Computational Management Science, 8(3):237-258, July 2011
DOI: 10.1007/s10287-009-0102-y

Cognitive and Self-Selective Routing for Sensor Networks*

Erol Gelenbe, Peixiang Liu
Imperial College, UK
e.gelenbe@imperial.ac.uk, p.liu@imperial.ac.uk

Boleslaw K. Szymanski, Christopher Morrell
Rensselaer Polytechnic Institute, USA
szymansk@cs.rpi.edu, morrec@cs.rpi.edu

Abstract

New approaches to Quality-of-Service (QoS) Routing in wireless sensor networks which use different forms of learning are the subject of this paper. The Cognitive Packet Network (CPN) algorithm uses smart packets for path discovery, together with reinforcement learning and neural networks, while Self-Selective Routing (SSR) is based on the "Ant Colony" paradigm which emulates the pheromone-based technique which ants use to mark paths and communicate information about paths between different insects of the same colony [12]. In this paper we present first experimental results on a network test-bed to evaluate CPN's ability to discover paths having the shortest delay, or shortest length. Then, we present small test-bed experiments and large-scale network simulations to evaluate the effectiveness of the SSR algorithm. Finally, the two approaches are compared with respect to their ability to adapt as network conditions change over time.

1 Introduction

The requirement for timely delivery of digitized audio-visual information has generated much interest in QoS routing [6]. However the shortest path algorithm still plays a very important role in communication networks [13]. In the Internet, Distance Vector [14] and Link State [1, 15] algorithms are usually used to find the shortest path between source and destination. Many factors, including non-negligible delay and relatively infrequent link state updates due to overhead concerns, have an impact on the precision of global network state information required to make "good" routing decisions.

Other QoS routing algorithms based on an adaptive and incremental approach have also been suggested, and this paper compares two of them which use different forms of learning: the Cognitive Packet Network (CPN) algorithm which uses smart packets, and reinforcement learning with neural networks, to find the best paths, and the Self-Selective Routing (SSR) paradigm inspired by ants that mark traversed paths with pheromone to communicate information about those paths to insects of the same colony.

The CPN routing algorithm adaptively selects paths so as to offer best effort QoS to the end users, based on different user-defined QoS criteria [10, 8], including power saving [9]. CPN uses *Random Neural Networks with Reinforcement Learning* (RNNRL) [7, 11] to make routing decisions in a distributed fashion at each node. Each output link of a node is represented by a neuron in the RNN at that node. The arrival of *Smart Packets (SPs)* triggers the execution of RNN and the output link corresponding to the most excited neuron is chosen as the routing decision.

The SSR protocols use broadcast communication, a de facto standard for nodes of wireless sensor networks, with a prioritized transmission back-off delay scheme to enable each receiving node to decide autonomously whether to forward a packet [5]. The prioritized back-off delay for nodes that currently forward a packet is reduced to nearly zero to practically guarantee that they will also forward future packets of the same flow if the same nodes do not fail, thereby assuring that if a stable path exists, sooner or later, the flow will follow it. When a severed route

*Research was sponsored by US Army Research Laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The collaboration was also supported by the NSF Grant OISE-0334667. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defence, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

is encountered, the protocol dynamically and locally re-routes packets so that they traverse the shortest surviving route.

This paper is organized as follows. CPN routing is described in Section 2, in which performance results concerning CPN are presented in Section 2.1, while CPN's ability to adapt to traffic load is quantified in Section 2.2. SSR is introduced in Section 3 and its operation and route repair algorithm are detailed in Section 3.1. The experimental and simulation results for SSR are presented in Section 3.2. Comparisons between the two discussed protocols and conclusions are presented in Section 4.

2 CPN Routing

CPN includes three types of packets which play different roles. *Smart or Cognitive Packets (SP)* are used to discover routes for connections. They are routed using a reinforcement learning-based (RL) algorithm on a QoS "goal". We use the term "goal" to indicate that there is no guaranteed QoS and that CPN provides a best effort to satisfy the QoS demands. SPs find routes and collect measurements, they do not carry payload. The RL algorithm uses the observed outcome of a decision to "reward" or "punish" the corresponding decision of the routing algorithm so that its future decisions are more likely meet the desired QoS goal. The goal is the metric which characterizes the success of the outcome, such as packet delay, loss, hop count, jitter and so on. The specific goal used by a SP will depend on the user's QoS requirements. When a SP arrives at its destination, an *acknowledgment (ACK) packet* is generated and the ACK stores the "reverse route" and the measurement data collected by the SP. For each SP, a reverse route is first constructed at the destination from the SP's forward route, examining it from right (destination) to left (source), and removing any sequences of nodes which begin and end in the same node. For instance, the path $< a, b, c, d, a, f, g, h, c, l, m >$ will result in the reverse route $< m, l, c, b, a >$. Note that the reverse route is not necessarily the shortest reverse path, nor the one resulting in the best QoS. The ACK that goes back to the source as a result of a SP will then travel along the "reverse route", and the "reverse of the reverse route" is then used as the source route by subsequent DPs of the same QoS class having the same destination, until a new route is brought back by another ACK. ACKs deposit QoS measurement data in the *mailboxes (MBs)* of the nodes they visit. Each MB is organized as a Least-Recently-Used (LRU) stack. The entries in MB are identified with the QoS class and destination. *Dumb Packets (DP)* carry payload and use dynamic source routing.

Random neural networks (RNN) [7] at each node, where each output link of a node is represented by a neuron, together with reinforcement learning, are used to implement CPN routing. The RNN is an analytically-tractable spiked random neural network model whose mathematical structure is akin to that of queuing networks. Given the goal G (hop count or delay, or a combination) we formulate a reward R which is simply $R = \frac{1}{\beta G}$ where β is a constant. Successive measured values of R are denoted by $R_l, l = 1, 2, \cdots$. These are used to compute a decision threshold:

$$T_l = \alpha T_{l-1} + (1 - \alpha) R_l \qquad (1)$$

where α is some constant ($0 < \alpha < 1$) which determines the algorithm's memory. R_l is the most recently measured value of reward. The RL algorithm uses T_l to keep track of the historical value of the reward. Suppose we have made the lth decision which corresponds to output link (neuron) j and that the lth reward calculated for the QoS information received from the network is R_l. We first determine whether R_l is larger than, or equal to, the threshold T_{l-1}. If this is the case, then we first add R_l to all the excitatory weights going into neuron j to reward this neuron for its success, and then normalise all weights so that the sum of all weights remains constant, resulting in an increase of the excitatory weights going into j, and reducing all the other weights. If the R_l is less than T_{l-1}, then we add R_l to the inhibitory weights leading to neuron j, then normalise all weights so as to keep the sum constant, resulting in an increase of the inhibitory weights going into node j and a decrease in other weights. Finally, the state probabilities of each neuron are computed and the SP is forwarded to the output link which corresponds to the neuron with the largest excitation probability.

