

# Cognitive and Self-Healing Routing for Sensor Networks\*

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

Boleslaw K. Szymanski, Mark Lisee, Kamil Wasilewski  
Rensselaer Polytechnic Institute, USA  
szymansk@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-Healing Routing (SHR) is based on an ant colony paradigm which emulates the pheromone based technique which biological ants use to mark paths and communicate information about paths between different insects of the same colony. 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 SHR 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 [5]. However the shortest path algorithm still plays a very important role in communication networks [12]. In the Internet, Distance Vector [13] and Link State [1, 14] 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-Healing Routing (SHR) paradigm based on the pheromone based technique which biological ants use to mark paths and communicate information about paths between different 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 [9, 7], including power saving [8]. CPN uses *Random Neural Networks with Reinforcement Learning* (RNNRL) [6, 10] to make routing decisions in a distributed fashion at each node. Each output link of a node is represented by a neuron in our RNN. 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 SHR protocol uses broadcast communication, de facto a standard for nodes of wireless sensor networks, and a prioritized transmission back-off delay scheme to enable each receiving node to decide autonomously whether to forward a packet [4]. When severed routes are encountered, the protocol dynamically and locally re-routes packets so they traverse the surviving shortest route.

This paper is organized as follows. Section 2 presents CPN, and experimental results concerning CPN are presented in Section 2.1. SHR is introduced in Section 3 and its operation and route repair algorithm are detailed in Section 3.1. The experimental and simulation results for SHR are presented in Section 3.2. Comparisons between the two discussed protocols and conclusions are presented in Section 4.

---

\*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.

## 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 (RL) based 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 desired QoS. 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 measurements data collected by the SP. The ACK being returned as a result of a SP will travel along the “reverse route”, which is established by taking a SP’s 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  $\langle a, b, c, d, a, f, g, h, c, l, m \rangle$  will result in the reverse route  $\langle m, l, c, b, a \rangle$ . Note that the reverse route is not necessarily the shortest reverse path, nor the one resulting in the best QoS. ACKs deposit QoS measurement data in the *mailboxes (MBs)* of the nodes they visit. *Dumb Packets (DP)* carry payload and use dynamic source routing. The route brought back by an ACK is used as a source route by subsequent DPs of the same QoS class having the same destination, until a new route is brought back by another ACK. A *Mailbox (MB)* in each node is used to store QoS information. Each MB is organized as a Least-Recently-Used (LRU) stack. The entries in MB are identified with the QoS class and destination.

Random neural networks (RNN)[6] 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  $\beta$  is a constant. Successive measured values of  $R$  are denoted by  $R_l, l = 1, 2, \dots$ . These are used to compute a decision threshold:

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

where  $\alpha$  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  $l$ th decision which corresponds to output link (neuron)  $j$  and that the  $l$ th 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 increase very significantly the excitatory weights going into neuron  $j$  and make a small increase of the inhibitory weights leading to other neurons in order to reward it for its success. If the  $R_l$  is less than  $T_{l-1}$ , then we moderately increase the excitatory weights leading to all neurons other than  $j$  and increase significantly the inhibitory weight leading to neuron  $j$  in order to punish it for not having provided an accurate prediction. Finally, the state probabilities of each neuron are computed and the SP is forwarded to the output link which corresponds to the neuron which has 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

The 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. The links between each pair of nodes

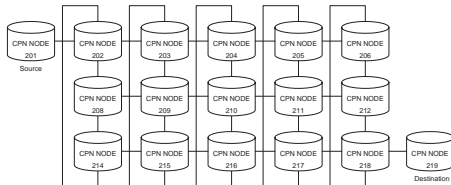


Figure 1: The current test-bed topology

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 the 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).

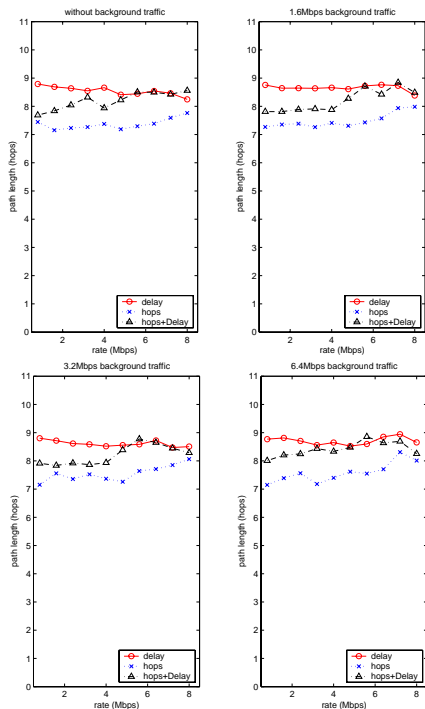


Figure 2: Path length comparison

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.

