Proc. IEEE Int. Conf. Wireless and Mobile Computing, Networking and Communications WiMob 2005, Vol. 3, Montreal, Canada, August 2005, pp. 57-64

Self-selective Routing for Wireless Ad hoc Networks

Gilbert G. Chen, Joel W. Branch, *Member, IEEE,* and Boleslaw K. Szymanski, *Fellow, IEEE*

***Abstract*—This paper presents self-selective routing, a self-maintained and fault-tolerant routing protocol for wireless ad hoc networks. The protocol combines broadcast radio transmissions with the novel autonomous programming technique, called *self-selection*, to support routing without the overhead of traditional ad hoc route maintenance. Moreover, the protocol routes packets through a dynamically established and nearly optimal path between two wireless nodes. In the paper, we present simulation results that demonstrate that self-selective routing is scalable and maintains highly sustained performance, even for heavy network traffic and under frequent network node and link faults.**

***Index Terms*—Ad hoc networks, Ad hoc routing, Fault-tolerance, Wireless networks.**

I. Introduction

THE pervasive computing paradigm has introduced a pivotal shift in the design of communication architectures, client devices, and network applications. A perpetually maturing component of this paradigm is the design and use of wireless ad hoc networks (WANETs) [1]. WANETs, usually composed of small portable clients (or nodes), range in size from simple piconets used for device synchronization and distributed portable gaming to large-scale networks used for supporting military and emergency-response scenarios. Recently, perhaps the most popular WANET applications have been wireless sensor networks [2], which are most often used for remotely collecting environmental and contextual data.

To date, a plethora of research efforts has addressed some significant challenges arising in implementation of various aspects of WANET operation, most notably *routing*. In general, WANET routing protocols are expected to satisfy the following essential principles:

- Tolerance of unexpected network faults (e.g., device and link failures).
- Resilience to increasing traffic loads.
- Minimal energy consumption (especially for smaller clients).

Previous efforts have attempted to deliver these features simultaneously, although often at the cost of significant algorithmic overhead leading to diminished scalability and performance. Furthermore, a common thread amongst the most standardized WANET routing protocols is their explicit accommodation of ad hoc architectures, but abstraction of the actual *wireless* medium. Thus, simultaneously satisfying the afore-mentioned principles and assuring high performance of applications remains an open research issue.

This paper presents a new WANET routing protocol termed by us as self-selective routing (SSR). Exploiting the inherent broadcast nature of wireless communication, SSR uses a self-selection algorithm with synchronization achieved implicitly by a packet broadcast to dynamically determine the nearly optimal path between two wireless nodes, even in presence of network faults. Moreover, SSR avoids explicitly maintaining clearly defined routes. The result is a locally autonomic routing protocol aimed at efficiently addressing the essential principles of WANET routing.

The remainder of this paper is organized as follows. Section II describes SSR's background. Section III introduces the self-selection algorithm. Section IV follows with a full description of SSR and its performance is analyzed in Section V. Section VI describes related works and Section VII concludes the paper.

II. SSR Background

Fault-tolerance is not easily attainable in WANETs. Infrastructure-based wireless networks are subject to the same challenge; however, strategically placed wireless access points help palliate the effect of transient links. However, WANETs are afforded no such luxury.

Various works have addressed the salient challenges and pitfalls of wireless ad hoc networking and overall wireless networking in general, such as limited available bandwidth, dynamic topology changes, and adverse signal-propagation effects (see [3], [4], [5] and further references therein). A notable study described in [6] even shows that WANET packet delivery performance may be characterized by more human-oriented factors such as client orientation and movements of vehicles rather than message length, payload pattern, and communication load. Regardless of fault origins, however, the required *simultaneous* combination of fault-tolerance, scalability, and energy-efficiency creates a formidable challenge for deploying ad hoc wireless networks. For instance, traditional ad hoc protocols tend to implement inefficient route repair routines that employ aggressive message-passing resulting in increased end-to-end packet delay. Moreover, energy-conservation techniques, which frequently involve radio deactivation (i.e., [7], [8]), are often designed independently of routing protocols. As such, they are not fully integrated with routing layer activity, thus causing increased communication delay and the number of dropped packets. All of these challenges motivate our research.

