M^2RC: Multiplicative-increase/additive-decrease Multipath Routing Control for Wireless Sensor Networks*

HANY MORCOS IBRAHIM MATTA AZER BESTAVROS
{hmorcos, matta, best}@cs.bu.edu
Computer Science Department, Boston University

Abstract

Routing protocols in wireless sensor networks (WSN) face two main challenges: first, the challenging environments in which WSN's are deployed negatively affect the quality of the routing process. Therefore, routing protocols for WSN's should recognize and react to node failures and packet losses. Second, sensor nodes are battery-powered, which makes power a scarce resource. Routing protocols should optimize power consumption to prolong the lifetime of the WSN. In this paper, we present a new adaptive routing protocol for WSN's, we call it M^2RC. M^2RC has two phases: mesh establishment phase and data forwarding phase. In the first phase, M^2RC establishes the routing state to enable multipath data forwarding. In the second phase, M^2RC forwards data packets from the source to the sink. Targeting hop-by-hop reliability, an M^2RC forwarding node waits for an acknowledgement (ACK) that its packets were correctly received at the next neighbor. Based on this feedback, an M^2RC node applies multiplicative-increase/additive-decrease (MIAD) to control the number of neighbors targeted by its packet broadcast. We simulated M^2RC in the ns-2 simulator [4] and compared it to GRAB [1], Max-power, and Min-power routing schemes. Our simulations show that M^2RC achieves the highest throughput with at least 10-30% less consumed power per delivered report in scenarios where a certain number of nodes unexpectedly fail.

1. Introduction

Wireless sensor networks (WSN) are composed of small sensors that are able to sense some phenomenon in the environment, and communicate the sensed data. Data is wirelessly communicated between sensors until it reaches a central processing unit, referred to as the sink. This vision is based on recent technological breakthroughs that enabled the fabrication of small sized, light-weight, and cheap sensor nodes. WSN's have potential applications in many fields. Examples include environmental applications like habitat monitoring and environment monitoring systems [3], military applications like fleet control and monitoring, agricultural applications like crops monitoring, and medical applications like on-line patient monitoring.

Routing protocols in sensor networks should be able to withstand two main challenges. The first is the environmental effect on sensors. Temporary packet losses and node failures are examples of this effect to which routing protocols should react. Moreover, as network conditions may change over time, routing protocols should react to temporary changes in the reliability level of the routing environment. The second challenge is the limited power supply available to sensors. Sensors are battery-powered; hence power usage should be carefully designed in routing protocols to prolong the lifetime of the whole network.

In this paper we present the design of a new routing protocol, Multiplicative-increase/additive-decrease Multipath Routing Control (M^2RC). M^2RC operation has two phases. The goal of the first phase, called the mesh establishment phase, is to setup the routing state of the forwarding nodes. First, it assigns a cost value to each node. The cost of any node is the minimum amount of power needed to go from this node to the sink. Also, during this phase, each node forms its *neighbors* list (routing table). The second phase is the data forwarding phase. After setting the routing state and tasking the sensors, they start sensing the environment. More than one node can detect the stimulus of interest based on the strength of the stimulus signal; one of the nodes is elected to forward its measurements to the sink based on the received signal strength. When sending a data packet back to the sink, forwarding nodes have to decide on the amount of power to use to forward packets. The possible set of such decisions yields a spectrum of routing protocols. Increasing the power used, increases the reliability of the transmission and makes it more immune to packet losses and node failures, but obviously, consumes more power. At one end of the spectrum, every node that receives the data packet may decide to rebroadcast it using maximum power. We will call this scheme Max-power routing. Max-power routing maximizes the probability of successfully sending the packet *over a single hop*. In other words, if nodes fail or packets are lost within the neighborhood of a sensor, using maximum power will enable the transmission to reach the maximum number of neighbors, which maximizes the probability of successful transmission over a single hop. Allowing a sensor node to reach the maximum number of neighbors is effectively maximizing the multiplicity of paths available to that node, which in turn maximizes the reliability of the forwarding process as multiple copies of the data packet are forwarded through each neighbor. In doing so, Max-power routing uses a large amount of transmission power. This behavior causes forwarding nodes to deplete their batteries very fast, which will decrease the lifetime of the whole network.

On the other end of the spectrum, nodes may choose to forward packets using the minimum amount of power, which

*This work was supported in part by NSF grants ITR ANI-0205294, EIA-0202067, ANI-0095988, and ANI-9986397.

routes packets on the minimum-power path from the source to the sink. We call this scheme Min-power scheme as it minimizes the power consumed to deliver packets. Min-power minimizes the probability of losing packets due to collisions. Also it maximizes the lifetime of the sensor network. Contrary to Max-power scheme, Min-power scheme favors short links to long ones, thus taking more short hops to the final destination. Between these two schemes there is a whole spectrum of options that forwarding nodes can choose from. M^{2}RC is one *moving* point on this spectrum. By adapting the number of targeted neighbors based on the current reliability level in the environment, M^{2}RC adapts the topology and varies the number of multipaths available to any node to increase the probability of successful transmission over a single hop using the least amount of power.