The arrival of a SP to a node triggers the execution of the RNN algorithm and the output link corresponding to the most excited neuron is chosen as the routing decision. The weights of the RNN are updated so that decisions are reinforced or weakened depending on how they contribute to the success of the QoS goal.

2.1 Experiments with CPN

Experiments are run on a wired test-bed consisting of 17 nodes shown in Figure 1, which was chosen because it offers ease of programming to modify the routing algorithms, more so than would have been possible if we had used simple sensor nodes, and realistic timing values for processing times and network delays. Each node consists of 2.4 GHz Pentium-4 PCs running Linux into which the test-bed software has been integrated. The left-most node in the figure is used as the source, while the right-most is the destination node. The links between each pair of nodes

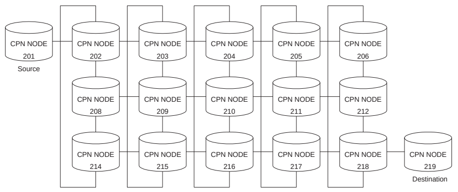


Figure 1: The current test-bed topology

are 10Mbps Ethernet links. All tests were performed using a flow of UDP packets entering the CPN network with constant bit rate (CBR) traffic and the packet size is 1024KB. For each experiment, 10,000 packets were sent out from the source to the destination. In addition, we introduced background traffic to each link in CPN network. The average hop count, forward delay and packets loss rate under different background traffic are reported. We also report the measurements of SP routing algorithm which uses forward delay, the combination of hop count and forward delay as the desired QoS goal respectively. We use *Algorithm-H*, *Algorithm-D* and *Algorithm-HD* to denote the RNN routing algorithms using hop, delay and the combination of hop count and delay as QoS goal, respectively. The length of the shortest path between the source node (#201) to the destination node (#219) is 7 and there are only five different shortest paths in all (see Figure 1).

Figure 2 reports the average number of hops of the routes when different algorithms are used. When hop count is used as the QoS goal, we find that the average number of hops under different background traffic rates are all close to 7. The following figures show the routes used by the first 2000 DPs without background traffic.

Figure 3 shows the routes used when the packet traffic rate is 100 packets/sec. 25 different routes are used in total and one shortest path is discovered. We notice that 1805 of 2000 packets use the route $\langle 201 \rightarrow 202 \rightarrow 203 \rightarrow 204 \rightarrow 205 \rightarrow 206 \rightarrow 218 \rightarrow 219 \rangle$, which is one of the shortest routes. When the traffic rate is 1000 packets/sec (Figure 4), 12 routes are used and four of them represent the shortest paths. In this case, only 1021 packets follow the shortest paths.

From the above results, we can draw the conclusion: when *Algorithm-H* is used, the shortest paths

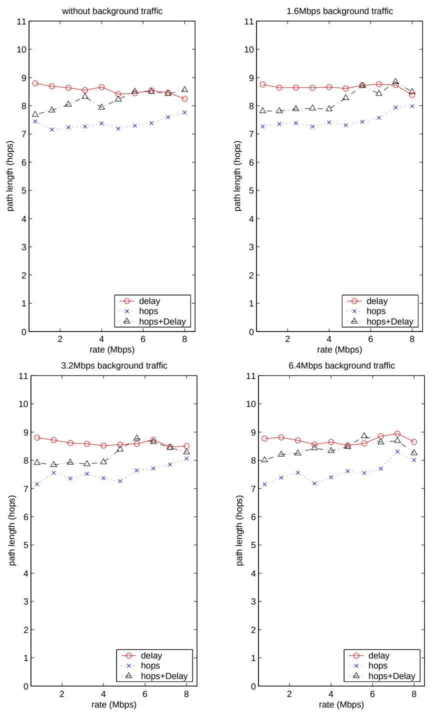


Figure 2: Path length comparison

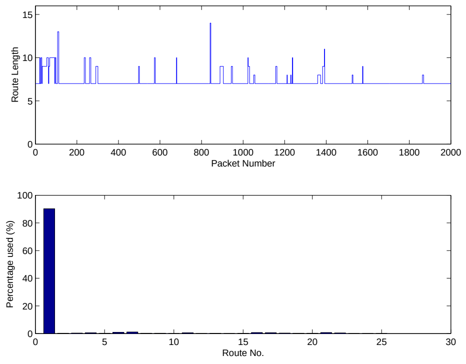


Figure 3: Route usage at low traffic rate

are discovered by the the SPs and are used by most of the DPs. We note that most of the DPs keep on using the same shortest path even if more than one have been discovered. That is because the topology of our test-bed does not change and thus the shortest path does not change either. Once a shortest path is discovered and used, the positive feedback brought back by the ACKs will always reward the previous choice so that the RNN will choose the same shortest path. When the traffic rate is very high, the RNNs do not get enough QoS information from the ACKs before the first packets are sent out. That is why only 12 routes were discovered and there was no usage preference on the routes when the traffic rate is

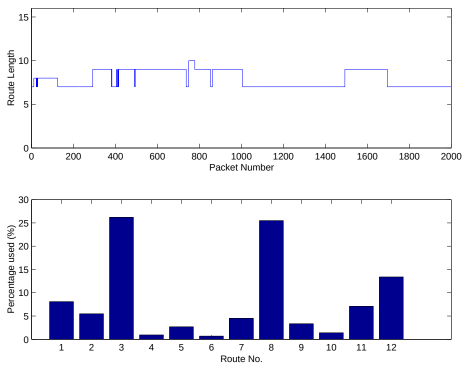


Figure 4: Usage of routes with high traffic rate

1000 packets/sec.

When forward delay is used as the desired QoS goal, the average number of hops is close to 9 (Figure 2). Figure 5, and 6 show the routes information when the delay is used as the QoS goal. Comparing to the *Algorithm-H*, more routes are discovered and used by the dumb packets. The numbers of routes used are 40, 35 and 15 when the packet traffic rate is 100 packets/sec, 500 packets/sec and 1000 packets/sec respectively. The reason is that the more a route is used, the larger the delay over that route will be. So the RNNs can pick different routes to use for the DPs.