Here, we introduce SSR's approach to WANET routing. Most wireless networks use broadcast antennas for communication, as opposed to point-to-point communication typical of wired networks. SSR embraces this characteristic by exclusively using broadcast transmission, without emulating point-to-point communication, for multi-hop routing. Hence, in SSR, a node transmitting a packet does not use forwarding table entries to determine an individual neighbor that will continue packet's passage to the destination. Instead, a packet is broadcast to and received by all neighbors. The receiving node with the lowest forwarding cost to the destination (in our case, hop count) *self-selects* itself to transmit the packet further. This concept is illustrated in Fig. 1, where the gradient of the network's plane represents the decreasing cost of forwarding a packet towards the destination. Note that of all the nodes that received the packet (indicated by the arrows), only the one with the lowest cost forwards it further.[1]

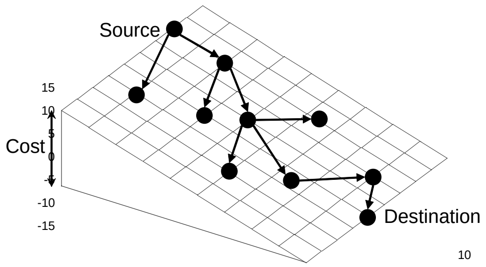


Fig. 1. Simplified example of SSR routing behavior using a cost metric.

SSR's most salient features, including fault-tolerance, optimal route selection, and congestion avoidance, arise from its novel use of a technique called *self-selection*. The crux of this technique relies on three wireless broadcasts and a back-off delay. The first broadcast defines temporal implicit synchronization points for starting back-off delays used for self-selection of the node with the most desirable property. The second broadcast is sent by the node with the shortest back-off delay to inform originating node of the self-selection. The third broadcast announces the self-selection result to all participating nodes. When applied to routing, the result is a locally autonomic routing protocol that maintains near-consistent packet delivery delay and near-optimal route length in presence of increasing frequency of network faults, including those potentially induced by energy-efficient frameworks using radio deactivation.

III. Self-selection

When a group of nodes participates in a self-selection process, some form of synchronization between them is needed to obtain correct results. In spite of many attempts, enforcing *continuous* clock synchronization in WANETs is still an infeasible solution, as demonstrated by experimental results reported in [9]. However, it is feasible to achieve synchronization on an "as needed" basis via some event(s) jointly observed by the synchronized nodes. Tasks may then be coordinated according to such an event, allowing implicit synchronization amongst nodes. Hence, these events represent *implicit synchronization points (ISPs)*.

Amongst many useful operations, ISPs can be used to synchronize timers for back-off delay across multiple nodes. Back-off delay has been widely used in CSMA protocols, such as that used in the IEEE 802.11 wireless standard [10], to avoid collisions resulting from simultaneous packet transmissions. However, we observe that back-off delay not only promotes collision avoidance, but also offers a precious opportunity to prioritize the operations of different nodes, providing an elegant solution to a selection, called here self-selection, of a node with the most application-specific desired properties.

In SSR, the self-selection algorithm determines at each hop which node forwards a packet closest to the destination. Eligible nodes, those receiving a packet from the sender, use the packet reception event as the ISP. We assume that the propagation delay between the sending and receiving nodes negligibly skews inter-node synchronization.

Upon packet reception, the receiving nodes start timers that implement broadcast back-off delay. To ensure that optimal routes[2] are determined, each receiving node calculates its broadcast back-off delay as a function of its distance away from the destination (this delay normally decreases along with each hop). When a node's back-off timer expires, it forwards the packet via a broadcast, which also acts as an implicit acknowledgement (ACK) to the previous sender of this packet as well as the signal for the other nodes synchronized with the same ISP that the passage of the packet has been made. If a node receives such an implicit ACK before its timer expires, it cancels its back-off timer and packet transmission. Thus, in most cases, the node with the smallest number of hops to the destination will self-select itself to forward the packet, simultaneously making other nodes aware of its self-selection.