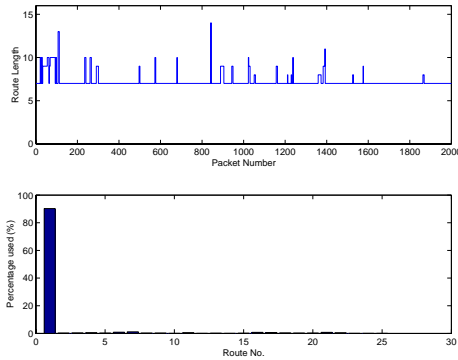


Figure 3: Route usage at low traffic rate

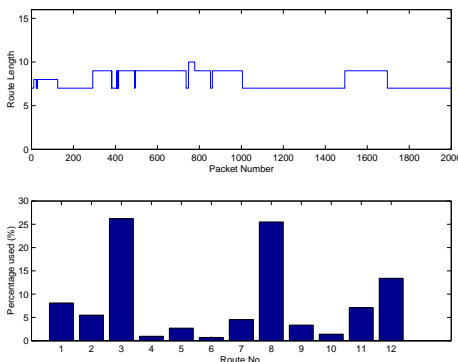


Figure 4: Usage of routes with high traffic rate

From the above results, we can draw the conclusion: when *Algorithm-H* is used, the shortest paths 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 the shortest path is always the shortest path. 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 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 packets traffic rate is 100 packets/sec, 500 packets/sec and 1000 pack-

ets/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.

