate that the DREP packet has reached the source, since otherwise other nodes would keep trying to retransmit the packet. For instance, in Figure 2, if the DREP packet sent by node B reached the destination node D and node D had neglected to transmit the explicit ACK, both nodes C and E would have continued in self-selection and one of them would have retransmitted the packet. Therefore, the purpose of the ACK packet sent by the target node D is to inform other nodes that the DREP packet reached the destination.

4.1.2. Data transmission process

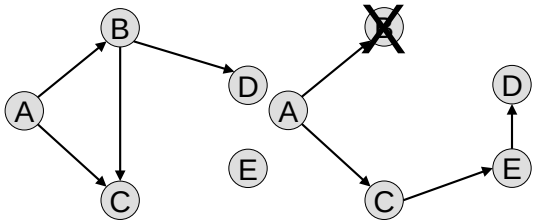
Upon receiving a DREP packet, the source can start transmitting DATA packets towards the destination. DATA packets are transmitted and treated the same way as DREP packets, as both use an actual hop count field and self-selection for forwarding. Therefore, upon the receipt of either a DATA or DREP packet, the receiving node can update in its target node cost table the entry corresponding to the node from which the packet originated.

4.1.3. SSR algorithm properties

Since SSR does not maintain explicit routes, there is no need to constantly monitor the routes' connectivity. Hence, as opposed to most traditional routing protocols, SSR can handle node or link failures without incurring any control packet overhead. Also, with traditional protocols, if a node on the route wants to go to sleep, it must inform its neighbors and pass them the task of relaying packets [26]. It is even more troublesome when a node or a link suddenly goes down, for it is difficult to quickly distinguish a temporary fault from a permanent one. Furthermore, a substantial amount of time may elapse before the nature of the fault is discovered. In contrast, under SSR, when a node or link fails, other nodes will self-select themselves to quickly form a new route; the transition is seamless and no extra actions are needed. As a result, any node, even if it is

[3] For simplicity, arrows to the previous sender are omitted.

on the route, can freely switch to the sleep or standby mode to save energy, making SSR well suited for energy-constrained WANETs.



(a) (b)

Figure 3. Packets immediately seek an alternative route in case of a node failure.

Figure 3 demonstrates that packets can immediately seek an alternative route after a node in the previous route fails. In Figure 3(a), node B is an intermediary on the path from node A to D. However, if node B is deactivated, either passively by a sudden failure or actively by itself in order to conserve energy, the next packet would naturally travel through node C, since node B would no longer be involved in self-selection. This is reflected in Figure 3(b). The transition from a path going through node B to a path going thought node C is seamless, and does not require extra control packets.

In SSR, DATA packets and DREP packets always carry the most up-to-date information about the distance from the originating node. Hence, SSR can often choose the shortest paths to the destination. In other routing protocols, such information could also be made available to intermediate nodes. However, for packets to find the shortest paths, constant route changes would be required in those protocols, and the overhead of excessive route maintenance would likely offset the benefit. It is SSR's ability to handle topology changes effortlessly that makes it capable of always looking for the shortest paths as well.

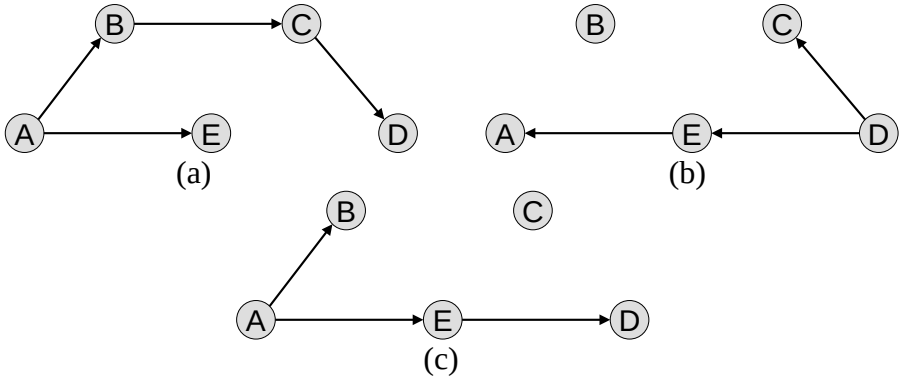


Figure 4. SSR is capable of constantly looking for and switching to shorter paths.

