A Tiny and Efficient Wireless Ad-hoc Protocol for Low-cost Sensor Networks

Pawel Gburzynski
University of Alberta
Department of Computing Science
Edmonton, Alberta, Canada T6G 2E8
pawel@cs.ualberta.ca

Bozena Kaminska
Simon Fraser University
School of Engineering Science
Burnaby, BC, Canada V5A 1S6
kaminska@sfu.ca

Wlodek Olesinski
Olsonet Communications Corporation
51 Wycliffe Street
Ottawa, Ontario, Canada K2G 5L9
wlodek@olsonet.com

Abstract

We introduce a simple ad-hoc routing scheme that operates in the true spirit of ad-hoc networking, i.e., in a modeless fashion, without neighborhood discovery or explicit point-to-point forwarding, while offering a high (and tunable) degree of reliability, fault-tolerance and robustness. Being aimed at truly tiny devices (e.g., with 1KB of RAM), our scheme can automatically take advantage of extra memory resources to improve the quality of routes for critical nodes. In contrast to some popular low-cost solutions, like ZigBee,™ our approach involves a single node type and exhibits lower resource requirements. The presented scheme has been verified in an industrial deployment with stringent quality of service requirements.

1. Introduction

The prevailing wisdom regarding the organization of ad-hoc wireless networks assumes point-to-point (or hop-by-hop) communication, whereby each node forwarding the packet on its way to the destination sends it to a specific neighbor. Popular ad-hoc routing protocols can be broadly divided into two groups. Proactive protocols try to maintain up-to-date routing information at every node in anticipation of demand [11, 3, 9], while reactive protocols collect the necessary information only when it is explicitly needed to sustain an actual session [12, 8, 7, 10, 14, 6]. In this context, AODV [12] (which constitutes the basis of ZigBee) is a reactive scheme. Regardless of the paradigm, one can always see at least two separate *modes*: route discovery/maintenance and the actual forwarding of application packets, with the first mode involving special traffic that does not directly originate at the network's application.

1.1. A critique

This "traditional" view on ad-hoc networking appears to us as a flawed relic of wired routing, where a forwarding node would insist on knowing the exact identity of the next-hop neighbor. While perfectly normal in a situation where the node is directly and selectively linked to its neighbors by well-behaved wired channels, this insistence loses is merit in the wireless environment where all the one-hop neighbors of the forwarding node are capable of receiving the packet, regardless of its formal destination. Owing to the rather capricious nature of that inherent broadcast interconnection of neighbors, it has been viewed as a drawback rather than advantage: it has brought about the notorious hidden/exposed terminal problems and triggered lots of effort aimed at their nullification. Consequently, the most popular medium access control schemes employed in forwarding, i.e., derivatives of IEEE 802.11, strongly favor point-to-point communication: the 4-way RTS/CTS/DATA/ACK handshake only works if the packet is intended for one specific neighbor.

In a nutshell, a "traditional" wireless routing protocol works like this. Every node keeps track of other nodes present in its neighborhood. To this end, the nodes periodically broadcast *HELLO* messages, which, in addition to the node identity, may include extra information related to routes (e.g., as in OLSR [4]). Either proactively or reactively, depending on the assumed paradigm, nodes determine best routes between source-destination pairs and store

978-3-9810801-2-4/DATE07 © 2007 EDAA

information about those routes either in a distributed fashion (as in AODV) or at the source (as in DSR [7]). Note that during the neighborhood discovery phase, the nodes necessarily resort to broadcasting. Also, in some protocols (e.g., DSR and AODV), a node is allowed to overhear routing information exchanged by other nodes, thus capitalizing on the broadcast properties of the medium. However, while forwarding an application packet, the sender addresses it explicitly to a specific next-hop neighbor according to the precomputed notion of best route.

The primary difference between the broadcast and unicast transmission is its implied reliability. While a unicast transmission can take advantage of the 4-way handshake with provision for reliable delivery, a broadcast transmission does not even have a well-defined notion of reliability. The role of the latter, which is unavoidable in the face of dynamic and unknown *a priori* neighborhoods, is thus subservient to the former, which is amenable to the reliability-enhancing MAC-level techniques.