The solution described above may result in more than one node self-selecting itself for two reasons. First, not all receiving nodes may be in the broadcast range of the first self-selected node to overhear its implicit ACK. To avoid such a possibility, upon receiving the implicit ACK, the original sending node broadcasts an explicit ACK, so that *all* synchronized nodes know that self-selection has already been made. This additional step achieves its goal under the assumption that all links of the original sending node are bi-directional. The reception of the explicit ACK marks the end of the current self-selection round.

The second reason for multiple self-selections is the

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

[2] The route is optimal if there are no faults and is near-optimal otherwise.

presence of nodes with identical hop counts for the given destination, and thus identical back-off delays, ultimately causing collisions. A solution to this problem involves calculating the back-off delay randomly, but controlling the degree of responsiveness to the hop count and randomization. We present this particular solution with greater detail in the next section.

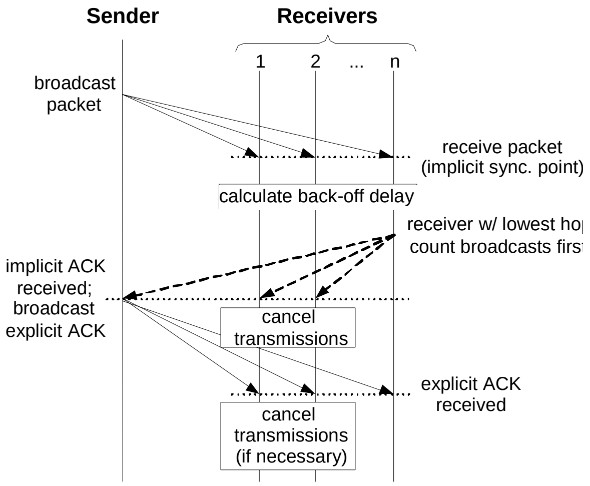

Sender
Receivers
1
2
...
n
broadcast packet
receive packet (implicit sync. point)
calculate back-off delay
receiver w/ lowest ho count broadcasts firs
implicit ACK received; broadcast explicit ACK
cancel transmissions
explicit ACK received
cancel transmissions (if necessary)

Fig. 2. Graphical representation of the self-selection algorithm.

Fig. 2 concisely illustrates the self-selection algorithm, in which node n self-selects itself to forward the packet from the sending node. Fig. 2 also highlights the three important roles assumed by a packet broadcast: (i) it ends the previous self-selection, (ii) it transmits the packet to all potential successor nodes, and (iii) it synchronizes all the potential successor nodes to start a new self-selection cycle.

IV. The SSR Protocol

Before explaining SSR in detail, we state a couple of assumptions made in the paper. As previously mentioned, we assume that all inter-node links are bi-directional. Uni-directional links may negatively affect the efficiency, but not the correctness of the protocol. We also assume nodes to be either stationary or mobile. As with other protocols, we assume that if nodes are mobile, their movements should not be so rapid that any routing protocol would be useless.

A. 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.

(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: Similar to other WANET routing protocols, SSR's path discovery process starts with a 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, as shown in Fig. 3(a) 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.

Source ID	Sequence Number	Destination ID	Actual Hop Count

(a)

Source ID	Sequence Number	Destination ID	Actual Hop Count	Expected Hop Count

(b)

Fig. 3. (a) SSR destination request packet. (b) SSR destination reply packet.

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 (shown in Fig. 3(b)). 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 solution presented in Section III 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 bi-directional links is needed, as a uni-directional link in the forward path may result in a longer return path.