Figure 4 shows such an example. At first (Figure 4(a)), the route between nodes A and D is passing through nodes B and C. Although there is a shorter route via node E, these nodes are not aware of it, either because (i) node E is excluded from the path discovery phase either by randomness of delays, or (ii) by the recent arrival of node E in this neighborhood. If the communication continues to flow one way from A to D, then node E would never get a chance to know that it is within one hop of node D. However, as soon as node D transmits a packet, node E immediately updates D's entry in its target

node cost table. After that, both nodes E and C will compete for the next hop, and E will win because its distance to node A is just one. The next time node A sends a packet to node D, node E will self-select itself for transmission, which effectively shortens the route between nodes A and D by one.

Another less obvious feature of SSR is that it automatically avoids congestion. In dense physical regions, the nodes may subject packets to excessive wait times for transmission in the MAC queue. Even if a node with a large MAC queue is assigned a small back-off delay, most likely it will not be able to self-selected itself as quickly as nodes in less congested areas [13, 14].

4.2. Performance evaluation

In this section, we present an analysis of SSR's performance using the SENSE simulation framework [18]. We compare various performance factors between using SSR and AODV [21]. We have chosen AODV for comparison primarily because it is a straight-forward example of a table-driven routing algorithm that must maintain it neighbors' states and also has no significant tunable parameters that can affect its behavior. Hence, AODV serves as a good, standard benchmark for comparison purposes. In our experiments, we analyze the following performance factors:

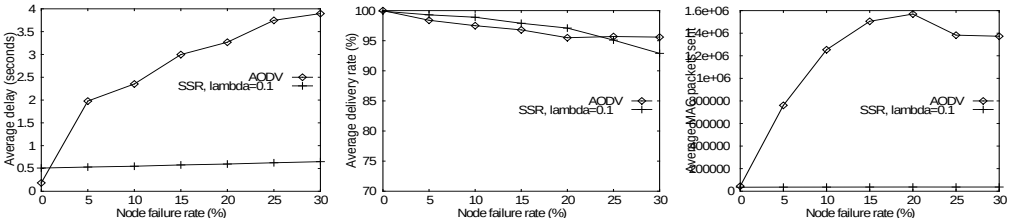
- Average delay: The average time required for a packet to travel from a source to a destination.
- Average delivery rate: The average percentage of packets sent by sources that are successfully received by destinations.
- Average number of transmitted MAC packets: The average number of MAC layer packets transmitted during the entire simulation.

The average number of transmitted MAC packets was included to compare the message-passing overhead of the two protocols and also to give an indication of the energy consumed in the network.

A network consisting of 500 nodes randomly distributed in a 2000 x 2000m^2 terrain is simulated. Nodes had transmission ranges of 250m (under the free space signal propagation model). We executed two sets of simulations. The first set compared the protocols' performance under increasing node failure rates while the number of source-destination pairs was held at 10. The second set maintained a failure rate of zero but varied the amount of network traffic by changing the number of communicating source-destination pairs. Network traffic was generated using a constant-bit-rate model that generated bidirectional traffic at a random rate.

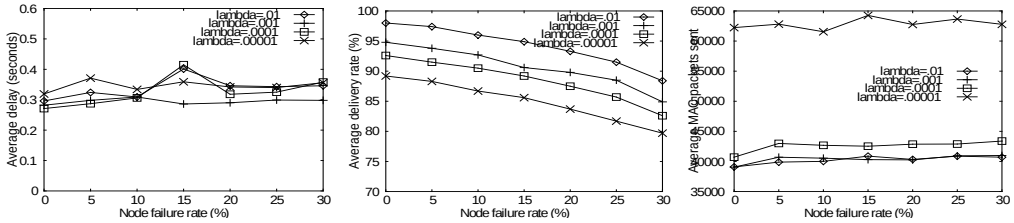
We further evaluated SSR's performance in two ways. One, we ran simulations using varying values of λ (see Eq. 4.1). Two, in addition to using the free space signal propagation model, we also used a signal propagation model, described in [27], that uses log-normal signal propagation and other parameters to simulate the *transitional region* in a node's transmission range [28]. We used this model because empirical data has shown that the packet reception rate does not ideally decrease with increasing distance. Instead, there are three distinct regions in a node's transmission range: *connected*, *transitional*, and *disconnected*. In the first and last regions, the packet reception rates are near 100% and 0%, respectively. However, the transitional region is characterized by highly varying reception rates between 0% and 100% and hence, introduces asymmetric wireless links. The transitional region starts at a transmission distance of approximately 10m and is

approximately 20m long. Hence, for all simulations using the transitional region model, we used a terrain size of 100 x 100m^2 to account for smaller transmission ranges. We did not compare SSR with AODV under this model since AODV is not designed to handle asymmetric links. Finally, all simulations were repeated four times using different randomly generated seed values.