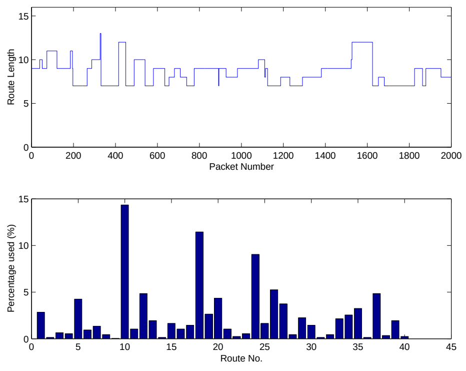


Figure 5: Usage of routes with low traffic rate and delay as the QoS goal

Figure 7 and 8 show how the routes are used when the combination of hop count and delay is used as the QoS goal. The numbers of routes discovered by the smart packets are 43, and 12 when the packet rate is 100 packets/sec and 1000 packets/sec respectively. From Figure 2, we can see that the average path length is close to 8 when the packet rate is low or medium. If the traffic rate is high, the average path length is close to 9. If we only consider the path length of the routes, *Algorithm-H* is the best, while

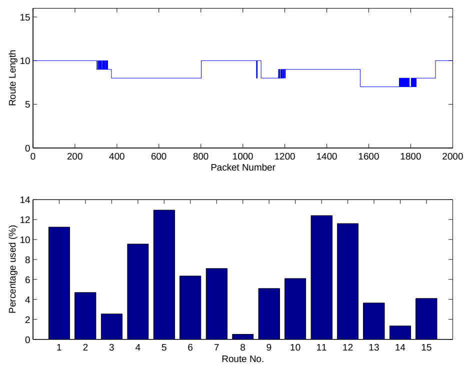


Figure 6: Usage of routes with high traffic rate and delay as the QoS goal

Algorithm-HD is better than *Algorithm-D*.

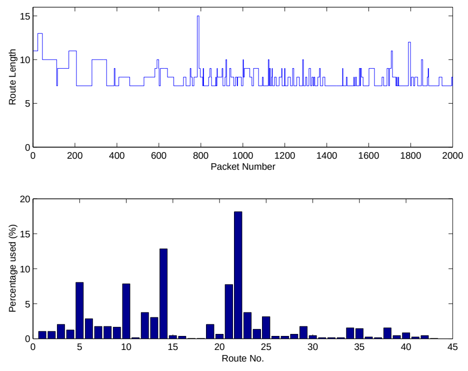


Figure 7: Usage of routes with low traffic rate

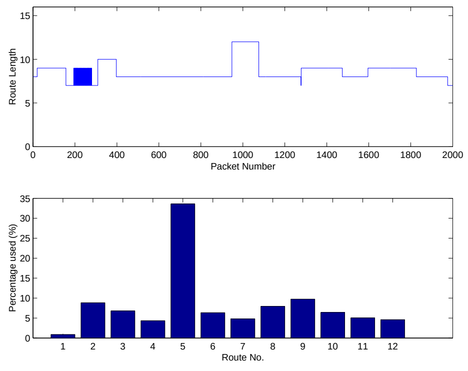


Figure 8: Usage of routes with high traffic rate

We also measure the forward delay using the round trip delay divided by two. As for the packet loss rate, we keep track of the number of packets which are sent out from the source node, s, and the number of packets which are received by the destination node, r. The packets loss rate is then expressed by r/s.

From Figure 9, we note that if the packet rate is

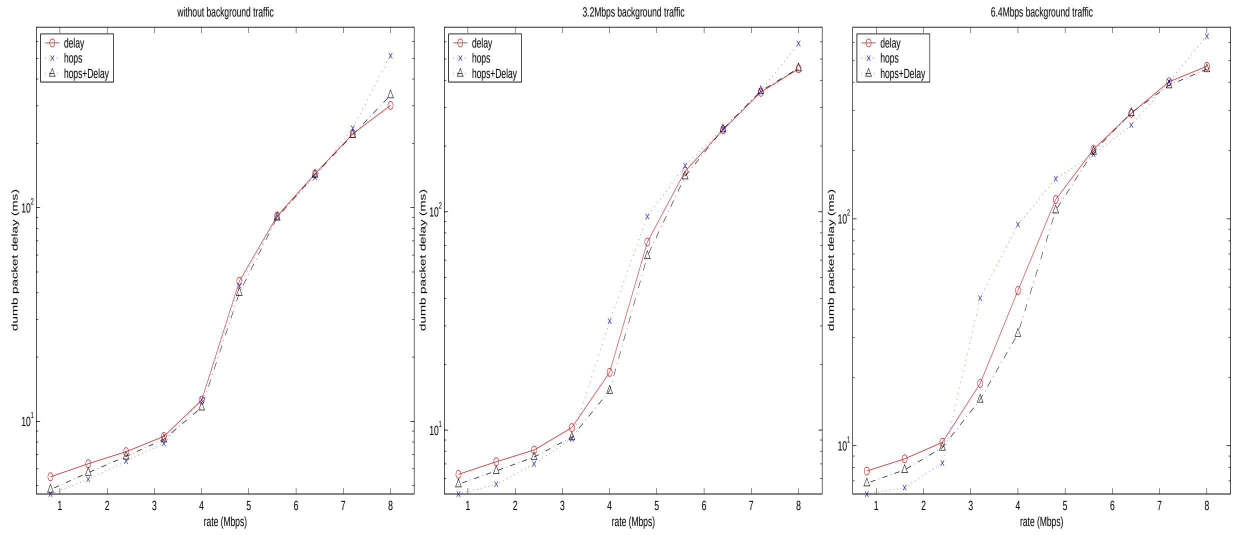


Figure 9: Delay with 0Mbps, 3.2Mbps and 6.4Mbps background traffic

less than or equal to 3.2Mbps and if the forward delay is used as a criterion, the performance of *Algorithm-H* is the best. When the network load is low, the shortest path has the lowest forward delay. The performance of *Algorithm-D* is worse than the other two. When traffic rate is between 3.2Mbps and 5.6Mbps, the performance of *Algorithm-HD* is a little bit better. The average forward delays of *Algorithm-H* and *Algorithm-D* are almost the same. We can not tell the difference between these three algorithms when the traffic rate is between 5.6Mpbs and 7Mbps. When the traffic rate is extremely high (>7Mbps) *Algorithm-D* performs the best and *Algorithm-H* is the worst.