Having received a new DREP packet, indicating an implicit synchronization point, a node determines the back-off delay, $d_{back\text{-}off}$, according to the following equation[3]:

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

In (1), h_{table} is the known number of hops to the target node (available from the current node cost table), $h_{expected}$ is the number of expected hops carried by 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 different nodes will be too small to avoid collisions. However, a large λ would increase the end-to-end packet delivery delay. As indicated by equation (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 in two ways. First, the node will perform cancellation if 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.

The second way of invoking cancellation involves the destination node, which acts as an arbiter and continues to listen on the wireless medium after it has transmitted the original DREP packet. If it captures the re-broadcast of the same DREP packet by another node, it will immediately follow by transmitting an explicit ACK packet that contains the source id and the sequence number of the DREP packet. The explicit ACK packet serves the purpose of notifying nodes out of range of the original re-broadcast that the DREP packet has been relayed. Thus, the reception of this explicit ACK will also cause cancellation.

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.

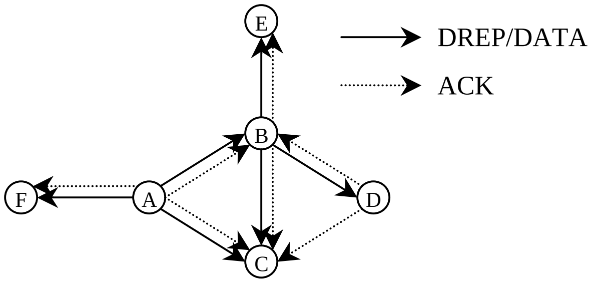


Fig. 4. ACK packets suppress multiple copies of the same DREP/DATA packet.[4]

Fig. 4 illustrates the necessity of ACK packets. In it, 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 among them, it wins the selection and broadcasts the same DREP packet. Node C can receive the re-broadcast 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 re-broadcast of node B, it broadcasts an explicit ACK packet containing the same source id and sequence number as that of the 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 has been.

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 Fig. 4, 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.

[3] 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.

[4] Again, for simplicity, arrows to the previous sender are omitted.

B. Data Transmission Process

Upon receiving a DREP packet, the source can start transmitting DATA packets towards the destination. DATA packets are transmitted and treated in 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 the entry in its target node cost table corresponding to the node from which the packet originated.

C. SSR Algorithm Properties

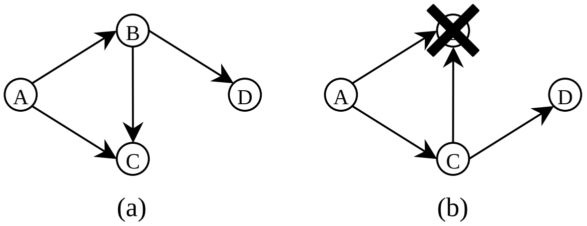


Fig. 5. Packets immediately seek an alternative route when a node in the previous route fails.

A strong feature of SSR is that it does not maintain explicit routes, so 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 [8]. 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 on the route, can freely switch to the sleep or standby mode to save energy, making SSR well suited for energy limited sensor networks.

Fig. 5 demonstrates that packets can immediately seek an alternative route after a node in the previous route fails. In Fig. 5(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 Fig. 5(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.

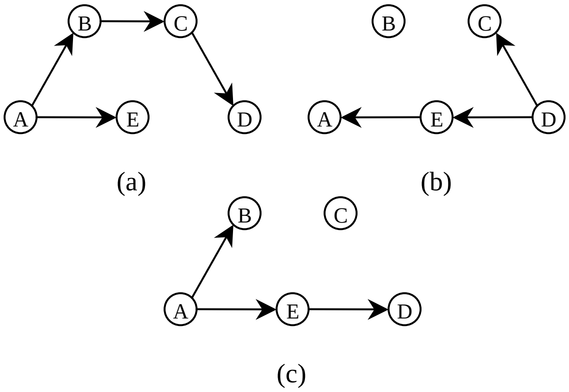


Fig. 6. SSR is capable of constantly looking for and switching to shorter paths.