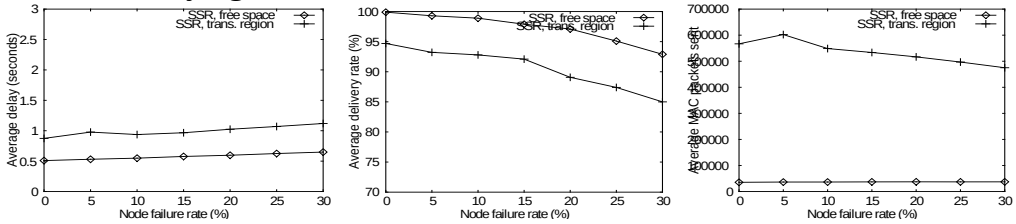
(a) (b) (c)

Figure 5. SSR's packet delivery performance versus node failure rate for the free space model and λ=0.1.



(a) (b) (c)

Figure 6. SSR's packet delivery performance versus node failure rate for the free space model and varying λ.



(a) (b) (c)

Figure 7. SSR's packet delivery performance versus node failure rate for the free space and transitional region models and λ=0.1.

4.2.1. Effect of node failure rate

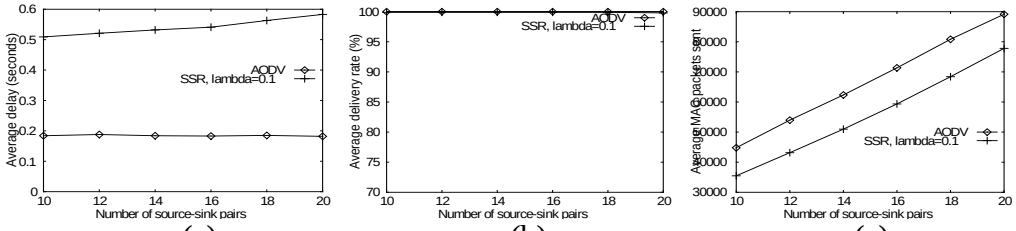
The graphs in Figures 5 and 6 illustrate the effect of node failure rate on both AODV's and SSR's performance under the free space propagation model. The graphs in Figure 7 illustrate the same behavior for SSR observed under the transitional region model. To simulate failure rate, we periodically forced each node to fail for a constant amount of time. For simplicity, the sources and destinations of the CBR traffic never fail.

According to Figure 5(a), SSR maintains a nearly consistent packet delivery delay as opposed to AODV, in which the delivery delay increases along with the node failure rate. Figure 5(b) shows that SSR is highly competitive with AODV in terms of the packet delivery rate. As the failure rate increases, AODV starts to beat the performance of SSR. However, in analyzing the graphs in Figures 5(b) and 5(c) together, it is apparent that for AODV to guarantee nearly the same rate of end-to-end delivery as SSR, it must use the order of magnitude larger number of MAC packets for packet retransmissions and route repairs; this magnitude generally increases along with the node failure rate.

The graphs in Figure 6 illustrate the effect of varying λ, which controls the trade-off between delay and potential collisions. Figure 6(a) shows that packet delivery delay only slightly increases for all λ values as the node failure rate increases. For smaller node failure rates, λ is generally proportional to delay. However, for the smallest λ value (0.00001), the delay is larger. This is expected because as λ decreases, more collisions will occur and result in increased delay. However, as the node failure rate increases, using low and high λ values start to yield similar results. Figure 6(b) shows that λ is proportional to the packet delivery rate, which is expected since shrinking the value of λ yields more collisions. As also expected, the node failure rate is inversely proportional to packet delivery rate. Finally, Figure 6(c) shows that λ is inversely proportional to the number of transmitted MAC packets, which is expected since more collisions cause more packet retransmissions.