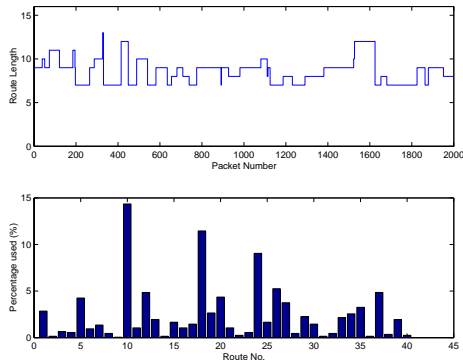


Figure 5: Usage of routes with low traffic rate and delay as the QoS goal

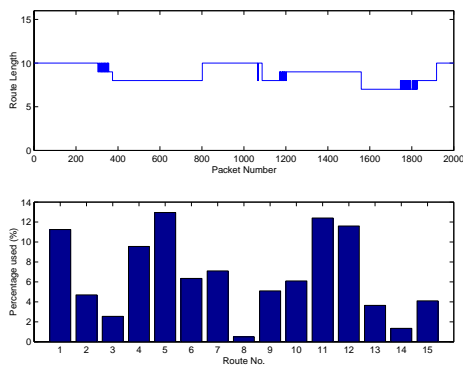


Figure 6: Usage of routes with high traffic rate and delay as the QoS goal

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

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

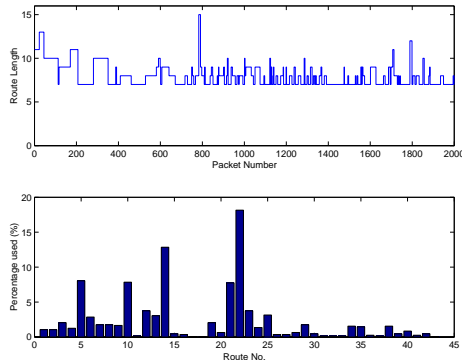


Figure 7: Usage of routes with low traffic rate

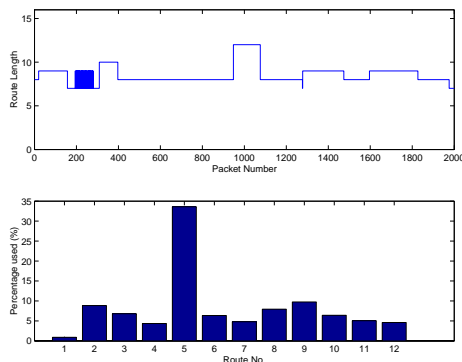


Figure 8: Usage of routes with high traffic rate

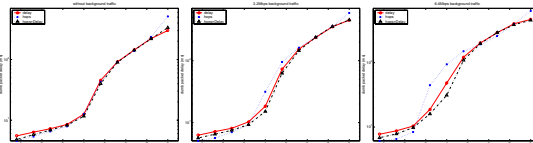


Figure 9: Delay with 0Mbps, 3.2Mbps and 6.4Mbps background traffic

is 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 packets traffic rate is between 5.6Mbps and 7Mbps. When the packets 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 6.4Mbps, *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. When the network is heavily loaded or saturated, sticking to the shortest path is not a good choice and using longer paths may result in better average forward 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  $5.7MB/sec$  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 individual delay values and individual routes taken by close to 200 packets is traced packet by packet. Each route label corresponds to some particular sequence of nodes which are 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 sending out a constant proportion of SPs test alternate routes, resulting in the selection of a new outgoing link at a node if the delay is estimated to be smaller according to the CPN algorithm.

In Figure 12 we plot the delays and routes experienced by individual packets on the traffic flow from Node 10 to Node 5. We now have obstructing traf-

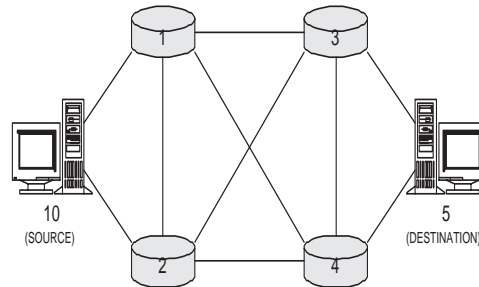


Figure 10: A test-bed topology

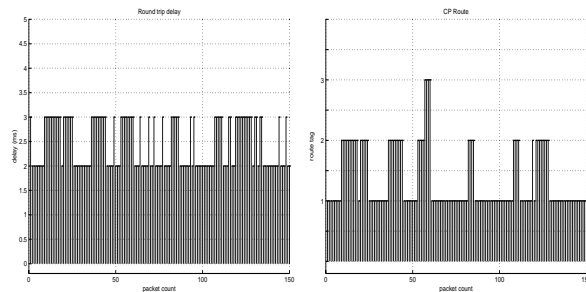


Figure 11: Network with no obstructing traffic.

fic which perturbs this traffic flow, and we still have 20% of SPs in all traffic flows. The additional traffic is introduced when the packet count reaches 30 on the traffic from Node 10 to Node 5. 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 delays as well losses. Then, thanks to the CPN algorithm, the network determines a new route and 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 some losses do occur again. SPs. In addition each  $N$  payload packets also result

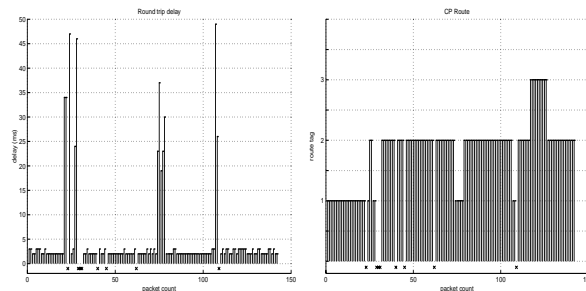


Figure 12: Network with obstructing traffic.

in an ACK packet and the absence of such an ACK after a round-trip delay is an indication of either a (QoS reduction )delay or loss) or of a node failure.

### 3 SHR: Self-Healing Routing

There is a fundamental difference between wireless and wired networks since the latter utilize point-to-point communication while the former use broadcast transmission as the communication primitive. Self-Selection [3] 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 to derive the back-off delay. Not only such a delay promotes 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 packet already transmitted.