Fig. 6 shows such an example. At first (Fig. 6(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.

This example indicates that in order for SSR to choose the shortest routes, the communication must be bi-directional, otherwise nodes in shorter routes would not able to update

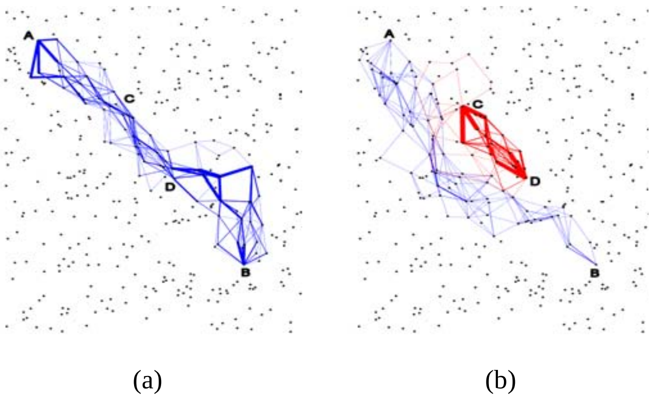


Fig. 7. (a) Normal traffic between A and B. (b) Traffic between A and B avoiding congestion due to traffic between C and D.

their target node cost table. This requirement is not too difficulty to meet in practice, since even if information flow just one way, such practical issues as congestion control (e.g. TCP congestion control protocol) or reliability assurance often result in two way communication.

Another less obvious feature of SSR is that it automatically avoids congestion. In dense physical regions, the nodes may impose packets to excessive wait 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.

Fig. 7 illustrates the actual paths taken by different packets in two simulations. Fig. 7(a) depicts the case with one communicating pair sending packets from node A to node B. Fig. 7(b) shows the same network with additional communicating pair sending traffic from node C to node D. As shown in this figure, SSR forwards packets around the congested area caused by the intensive traffic between C and D. In spite of the increased path length between A and B, the end-to-end delay may drop because the point-to-point delay incurred at each hop on the new path is smaller than that on the original (shorter) path.

V. Performance Evaluation

We implemented SSR using the SENSE simulation framework [11]. Our main purpose was inspecting the performance of SSR in comparison to one of the most standardized WANET routing protocols, AODV [12], which uses forwarding tables to always determine the next node to which to forward a packet.

We executed two sets of tests. The first set compared the protocols' performance under increasing node failure rates. The second set maintained a failure rate of zero but varied the traffic rate by changing the number of communicating node pairs. Performance was measured using the following metrics:

- Average end-to-end DATA packet delivery delay.
- Average end-to-end DATA packet delivery success rate.
- Average number of transmitted MAC packets.