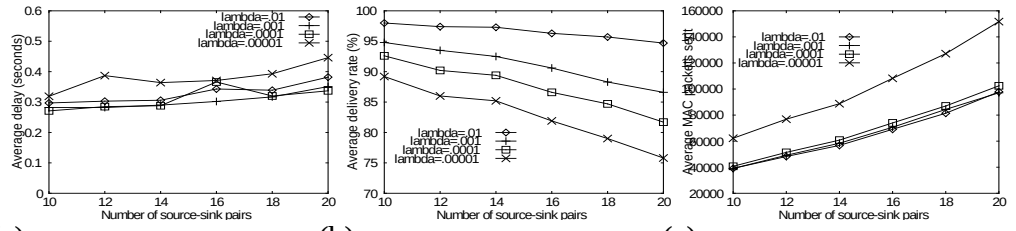
The graphs in Figure 7 show that SSR's performance degrades under the transitional region propagation model. However, the level of degradation is not severe. According to Figure 7(a), delay is increased only by 0.5s on average. This is still an improvement upon AODV's performance for the more ideal free space model which is shown in Figure 5(a). Figure 7(b) shows that at a node failure rate of 0%, the transitional region model degrades SSR's average packet delivery rate only by approximately 5%. As the failure rate increases, the delivery rate decreases at a slightly smaller rate under the transitional region model than the free space model. Figure 7(c) shows that the transitional region model induces SSR to transmit approximately 6 to 7 times more MAC packets. As the failure rate increases, the number of transmitted MAC packets eventually decreases as the MAC layer eventually stops attempting to retransmit packets.

In general, all of these results are expected since there are asymmetric links present that affect the efficiency of the algorithm. Yet, especially regarding end-to-end packet delivery delay and rate, SSR's performance still remains strong considering that the asymmetric links are compounded with re-occurring node failures, in turn creating a very challenging operating environment for any WANET routing algorithm.



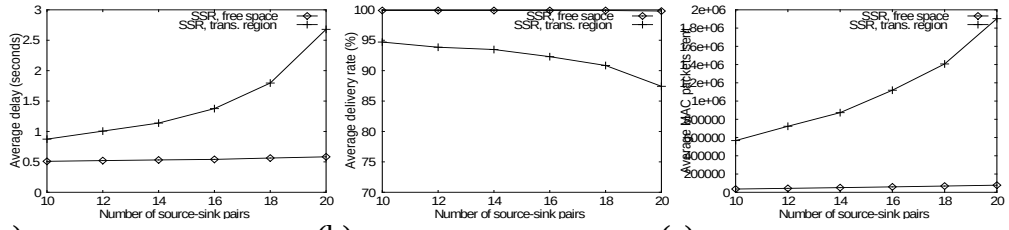
(a) (b) (c)

Figure 8. SSR's packet delivery performance versus network traffic activity for free space model and λ=0.1.



(a) (b) (c)

Figure 9. SSR's packet delivery performance versus network traffic activity for free space signal propagation and varying λ.



(a) (b) (c)

Figure 10. SSR's packet delivery performance versus network traffic activity for the free space and transitional region models and λ=0.1.

4.2.2. Effect of traffic rate

The graphs in Figures 8 and 9 illustrate the effect of network traffic on both AODV's and SSR's performance for the free space propagation model. Similar to the previous section, Figure 10 illustrates the same behavior for SSR observed under the transitional region model.

Figure 8(a) illustrates that SSR exhibits a higher packet delivery delay than AODV. This is expected since one of SSR's fundamental operations involves transmission back-off delay. As network traffic increases, back-off delay will also increase due to increased channel contention. Figure 8(b) shows that SSR remains competitive with AODV regarding packet delivery rate in the context of increasing network traffic. Figure 8(c) shows that for both protocols, the number of transmitted MAC packets increases as network traffic does. Since this set of simulations does not introduce topology changes, collisions can be blamed for the increase in MAC packets. However, SSR still requires fewer packets to be sent primarily because it finds shorter routes and also requires fewer packet retransmissions.

The graphs in Figure 9 generally illustrate the same behavior observed in Figure 6. As shown in Figure 9(a), decreasing λ yields lower delays. However, due to a higher number of collisions, SSR yields lower packet delivery rates and an increased number of MAC packet transmissions.