Note, however, that the benefits of those techniques tend to be questionable [13], and much more so in sensor networks, where packets tend to be very short. First, the action of announcing the (proper) transmission with the RTS/CTS handshake may take more bandwidth than the actual transmission; thus, the likelihood of damage to an unannounced transmission is in fact lower than the likelihood of damage to the announcement. Second, the neighbor identifier requires room in the packet header and thus incurs extra framing, which significantly inflates the otherwise short packet.

In summary, the traditionally viewed goal of route discovery in ad-hoc wireless networks resembles laying virtual wires where they do not fit very well. As we argue in the rest of this paper, one can try instead to harness the inherent broadcast nature of forwarding, which may compensate for the (apparent) reduction in reliability and bring about new qualities, viz. simplicity and fault tolerance. It is possible to eliminate the different modes of the overall routing scheme and attain scalability, flexibility and uniformity so much needed in low-cost practicable solutions.

1.2. The idea and related work

We would like to carry out efficient forwarding in a truly ad-hoc environment without first probing that environment for the momentary layout of virtual wires. We postulate no neighborhood identification and no extra traffic beyond the application packets. As a forwarding node does not care about the identity of the next hop neighbor that will pick up the packet, there is no need to re-address the packet and encapsulate it for the data-link layer. This also means that a purely internal node, i.e., one that is never a source nor a destination, needs no address!

Note that there exists a trivial scheme with the above properties, save for efficiency: flooding. In pure flooding, a node receiving a packet, always rebroadcast it, unless some trivial criteria (duplicate, time to live) instruct it otherwise. Our idea is to tame flooding and make it behave similar to a scheme in which the nodes have explicitly learned about the good paths to the destinations. This knowledge is acquired while the nodes carry out their forwarding duties.

A node receiving a packet decides to rebroadcast it or drop based on some notion of the node's "relevance" in the joint communal task of bringing the packet to the destination. This notion of relevance is based on information dynamically acquired by the node and stored in a cache and is driven by a set or *rules*. These rules are applied in turn to every received packet for which the node is not the destination (in that case the packet is simply absorbed and not forwarded). The first rule that *succeeds* results in the packet being dropped. If a rule *fails*, the next rule in the chain is evaluated until either one of them succeeds or all the rules have been examined. If all the rules fail, the packet is rebroadcast.

The way the rules are implemented makes them *fail* if no information is available in the cache to make an authoritative decision. This way, a node with limited RAM (and small cache) will behave conservatively, according to the principle *primum non nocere*, possibly forwarding the packets that a better-equipped node would drop. This scalability of behavior to memory size is completely automatic.

Notably, this kind of approach has received little attention in ad-hoc wireless networking. The closest work to our scheme is [15], where a broadcast forwarding protocol is proposed based on the notion of *gradient* understood as the node's perceived cost of forwarding the packet towards the *sink*. That solution is not mode-less, however, because the operation of acquiring the requisite metrics requires special (extra) messages. Moreover, it appears geared towards immobile systems with a single sink (destination), which property simplifies the acquisition and maintenance of the necessary knowledge. What it shares with our scheme is the intentional fuzziness of routes, which is deemed advantageous from the viewpoint of reliability and robustness.

2. TARP: the Tiny Ad-hoc Routing Protocol

Consider the following generic flooding scheme. A node willing to send a packet to some destination simply broadcasts it to the neighboring nodes. A node receiving a packet checks its destination address. If the node happens to be the intended recipient, the packet has reached the end of its path. Otherwise, the node *may* decide to re-broadcast the packet.

This scheme is generic because its essence is in the exact meaning of the word "may." Note that all forwarding protocols for wireless ad-hoc networks, including the point-

to-point ones, implement a variant of this scheme. This is because every transmitted packet is in fact broadcast: in principle, it can be received (and creatively processed) by all nodes within the transmitter's range.

2.1. TARP rules

Even a most naive flooding protocol must take measures to limit the extent of otherwise unlimited flooding. Among the simplest of those measures is restricting the number of hops that a single packet is allowed to travel. In addition to this obvious idea, TARP evaluates a series of rules applied to the received packet, as explained in Sec. 1.2.