Then, we set the background traffic to 3.2Mbps. When the traffic rate is less than 3.2Mbps, *Algorithm-H* is the best, then *Algorithm-HD* and again *Algorithm-D* is the worst. When the traffic rate is between 3.2Mbps and 6.4Mpbs, *Algorithm-HD* is the best and *Algorithm-H* is the worst, and when it is higher than 6.4Mbps, *Algorithm-D* performs almost the same as *Algorithm-HD* and both of them are better than *Algorithm-H*. We have a similar conclusion when the background traffic is increased to 6.4Mbps except that the threshold points change. Thus if the network topology is stable and the network is lightly loaded, the shortest path is the best choice with respect to the forward delay. One could expect that this would remain true at heavier loads, since the delay on a path is simply the sum of the delays through each node in a path, plus the link delays, so that the longer the path is, the greater the delay might be. However our experiments indicate that, contrary to this common sense analysis which follows commonly held beliefs in the Internet community, when the network is heavily loaded or saturated, the shortest paths may well not provide the shortest delay. It is then preferable to examine the paths with respect to delay and adaptively direct traffic to the ones that instantaneously offer a lower value of delay.

2.2 Adaptation to traffic overload

Another characteristic of the CPN algorithm is that it provides rapid adaptation to overload. To illustrate this property we present some experiments using the network topology in Figure 10, where we report measurements for a main flow of traffic from Node 10 (left) to Node 5 (right). For the first set of experiments, the input rate was fixed to 5 packets second, and there can be a flow of "obstructing" traffic over and above the main traffic flow. in excess of 5.7Mbs per obstructed link, in each direction. The $x-axis$ in all the plots refers to successive packets and it is scaled in packet counts. In each plot the $y-axis$ presents delay in milliseconds (left) or route number (right). Note that all time values are rounded up to the closest integer number of milliseconds, while the route numberings are indicated in the figure captions. On the plots, an "X" under the $x-axis$ in the route plot indicates an instance of packet loss of either smart or dumb packets. Figure 11 shows a traffic pattern with 20% of SPs and 80% of DPs. Here the network is only carrying the traffic from Node 10 to Node 5 with no interfering traffic. The successive values of the delay and the individual routes for each successive packet are being traced packet. Each route label corresponds to some particular sequence of nodes which are being traversed. We observe that routes do change despite the fact that there is no traffic on the network other than the one that is being traced. This is a consequence of the fact that SPs are being constantly sent to test alternate routes, resulting in the selection of an alternate outgoing link at a node when CPN estimates that another path, other than the one being currently used, will provide a lower delay.

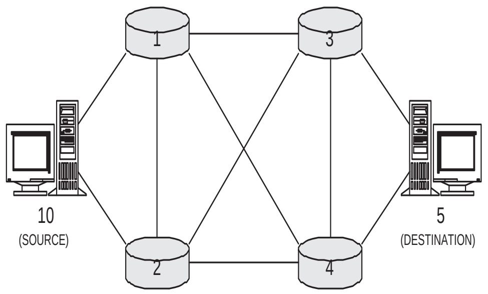


Figure 10: A test-bed topology

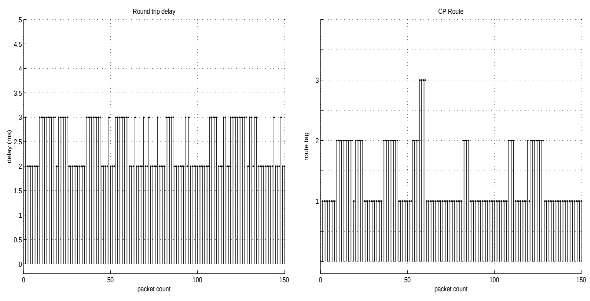


Figure 11: Network with no obstructing traffic.

In Figure 12 we plot the delays and routes experienced by successive packets on the flow from Node 10 to Node 5. We have introduced additional obstructing traffic which perturbs this traffic flow at packet count 30, and we still have 20% of SPs in all traffic flows. The obstructing traffic flows on the link from Node 10 to Node 1 . We see that when the obstructing traffic is initialized, the main traffic flow encounters significant delay as well as loss. Then, thanks to the CPN algorithm, the network determines a new route and the delays go back to a low level. Further spurious increases in delay occur but are short-lived each time the SPs probe the network for better routes, and packet loss does occur again.

3 SSR: Self-Selective Routing

There is a fundamental difference between wireless and wired networks because the basic communication mode is broadcast in the former and point-to-point in the latter. Self-Selection [4] 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 requester. Self-selection employs a prioritized transmission back-off delay scheme in which each node's delay of transmitting a signal is a measure of the node's fitness to perform a pertinent task and in turn, enables the node to autonomously select itself for the task. For the shortest path routing, we use the number of hops from each node to the destination of the given network flow to derive the back-off delay. Such a delay does not only promote avoidance of packet collisions, but it also prioritizes the status of different nodes. Other metrics for back-off delay are possible, combining, for example, the number of hops to destination with the amount of the remaining battery power or with the number of packets already transmitted.

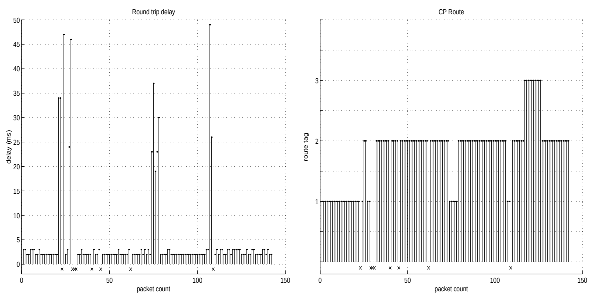


Figure 12: Network with obstructing traffic.

SSR has become a family of protocols, each with different reliability level and all based on the same principal of self-selecting the route at each hop of a multi-hop path. We compare here the basic member of this family, called Self-Healing Routing (SHR) protocol, with its most recent addition, Self-Selective Reliable Routing Protocol (SRP). The protocols share the following basic framework. Each node hearing a packet transmission selects a random time to use as its back-off delay for the given packet. The node whose back-off timer expires first becomes the winner of self-selection and immediately forwards the packet by decreasing the hop count in the packet and broadcasting it. Nodes overhearing this broadcast before their own back-off timers expire cancel their timers. If there is no response to the broadcast, the sender starts the route repair, which is described later. The crucial addition to SRP protocol is that the node that forwarded the previous packet of the same flow sets is delay to nearly zero, making it almost certain winner for forwarding the current winner, with practically no delay of forwarding on the current link. Ramifications of this change on protocol's performance are presented in the following section.

The SSR protocols have been inspired by use of pheromone by ants to mark paths and communicate information about food sources between different insects of the same colony [12]. Accordingly, the SSR protocols consist of two phases: (i) an initial destination request and destination reply flooding that essentially establishes hop distances between each node and their respective flow's destination, and (ii) a data transmission.