The graphs in Figure 10 show how SSR's performance degrades under the transitional region propagation model. Whereas SSR's performance was largely unaffected under the free space model, the transitional region model increases degradation for all performance factors as the amount of network traffic increases. These results are to be expected because as the number of packets being transmitted in the network increases in the context of asymmetric links, the opportunities for delays, dropped packets, and retransmissions increase as well.

5. RELATED WORKS

The classic leader election problem has been extensively studied in the context of radio networks [29, 30] in which a communication channel is shared by all nodes and the network is fully connected in the sense that any node can send packets directly to any other node. The time is slotted to avoid collisions, so that each packet transmission always starts at the beginning of a time slot. These networks require a synchronous version of the local leader election problem discussed here. However, since it is difficult, if not impossible at all, to synchronize clocks across all nodes in a WANET, the synchronous clock assumption cannot be satisfied in general. Several methods for electing a global leader in WANETs have been proposed [31, 32]. The term “local leader election” has also been referred to as the problem of finding leaders in partitioned

systems [33], where broken links cause the network to split into a number of disconnected sub-networks. One leader has to be elected for each connected sub-network. However, this problem is in fact a special case of the classic leader election problem, so its solution does not apply to the local leader election problem discussed in this paper.

Research in WANET routing protocols has spawned some notable related contributions. We classify our discussion of them according to two major approaches: *table-driven* and *cost-based routing*.

Table-driven routing includes most of the earlier WANET routing protocols. The primary commonality of these protocols is their use of forwarding tables that direct the node to what neighbor to transmit a packet so that it reaches its destination. Three significant contributions within this class are AODV [21], DSDV [20], and DSR [22]. In AODV, the most widely-accepted table-driven routing protocol, nodes use route request and reply packets to set up paths between each other. This route discovery process results in each node maintaining a forwarding table that explicitly identifies the next node on a path towards the target. AODV uses *hello* and *repair* messages to detect and fix broken links, which add high latency to finding new routes under dynamic network conditions. DSDV works in a very similar manner, but it constantly maintains routes to all nodes and also uses more elaborate message-passing techniques to cope with link additions and failures. In DSR, as opposed to all nodes maintaining forwarding tables, the source node includes the entire route in the packet header; multiple routes are stored at the source in case of link failures. This solution relieves intermediate nodes of table memory requirements but greatly increases the packet overhead and memory requirements at the source node.

Most table-driven routing protocols are close adaptations of traditional routing protocols of wired networks. SSR's significant advantage is its departure from these methods and its ability of taking full advantage of the wireless medium, leading to more efficient and autonomic route repair procedures.

SSR is more representative of cost-based routing protocols, which direct traffic towards its destination using some metric of cost routing cost. TORA [34] establishes directed links towards a destination using node "heights" which are proportional to the hop distance from the destination. Thus, traffic travels in the direction of decreasing height. This is similar to SSR, however, TORA relies on the Internet MANET Encapsulation Protocol (IMEP) [35] for explicit notification about broken links. SSR's avoids this additional overhead using self-selection. GRAd [36] and gradient broadcast routing (GRAB) [37] are more similar to SSR. SSR's use of intermediary explicit ACKs, as opposed to GRAd's use of it only at the end of a route, and SSR's specialized self-selection algorithm enable the reduction of network congestion and packet delivery delay to a larger extent than is possible in GRAd. GRAB employs an aggressive fault-tolerance technique by allowing DATA packets to simultaneously follow multiple paths to a destination. Self-selection achieves the same goal with lower packet delivery overhead. Other protocols that are similar to SSR include Geographic Random Forwarding (GeRaF) [38] and Implicit Geographic Forwarding (IGF) [39]. GeRaf also uses transmission back-off delay. However, it calculates delay as a function of geographic distance from destination, which requires the use of additional GPS hardware; SSR avoids this requirement. GeRaF also uses a more complex packet forwarding scheme that employs a request-to-send/clear-to-send (RTS/CTS) packet transmission mechanism. Last, GeRaF

uses two radios, one to check if a channel is idle before sending data and the other to actually send the data, to reduce the probability of collisions. SSR is comparatively more efficient in that it avoids all of this extra overhead. IGF is similar to GeRaF in that it also uses GPS and an RTS/CTS packet forwarding mechanism. However, in IGF, only the neighbors that reside within a 30 degree angle of a line connecting the sender with the destination are allowed to compete to rebroadcast a packet. If no neighbors within this region respond, then the sender will rebroadcast the packet and neighbors within a larger area will be considered. SSR reduces the probability of a sender having to rebroadcast a packet multiple times by allowing any neighbor with a hop count less than or equal to that of the sender to compete to forward the packet, allowing SSR to route around holes with lower message-passing overhead.