For illustration, consider the obvious rule named *DD* (for Duplicate Discard). The rule compares the *signature* of a received packet against a list of signatures of recently forwarded packets (stored in a cache). If the signature is found in the cache, the rule succeeds (and the packet is dropped). Otherwise, the rule fails and the packet may be rebroadcast (if all the subsequent rules fail). Most rules, including *DD*, are naturally parameterized by i) the amount of cache storage assigned to the rule, ii) the replacement policy for the cache, iii) some rule-specific parameters. For example, the *DD* rule assigns a shorter expiration time to the signatures of packets that appear to be closer to their destinations (based on the metrics described in Sec. 2.3). Generally, the parameters of a rule determine its *focus*. A less focused rule may allow for some fuzziness by exploring more (alternative) paths at the same time. While this approach will use more network resources, it will also provide for a higher reliability and better responsiveness to the dynamically changing configuration of the possibly mobile nodes.

2.2. Packet header

In addition to the obvious source/destination address pair <*S, D*>, the TARP-specific components of the packet header include: the session identifier (s) unique for a given <*S, D*> pair, the sequence number of the packet within its session (n), the retransmission count of the packet (k), the maximum length of the path that the packet is allowed to travel expressed as the number of hops (r), the number of hops traveled by the packet so far (h_f), the total number of hops traveled by the last packet on the reverse path (h_b), the slack parameter (m) and the Boolean *optimal path flag* denoted by opf.

Except for opf, the sizes of those fields are generally flexible and may depend on the application, but most of them can be very short. For example, in the present commercial deployment of TARP, both s and k use 4 bits each, n, r, h_f, h_b, fit into 5 bits each (note that the range of the packet sequence number depends on the ARQ scheme used by the transport layer and shouldn't be too large in a wireless environment), and m is stored on 3 bits. Together with the opf flag, the TARP-specific header takes 32 bits, with the packet sequence number being in fact shared with the application (transport layer). The tuple <*S, D, s, n, k*> is called the packet's signature. It uniquely identifies a single packet within a certain time frame.

The role of k is to tell apart the application's (transport-layer) retransmissions of the same packet and, in particular, avoid dropping them as duplicates of the previously seen original. The application may resort to retransmissions in its attempts to overcome the perceived failures of the forwarding mechanism. To honor those attempts, that mechanism should give each of the retransmitted copies the same amount of attention that it extends to every separate packet.

Note that in contrast to the traditional approach to implementing a hop number limit, whereby the remaining hop count of a packet is decremented towards zero, TARP uses two fields: the bound set by the source remains constant, while a separate field stores the increasing number of hops traveled by the packet. This is because both values are needed by some rules.

2.3. The SPD rule

The most powerful rule of TARP is called SPD for Suboptimal Path Discard. Its objective is to avoid forwarding a packet via a route that takes it too far from the shortest path between source and destination. The rule uses its own cache (the SPD cache) storing triplets $<N, h_{NK}, C_{NK}>$ indexed by nodes N interpreted as destinations, where K is the node caching the triplet (see Fig. 1). Whenever K receives a packet sent by node N, it extracts h_f from the packet's header and stores it (as h_{NK}) in the (updated) cache entry for N. As this operation does not concern duplicates discarded by the *DD* rule, h_f will tend to reflect the current best (shortest) path from N to K.

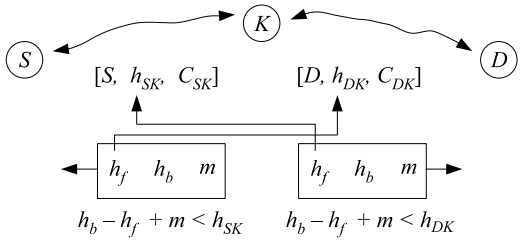

Figure 1. The SPD rule.

Suppose that K in Fig. 1 receives a packet traveling from S to D. When S dispatches such a packet, it inserts into its header h_b, the total number of hops made by the last packet received from D. Again, by the virtue of rule *DD*, this tends to be the minimum number of hops in which S

can be currently reached from D. Thus, K evaluates $h_b - h_f$, which is the expected number of remaining hops to be covered by the packet before it reaches D, assuming that it will follow the same (best) path as a previous packet from S that reached D a while ago. If $h_b - h_f < h_{DK}$, K can suspect that there is a better path from S to D than any path passing through K. Thus K may consider itself irrelevant for the forwarding task in this case and drop the packet.