One version of a protocol works as follows. Each node hearing a packet transmission calculates its back-off delay. The node whose back-off timer expires first becomes the winner of self-selection and immediately forwards the packet towards destination. On the other hand, the node overhearing forwarding before its own back-off timer expires cancels this timer. However, multiple nodes may choose almost identical back-off delays, leading to a collision of forwarding. Moreover, forwarding by the self-selected leader may be out of the listening range of some nodes, in which case these nodes may self-select themselves and forward duplicate packets. Fortunately, both cases can be easily addressed. If no local leader is elected (likely because of a collision of the forwarding message) at the first attempt, the previous leader can invoke the procedure repeatedly until there is a local leader. Likewise, the previous leader, which naturally is within the transmission range of all nodes involved in self-selection, can broadcast an acknowledgment packet when it hears the forwarded packet.

The SHR protocol has been inspired by the pheromone based technique which biological ants use to mark paths and communicate information about paths between different insects of the same colony [11]. It consists of two phases: (i) an initial destination request and destination reply flooding that essentially establishes hop distances between each node and the flow destination or destinations, 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 the pheromone whose strength defines distance to the home colony (retracing the path, an ant will follow the strongest marks as they were most recently visited on the way out). Packets sent in this stage are referred to as DREQ (Data Request Packet) and they play the same role as Smart Packets (SPs) play in CPN Routing. The destination reply phase corresponds to the walk back to the colony by an ant that found a food source, and again it will mark branches on the path home with pheromone whose strength defines the distance to the food source. 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 could be done once at the sensor network deployment to all potential destinations using the signal-strength aware flooding described in [2].

The data transmission phase corresponds to ants following the marked path to the food source, where at each step the strength of the pheromone dictates which branch on the path to select. This phase is summarized in the following subsection (its more detailed description is provided in [4]). Data Packets sent in this stage correspond to DP packets of CPN Routing. Yet, unlike DP packets, they are also used in route repair.

#### 3.1 Data Transmission in SHR

At each network node and for each flow passing through it, the SHR 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; and (iv) *IgnoreCount* that controls node's participation in a forwarding election for that flow (it's use is further described below).

The data transmission phase of the SHR 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. In addition, the DATA packet contains: (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; 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 the DATA packet, only neighbors that are closer to the destination than the sender will react as follows. If *IgnoreCount* is not zero, then it is decremented by one and the node will ignore this and all future packets with this sequence number. Otherwise, the node selects a transmission back-off time that is uniformly distributed between 0 and  $\lambda$ . The value of  $\lambda$  is a scaling factor that reduces the probability that the nodes' responses will collide. The cost of increasing  $\lambda$  is a corresponding increase in the average delay at each hop. If, during back-off time, a DATA packet is received from a node that is closer to the destination, the receiving node cancels the forwarding of the DATA packet but lets the timer continue. If it receives an ACK or a second DATA packet, it sets *IgnoreCount* to a non-zero value and ignores all future packets with this sequence number.

When the transmission back-off time expires, the node increments the packet's actual hop count, sets the expected hop count to the value stored at the node and transmits the packet. Then, the node selects another random interval within the range  $1.25\lambda$  to  $1.75\lambda$  during which it 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 interval chosen randomly from a range  $1.25\lambda$  to  $1.75\lambda$ . If, during this time, the node receives an ACK or DATA packet, it cancels the timer and ignores all future packets with this sequence number. If the timer expires, the node undertakes route repair by increasing the destination's hop distance stored at the node by 2. 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.

When the destination receives the DATA packet, it transmits an ACK packet and starts a timer with end-time of  $10\lambda$ . Any DATA packets received during this time period cause the destination to send another ACK packet. After thus set time period, the destination ignores all packets with this sequence number.

Finally, to promote reliable links, we introduced a version of SHR called SHR-PP, in which a node after

winning two subsequent self-selections, cuts its value of  $\lambda$  by factor of 5 up to  $\lambda/625$  thereby increasing its chances to self-select itself again. This change, as shown below, resulted in large improvements of the delay and delivery ratio of SHR protocol.