6. CONCLUSION

Our discovery of the local leader election problem and its solution may have a significant impact on protocol design for wireless networks. The local leader election solution can be applied to general protocol design in the following way. First, the protocol to be developed must be carefully analyzed to see if there are any instances of the local leader election problem, as such instances may not be apparent at the first glance. Next, implicit synchronization points must be identified, since they are valuable as they synchronize wireless nodes at no cost. Finally, an appropriately chosen metric for deriving the back-off delays must be found to complete the solution.

SSAF and SSR are two examples of the application of the local leader election solution. Of these two, SSAF has been shown to be capable of improving the efficiency of flooding. SSR, on the other hand, exemplifies a new generation of WANET routing protocols that do not attempt to maintain routes explicitly. The benefit of doing so is that it makes networks more adaptive to dynamic topology changes and therefore more fault-tolerant.

7. REFERENCES

[1] C. Sivaram Murthy and B. S. Manoj, *Ad Hoc Wireless Networks: Architectures and Protocols*, Prentice Hall, New Jersey, 2004.

[2] Bluetooth.com, http://www.bluetooth.com.

[3] F. Zhao and L. J. Guibas, *Wireless Sensor Networks: An Information Processing Approach*, Elsevier, San Francisco, CA, 2004.

[4] V. C. Barbosa, *An Introduction to Distributed Algorithms*, MIT Press, Cambridge, MA, 1996.

[5] N. A. Lynch, *Distributed algorithms, The Morgan Kaufmann Series in Data Management Systems,* Morgan Kaufmann, San Francisco, CA, 1996.

[6] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts, in *2002 Usenix Symposium on Operating Systems Design and Implementation (OSDI'02),* pp.147-163, 2002.

[7] D. Ganesan, S. Ratnasamy, H. Wang, and D. Estrin, Coping with irregular spatio-temporal sampling in sensor networks. *ACM SIGCOMM Computer Communication Review,* Vol. 34, No. 1, pp. 125-130, 2004.

[8] M. L. Sichitiu and C. Veerarittiphan, Simple, accurate time synchronization for wireless sensor networks, in *WCNC 2003 - IEEE Wireless Communications and Networking Conference,* pp.1266-1273, 2003.

[9] J. Van Greunen and J. Rabaey, Lightweight time synchronization for sensor networks, in *Proceedings of the Second ACM International Workshop on Wireless Sensor Networks and Applications, WSNA 2003,* pp.11-19, 2003.

[10] R. M. Metcalfe and D. R. Boggs, Ethernet: distributed packet switching for local computer networks, *Communications of the ACM*, Vol. 19, No. 7, pp. 359-404, 1976.

[11] B. P. Crow, I. Widjaja, L. G. Kim, and P. T. Sakai, IEEE 802.11 Wireless Local Area Networks, *IEEE Communications Magazine*, Vol. 35, No. 9, pp. 116-126, 1997.

[12] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, Vol. 8, No. 5, pp. 481-494, 2002.

[13] G. Chen J. Branch, B. Szymanski, Local leader election, signal strength aware flooding, and routeless routing, in 5*th IEEE International Workshop on Algorithms for Wireless, Mobile, Ad Hoc Networks and Sensor Networks (WMAN05),* April 2005.

[14] G. Chen J. Branch, B. Szymanski, Self-selective routing for wireless ad hoc networks, in *Proceedings of IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob05),* Vol. 3, pp. 57-64, Aug. 2005.

[15] Y. C. Tseng, S. Y. Ni, Y. S. Chen, and J. P. Sheu, The broadcast storm problem in a mobile ad hoc network, in *Proceedings of 5th Annual Joint ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'99),* pp. 153-167, 1999.

[16] T. S. Rappaport, *Wireless Communications: Principles and Practice,* Prentice Hall, New Jersey, 2001.

[17] J. Zhao and R. Govindan, Understanding packet delivery performance in dense wireless sensor networks, in *Proceedings of the First International Conference on Embedded Networked Sensor Systems*. pp.1-13, 2003.