This approach has two minor flaws. First, it appears to assume that paths in the S to D direction look the same as the ones from D to S, i.e., the symmetry of radio links, which need not always hold in real life. Second, if the rule always strictly follows the inequality $h_b - h_f < h_{DK}$, i.e., it succeeds whenever the inequality holds, sessions may not be able to recover from node failures or mobility. Therefore, there are two ways to relax the rule. By adding $m > 0$ to the left-hand side (see Fig. 1) the rule will allow paths that appear worse than the best one by a certain margin. This way, the population of nodes involved in sustaining the session between S and D will be larger than strictly needed to follow the shortest paths in the network. Second, each time the rule succeeds for a given destination N (i.e, the packet is dropped), it increments C_{NK}. When the counter reaches a certain threshold (which is a parameter of the protocol), the rule forcibly fails. This way, every once in a while, any node perceiving the session at all is given a chance to contribute to the community's effort of finding the best route.

2.4. The MAC layer

No handshake of the kind available in IEEE 802.11 is possible in TARP because 1) the recipient's identity, besides being unknown, is unimportant, 2) there can be multiple legitimate recipients that do not know about each other, 3) even if some neighbors do not receive the packet, the ones that have picked it up may still be able to push it forward on the way to the destination. Note that even a two-way handshake, DATA/ACK, is not possible, as a single broadcast might legitimately trigger multiple (colliding) acknowledgments.

A forwarding node in TARP would like to know whether the packet has been picked up by one or more nodes in the neighborhood, which will *bona-fide* try to forward it towards the destination. The identity of those nodes is of no direct importance to the forwarding node. The approach used in our first implementation of TARP was to listen for a copy of the transmitted packet (forwarded by one of the neighbors) and interpret it as an indication of success—in addition to timers and counters used to diagnose failures. There are two problems with this solution. First, depending on the load at the forwarding node, there can be a significant delay between packet reception and retransmission. Second, to make this idea work, the destination itself has to "forward" (i.e., retransmit) all received packets, which creates unnecessary noise in its neighborhood. A better solution employs the idea of *fuzzy acknowledgments*.

When a node receives a packet, it first evaluates the rules and then, if all of them fail (i.e., the packet will be forwarded), the node responds with a short burst of RF activity (a simple unstructured packet) of a definite duration. This activity, if present, will tend to occur after a very short period of silence (analogous to SIFS in IEEE 802.11) needed by the node to evaluate the rules. When multiple recipients send their acknowledgments at (almost) the same time, the sender will not be able to recognize (all of) them as valid packets. However, the sender can interpret any activity (of a certain bounded duration) that follows the end of its last transmitted packet as an indication that the packet has been successfully forwarded. Although the value of this indication is inferior to that of a "true" acknowledgment, it does provide the kind of feedback needed by the data-link layer to assume that its responsibility for handling the packet has been fulfilled.

Any "normal" packet transmission in TARP is preceded by a short LBT (listen before transmit period) whose duration guarantees that fuzzy acknowledgments (if perceivable at all) are not interfered with by packets. Note that the implementation of fuzzy acknowledgments violates the layering principles. This is because the acknowledgment can only be sent after the rules have been evaluated, i.e., the node concludes that its reception of the packet is going to contribute to its delivery. This is not the only place in TARP where layering gets in the way. Some rules operate best if their evaluation is postponed until the packet is about to be retransmitted, i.e., past the queuing in the data-link layer. Consequently, our implementation of TARP's "protocol stack" has no layers at all.

2.5. Avoiding multiple paths with the same cost

One redundancy problem that *SPD* is unable to solve is caused by possible multiple paths with the same smallest number oh hops. Consider the situation depicted in Fig. 2. Even with the most restrictive setting of the slack parameter, $m = 0$, both paths $<K_1, K_2, K_3>$ and $<L_1, L_2, L_3>$ will be occupied by the packets traveling between S and D. The duplicates will be eliminated at A (for the D–S direction) and B (for the direction from S to D); however, each of the K and L nodes will be consistently forwarding them because, according to *SPD*, each of those nodes is located on the shortest path between S and D. The problem is particularly nasty if the two rows of nodes can hear each other because then the redundant traffic contributes to the noise in their neighborhood and feeds into congestion.