The destination request flooding corresponds to the initial search for food in which ants randomly explore the environments and in the process mark the branching paths with pheromone. Packets sent in this stage are referred to as DREQ (Data Request) Packets and they play the same role as Smart Packets (SPs) play in CPN Routing. The destination reply phase corresponds to a walk back to the colony by an ant that found a food source. Walking back, an ant will mark branches on the path home with pheromone to distinguish the return path from others. Packets sent in that stage are called DREP (Data Reply) Packets, and they correspond to ACK packets of CPN Routing. However, unlike CPN Routing, in which SP and ACK packets are repeatedly regenerated, the initial flooding is only conducted once at the sensor network deployment for all potential destinations using the signal-strength aware flooding described in [3].

The data transmission phase corresponds to ants following the marked path to the food source, where at each path fork the strength of the pheromone dictates which branch of the fork to select. In the biological paradigm, each ant traversing the path reinforces the path marker's pheromone strength on repeatedly used paths while pheromone markers of paths not followed will decay with time. By analogy, in SRP protocol, the node winning subsequent self-selections decreases its back-off delay and therefore increases the probability of winning future selection, thereby stabilizing repeatedly traversed paths. This phase of the SRP protocol is summarized in the following subsection of this paper (a more detailed description of this phase in SHR in which the delay is the same for all nodes, regardless of their winning record, is provided in [5], and therefore omitted here). Data Packets sent in this stage correspond to DP packets of CPN Routing. Yet, unlike DP packets, they are also used in route repair and reinforcement of the current best paths.

3.1 Data Transmission in SRP

At each network node and for each flow passing through it, the SRP protocol stores: (i) the ID of the flow's target; (ii) the sequence number of the last packet received from the source; (iii) the hop distance to the target; (iv) a boolean that defines if the current node won the last election; and (v) if the node won the previous election, the sequence number of the then forwarded packet.

The data transmission phase of the SRP protocol uses DATA and ACK packets that contain: (i) the flow source's ID; (ii) the currently processed packet's unique sequence number; (iii) the destination's ID; (iv) the count of hops that the packet has traveled so far; (v) the expected number of hops to the destination; (vi) the maximum number of hops that the packet can travel before it is discarded (TTL); and (vii) the payload. The ratio between the maximum and the expected hop counts is a tuning parameter. A large ratio will allow the route repair mechanism to recover from severe breaks in the network topology but may cause the transmission of an excessive number of packets.

When the source transmits a DATA packet, neighbors that hear that broadcast will respond by selecting a transmission back-off delay that is based on their distance to the destination to the distance of the current sender of this packet. The back-off delay is selected between 0 and $\lambda/4$ for nodes 1 hop closer to the destination, delay is between $\lambda/4$ and $\lambda/2$ for nodes more than 1 hop closer, and between $\lambda/2$ and λ for nodes with the same distance. In addition to these time ranges, nodes that won the election for the immediately previous packet in the flow will set their delay to a random number between 0 and twice the time, t_{switch} that it takes the radio to switch from listening to broadcasting. This small delay is long enough to eliminate duplicate winners if a failure of ACK packet transmission after the forwarding of the previous packet allowed multiple winners to arise. Yet, such a small delay practically guarantees that the previously winning node will win again. Hence, once a reliable path is found, it will be exploited as long as possible. The value of λ is a global (the same for all nodes in the network) scaling factor that reduces the probability that the nodes' responses will collide. This bracketing of time, also provides priority to one hop closer links as they are (i) likely to be more reliable than those that are more than one hop closer, and (ii) on a shorter path than those links that make no progress towards its destination. Increasing λ increases the average delay at each hop not on reliable path, but decreases the probability of transmission collisions.

If, during back-off time, the node receives a DATA packet from a node that is closer to the destination than itself, the receiving node cancels both the forwarding of the DATA packet and the corresponding timer.

When the transmission back-off time expires, i.e. when the node becomes a winner of self-selection, then it (i) increments the packet's actual hop count, (ii) sets the expected hop count to the value stored at the node, (iii) marks itself as the winner of the election, (iv) records the packet's sequence number and (v) transmits the packet. Then, the node selects another random interval to hear the response to its broadcast and react if none is heard. Hence, the minimum delay needed must be larger than λ to makes sure that all potential responders clear their timers and forward the packet (we selected 1.1λ plus the transmission time of the largest packet). The randomize range should reduce possibility of multiple nodes overhearing the responses of interfering their reactions (we selected the range of 0.5λ because the number of such multiple nodes is smaller than the number of potential responders to the packet forwarding). During this time interval, the node monitors the carrier to determine if the packet has been forwarded. Ideally, only a single transmission of DATA packet by a node that is closer to the destination is heard. If a second transmission is detected, it is likely sent by a neighbor that is out of the transmission range of the node which sent the first transmission, so the monitoring node sends an ACK packet. The ACK will reduce the number of neighbors that participate in the election for the next DATA packet in this flow.

If the node determines that the packet was not forwarded, it retransmits the packet and continues to listen for another random interval chosen as above. If, during that time, the node receives an ACK or DATA packet, it cancels the timer and ignores all future packets with this sequence number, because forwarding of the packet was successful. If the timer expires, the node undertakes route repair by increasing the destination's hop distance stored at the node by 2, which is the minimum increase of the distance to the destination for this node. Indeed, there are no nodes closer or equal distance to the destination that are alive, so in the best case, the node one hop farther to the destination (that could be reached in one hop from the current node) may still have an alive path to the destination. If the new distance plus the packet's actual hop count is less than the maximum hop count, the node will transmit the packet with the new distance as the expected number of hops. After transmitting the packet, the node will ignore all future packets with this sequence number. The increase of the destination's hop distance stored in the node will also impact the response of this node to the future packets of this flow. Most likely the packets will be redirected to nodes that are closer to the destination than the changed distance of this node.

When the destination receives the DATA packet, it transmits an ACK packet and starts a timer with end-time of 10λ. Any DATA packet received during this time period causes the destination to send another ACK packet. After this time period, the destination ignores all packets with this sequence number, as the payload has already reached the destination. The Finite State Automate for the SRP protocol, encoding different states of a sensor node participating in the protocol as well as the corresponding state transitions has been presented in [2].

3.2 SHR Performance

We simulated a large scale network to compare performance of SRP, SHR and the Ad hoc On-Demand Distance Vector (AODV) routing [16]. The first of these protocols represent the latest development of self selective routing within the family of SSR protocols while the second one is an older, basic version of the protocol used for comparison. AODV is representative of traditional route-based routing protocols which find a single best route to the destination, store it in the source or over the route and use flooding to repair this route when it becomes damaged. It is also typical in its use of acknowledgments to ensure a high delivery ratio at the cost of additional packets sent and received during transmission.