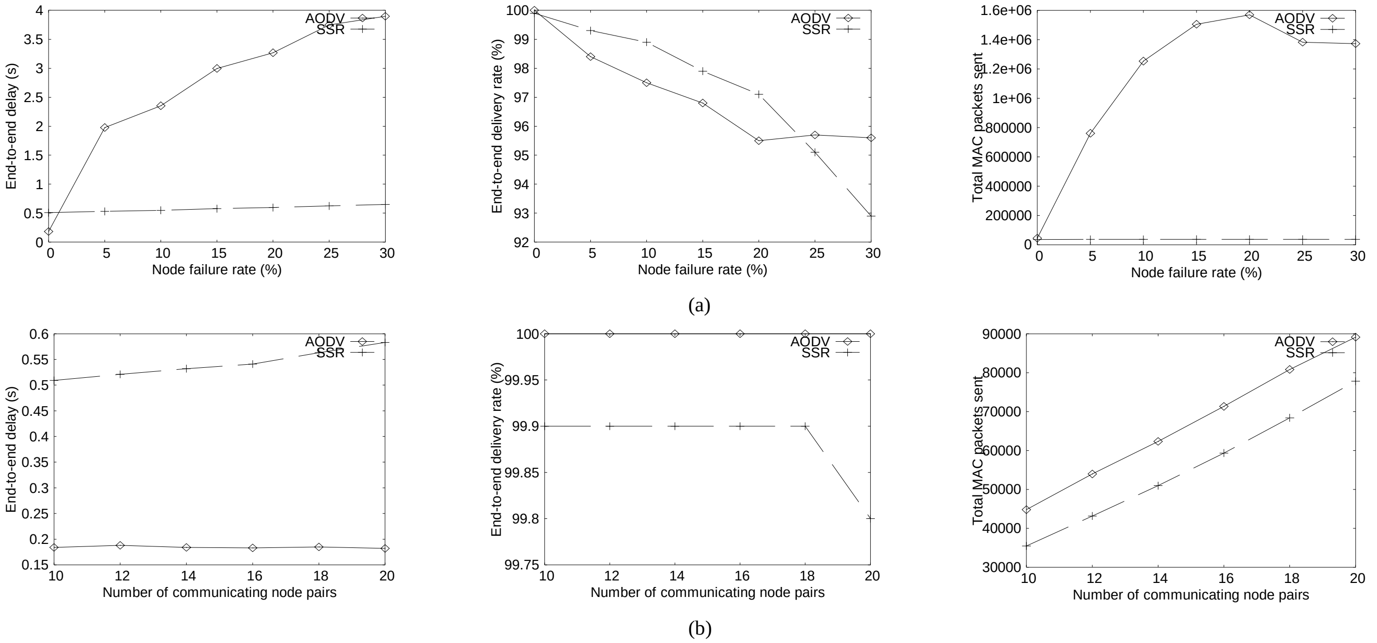
The average number of transmitted MAC packets was included to compare the message-passing overhead of the two protocols and also to give an indication of the energy consumed in the network.

A 2000 x 2000 m^2 terrain was populated with 500 nodes, all of which had a nominal transmission range of 250 m. The free space propagation model was used to simulate the wireless medium. The constant-bit-rate (CBR) model was used to simulate bi-directional traffic between source and destination nodes using a DATA packet size of 1000 bits. Each simulation was executed four times, each using different random number generator seed values.

A. *Effect of node failure rate*

The plots in Fig. 8(a) illustrate the effect of node failure rate on both AODV and SSR. To simulate failure rate, we periodically forced each node to fail for a constant time. The start of each failure was selected for each node randomly at the beginning of simulation, using uniform distribution of the failure start time over the period of the failure repetition. For simplicity, the sources and destinations of the CBR traffic never fail.

The plots show that for AODV to guarantee nearly the same rate of end-to-end delivery as SSR, it must use larger by the order of magnitude number of packets; this magnitude increases along with the node failure rate. While the delivery rate may remain tolerable, AODV's end-to-end delay increases linearly with the node failure rate growth, while



(a)

(b)

Fig. 8. (a) SSR vs. AODV with increasing node failure rate. (b) SSR vs. AODV with increasing traffic rate.

SSR's delay remains nearly constant in these circumstances. These results confirm that SSR is resilient to node failures.

B. Effect of traffic rate

The plots in Fig. 8(b) illustrate the effect of network traffic under both AODV and SSR. Again, AODV and SSR share similar packet delivery rates, with SSR using smaller number of MAC packets, although the difference is smaller than the one seen in simulations with varying node failure rates. However, SSR does incur larger end-to-end delays. This is understandable, as at each node along a path, SSR takes more time to make routing decisions. Surprisingly though, SSR still requires fewer transmissions at the MAC layer. This occurs mainly because of two reasons. One, SSR most likely determines shorter paths. Two, the AODV version used in the simulations relies on a more intensive flooding procedure than SSR does.

VI. Related Works

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

Directed routing includes most of the earlier WANET routing protocols. The primary commonality of these protocols is their knowledge of where to transmit a packet so that it reaches its destination. Three seminal contributions within this class are AODV [12], DSDV [13], and DSR [14]. In AODV, the most widely-accepted directed 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 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. To relieve intermediate nodes of table memory requirements, DSR greatly increases the packet overhead and memory requirements at the source node.