[18] G. Chen, J. Branch, E. Brevdo, L. Zhu, and B. Szymanski, SENSE: A Sensor Network Simulator, in *Advances in Pervasive Computing and Networking*, B. K. Szymanski and B. Yener (eds.), Springer, pp. 249-267, 2004.

[19] M. Abolhasan, T. Wysocki, and E. Dutkiewicz, A review of routing protocols for mobile ad hoc networks, *Ad Hoc Networks*, Vol. 2, No. 1, pp. 1-22, 2004.

[20] C. E. Perkins and P. Bhagwat, Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers, *Computer Communication Review, ACM SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, Vol. 24, No. 4, pp. 234-244, 1994.

[21] C. Perkins, E. Belding-Royer, and S. Das, RFC 3561-ad hoc on-demand distance vector (AODV) routing [Online], 2003, Available: http://www.faqs.org/rfcs/rfc3561.html.

[22] D. Johnson, D. Maltz, and J. Broch, DSR the dynamic source routing protocol for multihop wireless ad hoc networks, in *Ad Hoc Networking*, C. E. Perkins (ed), Addison-Wesley, Boston, MA, pp. 139-172, 2001.

[23] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks, in *Proceedings of HICSS33: Hawaii International Conference on System Sciences*, pp. 8020, 2000.

[24] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks, in *Proceedings of 5th Annual*

Joint ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'99), pp.174-185, 1999.

[25] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks, in *Proceedings of Sixth Annual International Conference on Mobile Computing and Networking (MobiCom 2000),* pp. 56-67, 2000.

[26] J. W. Branch, G. Chen, and B. Szymanski, ESCORT: energy-efficient sensor network communal routing topology using signal quality metrics, in *Proc 4th Int. Conf. on Networking*, Part I, LNCS, Vol. 3420, Springer Verlag, Berlin, pp. 438-448, 2005.

[27] M. Zuniga and B. Krishnamachari, Analyzing the transitional region in low power wireless links, in *Proceedings of the 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pp. 517-526, 2004.

[28] A. Woo, T. Tong, and D. Culler, Taming the underlying challenges of reliable multihop routing in sensor networks, in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pp. 14-27, 2003.

[29] D. E. Willard, Log-logarithmic selection resolution protocols in a multiple access channel, *SIAM Journal on Computing*, Vol 15, No. 2, pp. 468-477, 1986.

[30] K. Nakano and S. Olariu, Uniform leader election protocols for radio networks, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 5, pp. 516-526, 2002.

[31] S. Vasudevan, B. DeCleene, N. Immerman, J. Kurose, and D. Towsley. Leader election algorithms for wireless ad hoc networks, in *Proceedings DARPA Information Survivability Conference and Exposition*, pp. 261-272, 2003.

[32] N. Malpani, J. L. Welch, and N. Vaidya. Leader election algorithms for mobile ad hoc networks, in *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pp. 96-103, 2000.

[33] C. Fetzer and F. Cristian, A highly available local leader election service, *IEEE Transactions on Software Engineering*, Vol. 25, No. 5, pp. 603-618, 1999.

[34] V. D. Park and M. S. Corson, A highly adaptive distributed routing algorithm for mobile wireless networks, in *Proc. 16th Annu. Conf. of the IEEE Computer and Communications Societies*, pp. 1405-1413, 1997.

[35] S. Corson, S. Papademetriou, P. Papadopoulos, V. Park, and A. Qayyum, An internet MANET encapsulation protocol (IMEP) specification, IETF Draft, draft-ietf-manet-imep-spec02.txt, 1999.

[36] R. Poor, Gradient routing in ad hoc networks, unpublished.

[37] F. Ye, G. Zhong, S. Lu, and L. Zhang, GRAdient Broadcast: a robust data delivery protocol for large scale sensor networks, in *ACM Wireless Networks*, Vol. 11, No. 2, March 2005.

[38] M. Zorzi and R. R. Rao, Geographic random forwarding (GeRaF) for ad hoc and sensor networks: energy and latency performance, in *IEEE Transactions on Mobile Computing*, Vol. 2, No. 4, pp. 349-365, Oct.-Dec. 2003.

[39] B. Blum, T. He, S. Son, and J. Stankovic, IGF: A state-free robust communication protocol for wireless sensor networks, Technical Report CS-2003-11, University of Virginia Computer Science Department, 2003.