Recall the *opf* flag in the packet header (Sec. 2.2). This flag is set by a forwarding node when it knows that the

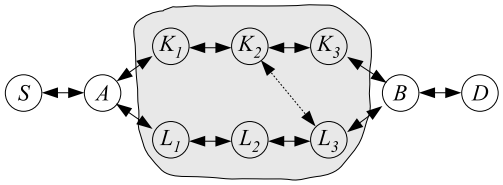

Figure 2. Multiple paths with the same minimum cost.

packet is being forwarded on one of the best paths, i.e., the SPD rule fails non-forcibly. This means that the packet should normally reach the destination, unless some nodes have moved away or failed. Consider nodes K_1 and L_1 in Fig. 2 receiving a packet from node A. Owing to the collision avoidance mechanism involving LBT and randomized retransmission delays (akin to IEEE 802.11), one of these nodes, say K_1 will be first to re-broadcast the packet. The other node, L_1 will yield to this transmission and overhear (receive) the packet re-broadcast by K_1. Normally, that packet would be diagnosed as a duplicate and promptly discarded by *DD*. However, if *opf* is set in the packet header, *DD* yields to another rule, which compares the signature of the received packet against the signatures of all packets currently queued for transmission. If a matching packet is found at L_1 and its h_f is not less than $h_f - 1$ in the received duplicate, then the packet at L_1 is dropped. In plain words, this means that by forwarding its copy of the packet, L_1 would not improve upon the forwarding opportunities already extended by K_1.

This mechanism will not help if the paths are disjoint, but it will kick in wherever they cross. Note that while long disjoint paths of the same length need not be rare in a realistic network, the ones for which the length is the shortest possible definitely are.

3. Concluding remarks

3.1. Performance

A meaningful performance comparison of wireless ad-hoc protocols is not a grateful task, especially when those protocols are so drastically different as, say DSDV and TARP. Even if we agree on common standards of the channel and mobility models, the multitude of idiosyncratic parameters of the routing algorithms (e.g., *HELLO* message intervals, cache expiration times, various damping factors for route update triggers, to mention just a few attributes of the most popular variant of DSDV) must be combined with the (not always quantifiable) parameters of the completely incompatible MAC schemes. Thus we cannot really compare the routing protocols as such, but only complete solutions, and, to be of value, the study must be comprehensive and account for the application context.

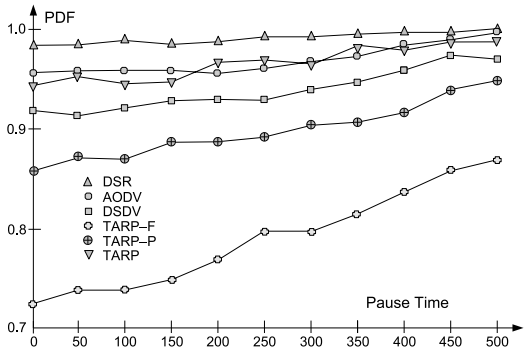

Figure 3. Performance comparison of TARP with other protocols under mobility.

Fig. 3 illustrates the performance of TARP in terms of the packet delivery fraction (PDF) and compares it with the performance of three popular ad-hoc routing schemes. The limited reliability of the network structure is captured by the mobility of nodes. The network consists of 50 stations moving within a $670m \times 670m$ square. Each node remains stationary for *Pause Time* seconds, then it selects a random destination within the square and moves there at a uniformly distributed speed between 0 and $10m/s$. Upon reaching the destination, the node pauses again, selects another destination, and so on, until the end of simulation. A single experiment continues for 500 seconds; thus, the pause time of 500 implies "no mobility." The radio model assumes the bit rate of 2 Mbps and the nominal range of $150m$. IEEE 802.11 is used in the MAC layer of all the point-to-point routing protocols. The broadcast packets needed for route discovery (in AODV, DSR, DSDV) are sent using physical carrier sensing and are not acknowledged. The traffic sources are CBR (continuous bit-rate), with the source-destination pairs spread randomly over all nodes. The amount of cache storage at every node allows it to store ca. 40 entries in the SPD cache and about 80 entries in the DD cache. The packet size is 128 bytes.

TARP-F is the basic protocol with the most simplistic MAC layer, which roughly corresponds to IEEE 802.11 with no handshake. As the MAC layer is assumed to be hermetic, no elimination of parallel paths (Sec. 2.5) is implemented. TARP-P is a version with fuzzy acknowledgments but still without the elimination of parallel paths. Finally, TARP denotes the full version of the protocol.

3.2. Applications

The target class of applications of our TARP-based sensor networks are monitoring and event-processing systems deployed in environments where human life is at risk. One such effort is our involvement in the Advanced Mobile Emergency Communications Prototype Project with a specialized vehicle capable of rapidly deploying emergency communications throughout regions of Brithish Columbia accessible by road. The vehicle is equipped with a range of facilities including terrestrial radio and satellite communications, telephone, video, Internet gateway and other systems to turn it into a field relay. Among the available communication facilities are cohorts of self-organizing ad-hoc wireless sensor nodes enabling rapid deployments of distributed (and not necessarily stationary) area monitors. The standart assortment of sensing capabilities of a node includes temperature, pressure, humidity and acceleration. It is possible to describe complex and distributed conditions to be used as alert triggers and/or data collection patterns.