### 3.2 SHR Performance

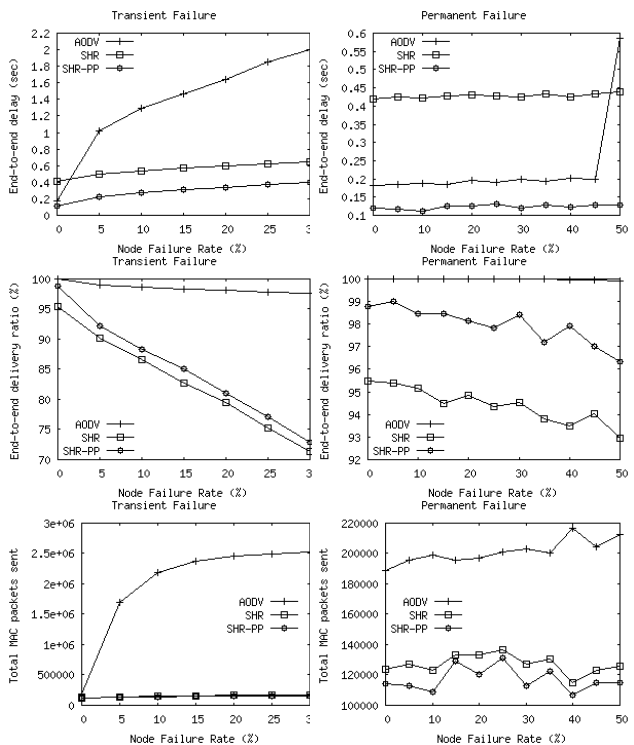


Figure 13: SHR versus AODV protocol performance over the sensor network with nodes' transient failures (left column), and nodes' permanent failures (right column); the first row shows delay, the middle row depicts delivery ratio, and the bottom row shows the total number of MAC packets used.

We simulated a large scale network to compare performance of SHR and AODV [15], the later is representative of traditional route based routing protocols. The base configuration 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 [16]. The simulated application sends packets of a mean size of 1000 bytes at a mean interval of 40 sec. There are five randomly chosen pairs of nodes, each node in a pair sending a traffic to its peer, so there are 10 flows in total. We performed several simulations and in each we tested the protocols' performance against a change in one of the two test parameters: (i) the

rate of permanent node failures, and (ii) the rate of transient node failures. SHR used  $\lambda = 100ms$ , *IgnoreCount* of 9, and the maximum hop count equal to the distance to the destination plus  $\log_2$  of this distance. We gathered the total communication delay on a path, the delivery ratio and the total number of MAC packets transmitted for each protocol tested.

We tested two node failure modes, transient and permanent. In sensor networks, transient failures are caused 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 may be 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.

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. SHR is somewhat affected by transient failures (98% delivery rate drops to 75%) but transmits much fewer packets than AODV (16 times less at 30% transient failure rate). As the transient failure rate increases, the failures may overcome SHR's ability to repair routes. SHR-PP is also faster, from slightly smaller delay with no failures to five times faster for 30% transient failure rate. Similar, but less pronounced trends are shown for permanent failures.

In addition, we 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. 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 min. For comparison, we used MintRoute which uses link-quality estimates to select a parent that minimizes the expected number of hops to the destination. MintRoute v1.7 with the window mean exponentially weighted moving average (WMEWMA) link estimator was used. All other MintRoute settings were left default. 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. SHR was able to quickly repair a broken route and find the next short-

est and reliable path, as seen in Figure 14. 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 14). The destruction of motes can be devastating to MintRoute, making it inadequate for a situation where motes can be compromised.

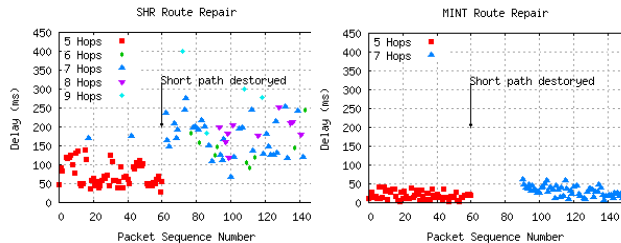


Figure 14: The delivery rate, the communication delay and the route length for SHR and MintRoute protocols when the shortest path was destroyed after packet with sequence number 60

## 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 going 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 network, 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). This approach introduces an additional delay at each hop (to enable collision-free self-selection) but allows for more rapid response to transient failures. 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.

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 with many destination, whereas SHR is more scalable for network with a single destination (base station in sensor networks) and many sources. 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 G. Apostolopoulos. QoS Routing Mechanisms and OSPF Extensions. RFC 2676,

August 1999.

[2] 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.

[3] 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*, 2005.

[4] 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.

[5] 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*, 12(6):64–79, 1998.

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

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

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

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

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

[11] 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.

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

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

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

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

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