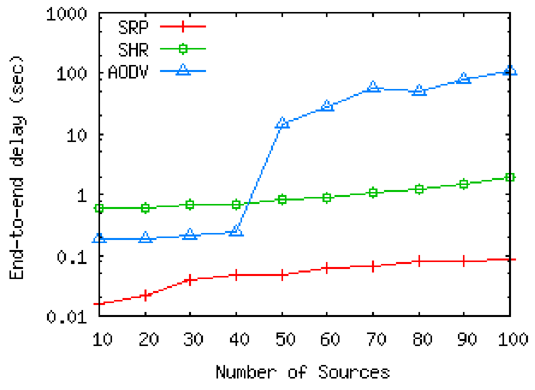
The base configuration for the simulations consists of a 2000 x 2000 square feet terrain populated with 500 nodes, each with a nominal transmission range of 250 feet. We simulated the free space propagation model [17]. The simulated application sends packets of a mean size of 1000 bytes at a mean interval of 40 sec. We performed several simulations and in each we tested the protocols' performance against a change in one of the following test parameter: (i) the rate of permanent node failures; (ii) the rate of transient node failures; and (iii) the number of sources communicating with a single destination (base station).

SRP and SHR used λ =100ms and the maximum hop count equal to the distance to the destination plus $\log_2$ of this distance. The first choice is dictated by the average number of neighbors of each node (which was 7 in our simulations) and the length of the time needed to switch the radio from listening to broadcasting t_{switch} (which was 0.1ms in our simulations). In [18], the probability of broadcast interference and the average delay of the forwarding were found as a function of the number of nodes participating in self-selection and the length of interval over which random back-off delay is chosen. Our choice of λ made this probability 0.25% at the cost of the average delay of each hop without a previous winner equal to 8ms. The second choice influences the packet loss in case of failures and the cost of transmissions needed for the packet to retract back long the severed path before finding the alive path to the destination. Clearly, sublinear function of the path length is needed, and experience indicates that $\log_2$ of the path length provides the best balance between the two above mentioned factors. During the simulations, we collected the communication delay at the destination, the packet delivery ratio at the destination and the total number of MAC layer packets transmitted.

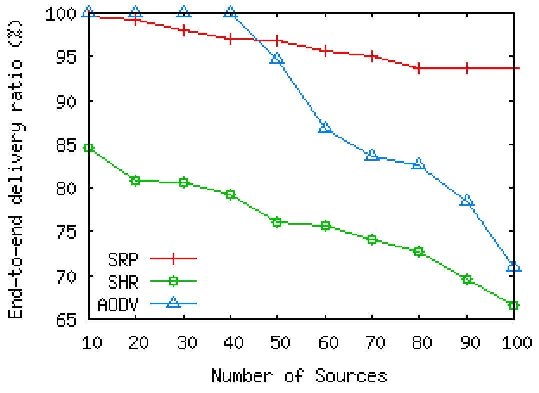
Effect of traffic volume. We tested an impact of increasing the number of sources communicating with a single destination, a situation that is common in wireless sensor networks. The results of this test are shown in Figure 13. Increased traffic causes more random collisions in SHR, decreasing the delivery ratio. AODV maintains higher delivery ratio at the cost of increased number of MAC packets produced and larger communication delay. When the number of sources passes 40, AODV must spend so much time maintaining its topology that its performance drops drastically. SRP on the other hand, maintains an extremely high delivery ratio at very quick speeds despite the large increase in traffic. The huge difference in performance between the SSR family protocols and AODV required the use of logarithmic scale on the end-to-end delay chart. It is also worth noting that since SRP uses so many fewer MAC packets than AODV, power savings becomes an added benefit.

Effect of node failure rate. We tested two node failure modes, transient (see Figure 15) and permanent (see Figure 14). In sensor networks, transient failures are caused mainly by error-prone links, power management induced duty cycles, and packet collisions. Of these, the duty cycle induced failures are the least disruptive since they are often coordinated with the networking protocol. The simulation results presented here are based on a random transient failure model, so they exaggerate the effect of duty cycles on the protocols.

When the topology changes, either by a node failing or returning to the network, extra work is required of the networking protocol. The goal is to minimize this work when the failure is transient, yet quickly update the route when the failure is permanent.



s



s

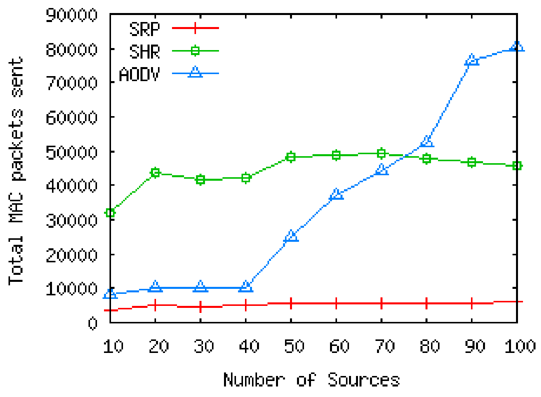
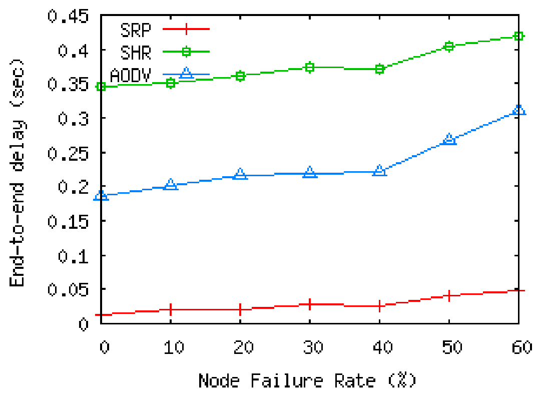
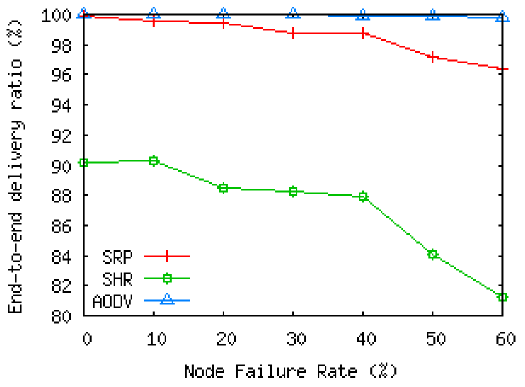


Figure 13: Performance of SRP and SHR versus AODV over a reliable sensor network with increasing number of sources reporting to the single base station