Most directed 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 form of cost gradient. TORA [15] 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) [16] for explicit notification about broken links. SSR's avoids this additional overhead using self-selection.

GRAd [17] and gradient broadcast routing (GRAB) [18] are most 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 exhibits an aggressive technique for fault-tolerance by allowing DATA packets to simultaneously follow multiple paths to a destination. Self-selection achieves the same goal with lower packet delivery overhead.

VII. Conclusion and Future Works

In this paper, we have presented the Self-Selective Routing protocol, SSR, which supports the salient principles of wireless ad hoc networking: tolerance of unexpected network faults, resilience to heavy network traffic and low energy consumption. SSR's use of both broadcast communication and self-selection algorithm provide an efficient approach to sustaining desirable quality of service in fault-prone wireless environments.

We have identified two major directions of future research to further enhance SSR. The first one is node mobility. While SSR is able to accommodate node mobility, it is not yet *optimized* for it. Nomadic environments will decrease the length of time for which target node cost tables entries remain valid. Hence, to retain SSR's locally autonomic behavior, we are researching how to add additional rules for efficiently updating tables based on local observances of node movement.

The second direction focuses on energy efficiency. We are researching how to increase SSR's energy-efficiency by integrating routing with radio state operation. Currently, SSR exhibits energy-efficiency in the sense that explicit ACKs and self-selection suppress unnecessary packet transmissions. However, over-listening also consumes energy and this is especially threatening to networks with resource-starved devices such as wireless sensor nodes. We have shown how SSR is resilient to faults primarily caused by temporary radio deactivation. Thus, in order to alleviate energy consumption due to over-listening, we plan to explicitly synchronize radio operation with the SSR protocol in the style of ESCORT algorithm presented in [4].

References

[1] C. Sivaram Murthy and B. S. Manoj, *Ad Hoc Wireless Networks: Architectures and Protocols*. Upper Saddle River, NJ: Prentice Hall, 2004.

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

[3] R. Schmitz, M. Torrent-Moreno, H. Hartenstein, and W. Effelsberg, "The impact of wireless radio fluctuations on ad hoc network performance," in *Proc. 29th Annu. Int. Conf. on Local Computer Networks*, Nov. 2004, pp. 594-601.

[4] C. –K. Toh, M. Delwar, and D. Allen, "Evaluating the communication performance of an ad hoc wireless network," in *IEEE Trans. on Wireless Communications*, vol. 1, no. 3, pp. 402-414, July 2002.

[5] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Upper Saddle River, NJ: Prentice Hall, 1995.

[6] G. Gaertner and V. Cahill, "Understanding link quality in 802.11 mobile ad hoc networks," in *IEEE Internet Computing*, vol. 8, no. 1, pp. 55-60, Jan.-Feb. 2004.

[7] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," in *Proc. 7th Annu. Int. Conf. on Mobile Computing and Networking*, Rome, Italy, 2001, pp. 85-96.

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

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

[10] *Wireless LAN Medium Access Control and Physical Layer Specifications*, draft amendment, ISO/IEC 8802-11 ANSI/IEEE Standard 802.11, 2003.

[11] G. Chen, J. W. Branch, M. Pflug, L. Zhu, and B. Szymanski, "SENSE: a wireless sensor network simulator," in *Advances in Pervasive Computing and Networking*, B. Szymanski and B. Yener, Ed. New York: Springer, 2005, pp. 249-267.

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

[13] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance vector routing (DSDV) for mobile computers," *Proc. ACM SIGCOMM '94*, London, United Kingdom, 1994, pp. 234-244.

[14] 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. Boston: Addison-Wesley, 2001, pp. 139-172.

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

[16] 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, Aug. 1999.

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

[18] 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*, to be published.