Our most serious industrial installation of a TARP network (the details are classified) consists of several hundred nodes forming a single mesh network covering the area of ca. $4\mathrm{km}^2$. The network demands non-trivial and reliable mesh operation (meaning that the multi-hop functionality is essential). Data collected at sensors are fed to a small number of sinks interfaced to infrastructure computers, which manifest their presence to the nodes via periodic beacons.

3.3. Work in progress

Our present efforts are focused on the development of a high-fidelity simulation (or rather emulation) environment for mesh networks running custom protocols (with complete modeling flexibility at all layers). As the programming platform of our devices, i.e., PicOS, descends from a network simulator [2, 5], this task turns out to be much easier than it would seem at first sight. The operation of transporting a PicOS application to a SMURPH model is almost mechanical, which allows us to think of a compiler to carry out this task with no or little guidance from the experimenter.

Our experiments with popular simulators, like ns-2 [1], have been rather disappointing, mostly because they make it difficult to adequately model phenomena occurring at the MAC layer, especially if that layer is nonstandard and has no ready built-in model. For example, the (randomized) packet's fate at the receiver is typically determined at the moment of its transmission. This makes it impossible to implement the timing of subtle events related to the partial and interrupted reception of the packet, which is not without impact on the exact behavior of the recipient and possibly other nodes in the neighborhood influenced by the recipient's subsequent activity.

References

[1] The Network Simulator: NS-2: notes and documentation. http://www.isi.edu/nsnam/ns/.

[2] E. Akhmetshina, P. Gburzynski, and F. Vizeacoumar. Picos: A tiny operating system for extremely small embedded platforms. In *Proceedings of ESA'03*, pages 116–122, Las Vegas, jun 2003.

[3] T.-W. Chen and M. Gerla. Global state routing: a new routing scheme for ad-hoc wireless networks. In *Proceedings of ICC'98*, June 1998.

[4] T. Clausen and P. Jacquet. Optimized Link State Routing protocol (OLSR). RFC 3626, IETF Network Working Group, October 2003.

[5] W. Dobosiewicz and P. Gburzynski. Protocol design in SMURPH. In J. Walrand and K. Bagchi, editors, *State of the art in Performance Modeling and Simulation*, pages 255–274. Gordon and Breach, 1997.

[6] M. Gunes, U. Sorges, and I. Bouazizi. ARA—the ant-colony based routing algorithm for manets. In *Proceedings of International Workshop on Ad-hoc Networking (IWAHN)*, Vancouver, British Columbia, Canada, August 2002.

[7] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[8] J. Li, J. Jannotti, D. D. Couto, D. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM' 00)*, pages 120–130, 2000.

[9] S. Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Networks and Applications Journal*, pages 183–197, October 1996.

[10] V. Park and M. Cors, on. A performance comparison of TORA and ideal link state routing. In *Proceedings of IEEE Symposium on Computers and Communications '98*, June 1998.

[11] C. Perkins and P. Bhagwat. Highly dynamic Destination-Sequenced Distance Vector routing (DSDV) for mobile computers. In *Proceedings of SIGCOMM'94*, pages 234–244, August 1993.

[12] C. Perkins, E. B. Royer, and S. Das. Ad-hoc On-demand Distance Vector Routing (AODV), February 2003. Internet Draft: draft-ietf-manet-aodv-13.txt.

[13] A. Rahman and P. Gburzynski. Hidden problems with the hidden node problem. In *Proceedings of 23rd Biennial Symposium on Communications*, pages 270–273, Kingston, Ontario, Canada, May 29-June 1 2006.

[14] C.-K. Toh. A novel distributed routing protocol to support ad-hoc mobile computing. In *Proceedings of IEEE 15th Annual International Phoenix Conf. on Comp. and Comm.*, pages 480–486, March 1996.

[15] F. Ye, G. Zhong, S. Lu, and L. Zhang. Gradient broadcast: a robust data delivery protocol for large scale sensor networks. *Wireless Networks*, 11:285–298, 2005.