When we introduce a single permanent failure at a fraction of the nodes, both AODV and SRP coped well with the disruption and relatively quickly and efficiently found an alternative route. SRP achieved this with smaller delay and significantly fewer packets than AODV, however with a slightly lower delivery



s



s

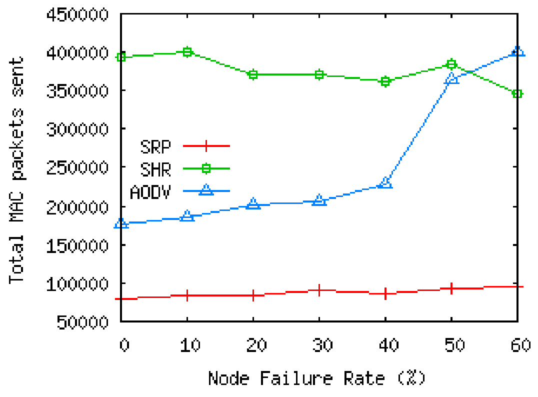
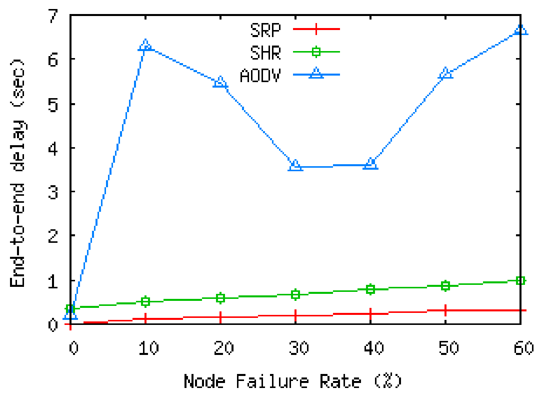


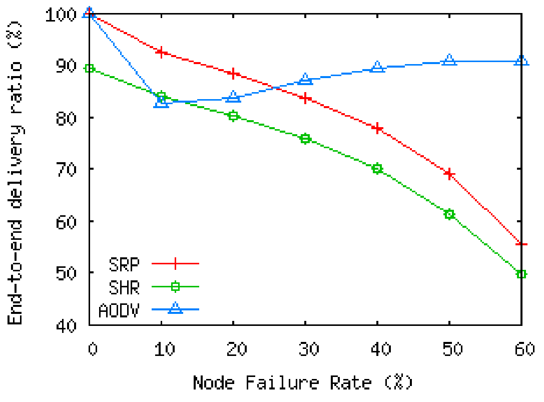
Figure 14: Performance of SRP and SHR versus AODV over a sensor network with permanent failures

ratio.

In case of repetitive failures, AODV is strongly impacted by topology changes. Link layer failure causes AODV to flood the network looking for a new route. The flooding may stop after a few steps, but it is still disruptive. SRP is affected by transient failures (100% delivery rate drops to 57%) but transmits much fewer packets than AODV. As the transient failure rate increases, the failures may overcome SRP's ability to repair routes. A simple solution would be to simply send each packet twice in a transient failure prone network, which would increase delivery ratio, maintain faster speeds than AODV, and still use significantly fewer MAC packets.



s



s

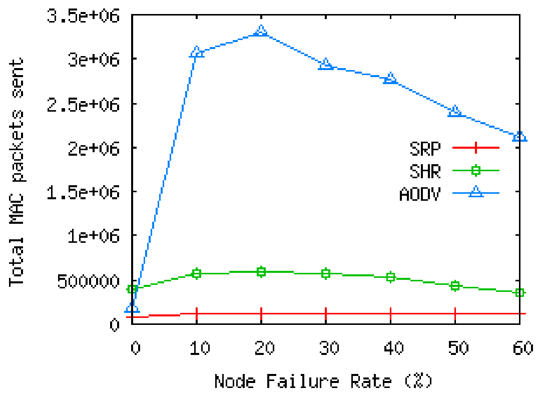


Figure 15: Performance of SPR and SHR versus AODV over a sensor network with transient failures

We also implemented the SHR protocol on Crossbow's MicaZ motes and tested it on a simple topology in which three paths of different length existed between the source and destination [19]. DATA packets were 29 bytes long. TinyOS version 1.1.7 was used with the MicaZ CC2400 radio library extended with the time stamping interface. B-MAC with acknowledgments disabled provided link layer functionality. DATA packets were sent for 12.5 minutes.

For comparison, we used the MintRoute protocol with link-quality estimation to select a parent that minimizes the expected number of hops to the destination. The specific version used is MintRoute v1.7 with the window mean exponentially weighted moving average (WMEWMA) link estimator. All other MintRoute settings were left at their default values. The Surge application was used to send a DATA packet every 5 seconds from the source mote. After sixty packets were sent, we destroyed the shortest path in the tested topology by putting a metal cylinder over one of the nodes in it, thereby removing a communication link on that path.

As is apparent in Figure 16(a), SHR was able to quickly repair a broken route and to find the next shortest and reliable path. Even the longest path of the topology was used to compensate for dropped packets. Hence, the removal of motes is not detrimental to SHR's performance. MintRoute recovered from the broken shortest path but required 150 seconds to do so (see Figure 16(b)). The destruction of motes can be devastating to MintRoute, making it inadequate for a situation where motes can be compromised.

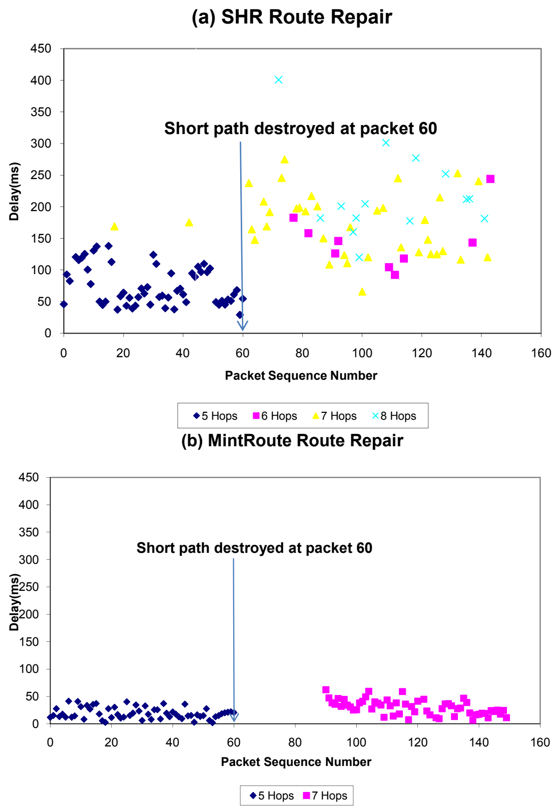


Figure 16: Delivery, delay and hop count of protocols. The shortest path was destroyed after packet with sequence number 60. (a) SHR responds with recovery and increase in hop count when the shortest path is destroyed. The longest path is occasionally used when the medium path is under repair. (b) MintRoute recovers from the failure but after a much longer time reducing its delivery rate significantly

4 Discussion

Although the general structures of CPN and SHR routing protocols are similar, their approaches differ in several important aspects reflecting different assumptions about their deployment and operational use. Both algorithms consist of three distinct stages: discovery, confirmation and transmission.

In CPN, the discovery stage uses Smart Packets (SPs) to traverse possible routes from the source to destination. After SPs reach the destination, the confirmation stage sends ACK packets to the source to establish the best routes and to store them in the source. These two stages are constantly repeated to enable CPN to respond to topology changes. In SHR, the discovery stage followed by the confirmation is performed only once for each flow, or, alternatively once for the entire topology. The discovery stage establishes initial distances to the source for all nodes that lie on potential routes from the source to destination. The DREQ packets are used with a structure similar to SPs of CPN. The confirmation stage uses DREP packets that traverse back the nodes marked by the discovery stage and provides them with the distance to the destination.

In summary, information about routing is distributed to the sources in CPN, with each source storing paths to destinations together with their measured and frequently updated QoS. In addition, each intermediate node in the RNN also stores a ranked list of the alternative next steps which are used to route the SPs. SHR distributes routing information among nodes belonging to potential paths, so each node knows its distance to the destination for any flow passing through it. This difference in information distribution results in radically different transmission stages. In CPN, Dumb Packets (DPs) carry information from the source to the destination, blindly following the path decided by the source. This allows for efficient and fast forwarding in reliable networks, but is vulnerable to transient topology changes that appear and disappear frequently. In SHR, transmission follows the path selected at each node hop, based on the current availability of forwarding nodes (so if the node normally forwarding packets for the given flow is down or its link to the predecessor is down, another node will forward the packet). By shortening the range of randomized delay for a node with the best link among those closer to the destination, SHR supports short delay in stable networks. SHR allows for rapid response to transient failures; however, an additional delay at each hop (to enable collision-free self-selection) will occur when new routes need to be established.

The self-selection of the forwarding node at each hop can take into account additional factors in routing decisions, such as energy level of responding nodes. We are currently working on a version of the SHR protocol that enables energy sensitive routing.

In conclusion, each of the two presented protocols targets different execution environments and different design criteria. When SPs carry payload in CPN, the two approaches become very similar. Both are capable, to a different degree, of supporting fault-tolerance. CPN is more scalable for network with flows to many destination, whereas SHR is more scalable for network with a single destination and many sources (sensor networks with flows from sensor nodes to the base station are the most prominent examples of such networks). CPN is faster in transmitting packets, but slower in responding to topology changes. It works well with networks with low rate of failures and it enables the source nodes to make routing decisions. It also is more scalable in case of many flows with different sources and destinations. SHR works well on unreliable networks with transient links and well interconnected topology with high redundancy of paths between communicating nodes. Further study of application domains for these two algorithms is needed. However, it is clear that the similar overall structure of a protocol (in the discussed case, establishment of three similar stages: discovery, confirmation and transmission) still allows for different levels of efficiency in diverse application contexts.

References

[1] D. Williams and G. Apostolopoulos. QoS Routing Mechanisms and OSPF Extensions. RFC 2676, August 1999.

[2] T.A. Babbitt, C. Morrell, B.K. Szymanski, and J.W. Branch. Self-selecting Reliable Paths for Wireless Sensor Network Routing. *Computer Communications* 20(in print), 2008.

[3] G. Chen, J. Branch, M. Pflug, L. Zhu, and B.K. Szymanski. Sense: a wireless sensor network simulator. *Advances in Pervasive Computing and Networking*, 249–267. Springer, New York, NY, 2004.

[4] G. Chen, J. Branch, and B.K. Szymanski. Local leader election, signal strength aware flooding, and routeless routing. In *5th IEEE Intern. Workshop Algorithms for Wireless, Mobile, Ad Hoc Networks and Sensor Networks, WMAN05*, 2005.

[5] G. Chen, J. Branch, and B.K. Szymanski. A self-selection technique for flooding and routing in wireless ad-hoc networks. *J. Network and Systems Management*, 14(3):359–380, 2006.

[6] S. Chen and K. Nahrstedt. An overview of quality-of-service routing for the next generation high-speed networks: Problems and solutions. *IEEE Network Magazine, Special Issue on Transmission and Distribution of Digital Video*, 12(6):64–79, 1998.

[7] E. Gelenbe. Learning in the Recurrent Random Neural Network. *Neural Computation*, 5(1):154–164, 1993.

[8] E. Gelenbe, M. Gellman, R. Lent, P. Liu, and P. Su. Autonomous smart routing for network qos. *International Conference on Autonomic Computing*, 2004.

[9] E. Gelenbe, and R. Lent. Power aware ad hoc cognitive packet networks. *Ad Hoc Networks*, 2:205–216, 2004.

[10] E. Gelenbe, R. Lent, and Z. Xu. Measurement and performance of a cognitive packet network. *Computer Networks (Amsterdam, Netherlands: 1999)*, 37(6):691–701, December 2001.

[11] U. Halici. Reinforcement learning with internal expectation for the random neural network. *Eur. J. Oper. Res.*, 126(2):288–307, 2000.

[12] S. Koenig, B.K. Szymanski and Y. Liu. Efficient and Inefficient Ant Coverage Methods. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):41–76, 2001.

[13] K. Kowalik and M. Collier. Should QoS routing algorithms prefer shortest paths? *Proc. IEEE International Conference on Communications*, 213–217, 2003.

[14] G. Malkin. RIP Version 2. RFC 2453, November,1998.

[15] J. Moy. OSPF Version 2. RFC 2328, April,1998.

[16] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. RFC 3561, 2003.

[17] T.S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, 1992.

[18] B.K. Szymanski and G.G. Chen. Computing with Time: From Neural Networks to Sensor Networks. *The Computer Journal*, 51(4):511–522, 2008.

[19] K. Wasilewski, J. Branch, M. Lisee, and B.K. Szymanski. Self-healing Routing: a Study in Efficiency and Resiliency of Data Delivery in Wireless Sensor Networks. *Proc. Unattended Ground, Sea, and Air Sensor Technologies and Applications IX*, Orlando, FL, April, pp. 9–13, 2007.