Initially, an M^{2}RC node forwards packets to its closet neighbor. A node's job is done concerning any packet once it gets an acknowledgment (ACK) for this packet from the next node toward the source. Otherwise, the node keeps on resending the same data packet using more power. In each subsequent attempt, the node tries to reach more neighbors to maximize the probability of successful delivery. When a node tries to resend the packet to all its neighbors (at the maximum possible power) and still doesn't get an ACK, it gives up on this packet. Inspired by control-theoretic adaptations similar to those widely used in the Internet [5], we designed a Multiplicative-Increase/Additive-Decrease (MIAD) controller to control the number of neighbors to which a packet is forwarded. Other routing schemes do not include such adaptation. GRAB [1] is an example. Under GRAB, nodes forward packets either to 3 neighbors or to a single neighbor irrespective of the current level of reliability in the environment.

We simulated M^{2}RC in the ns-2 simulator [4] and compared it to Min-power scheme, Max-power scheme and GRAB protocol using different node failure models. Our results show that, when the failure conditions are not very severe, M^{2}RC delivers at least as much as other protocols deliver with less amount of consumed energy. M^{2}RC uses about 10-30% less energy than GRAB when a limited number of nodes in the simulation go through temporary blackout periods.

The rest of the paper is organized as follows: Section 2 reviews related work. We then give the design details of M^{2}RC in Section 3. In Section 4, we describe our performance evaluation methodology and the simulation results. Section 5 concludes with future work.

2. Related Work

There has been much interest recently in routing protocols for WSN's. Diffusion [3] establishes multiple single-path routes between the source and the sink. Based on some performance metrics (e.g. delay), the sink evaluates the performance of each route and selects one or more of them to be the primary route(s). The sink reinforces the selected routes by assigning them higher reporting rate. Diffusion uses other routes with low reporting rate to keep them alive and evaluate their quality. Based on path performance, the sink (or any intermediate node on a previously reinforced path) may switch its primary route. This allows diffusion to react locally to route failure or degradation. This technique, however, has one problem pertaining to resilience; its dependence on single primary paths. It has been shown that the performance of multiple single paths is inferior to that of braided paths of the same number [1][6].

GRAB is another routing protocol for WSN's. GRAB assigns a cost value to the forwarding nodes that is proportional to the minimum amount of energy needed to forward packets from this node to the sink. In forwarding data packets from the source to the sink, only nodes with cost less than that of the sender of any packet can forward this packet. This restriction ensures that GRAB is loop-free. Moreover, the source assigns each report/data packet a fixed budget, which is the total amount of energy that may be used to forward this packet to the sink. This budget is not to be exceeded otherwise the packet should be dropped. When receiving a packet, any node X checks if it has enough credit. Credit is calculated by a function that involves the cost of this node and that of the source as well as the amount of power consumed so far to forward the packet. If a node estimates that it has enough credit, it sends the packet to 3 of its closest neighbors. Otherwise, it sends the packet only to the next neighbor on the least-cost path to the sink. This credit distribution function is shown in [1] to allot more credit to nodes closer to the source, which is important to establish a forwarding mesh as fast as possible to overcome node failures or packet losses.

GRAB's fixed credit distribution methodology is independent of the network/routing conditions. A node that has credit will always forward the packet to 3 neighbors (or less, only if it has less than 3 neighbors) while the node that does not have credit will always forward the packet to 1 neighbor. Thus, the *branching factor*, which defines how many neighbors a packet is forwarded to, is fixed and independent of the local routing conditions at the forwarding node. GRAB depends on the redundancy in the forwarding mesh to overcome unreliability in the routing environment. GRAB also calculates the average throughput in a fixed-size window of reports. When this average falls below a certain threshold, the sink reestablishes the cost field in the forwarding nodes to restore the throughput. The underlying assumption is that throughput decreases as a result of node failures which may create holes in the routing tables. Reestablishing the cost field in the remaining nodes would restore throughput. This mechanism is an example of global adaptation to local changes in routing conditions.

The new design philosophy in M^{2}RC is to adapt locally to local changes in routing conditions. This technique allows M^{2}RC to locally react to local routing problems without incurring global overhead over the whole network. This

ensures that energy, a scarce resource in sensor networks, is only used wherever it is needed. The benefit of this design goal is reflected in the results shown in Section 4.

3. M^{2}RC Protocol Details

The operation of M^{2}RC has two phases. The first phase, the mesh establishment phase, sets the routing state in the forwarding nodes. The second phase, the data forwarding phase, is the actual forwarding of data reports.

In the first phase, the sink broadcasts an advertisement packet (ADV). An advertisement packet has a sequence number, and a cost value. The cost value in an ADV packet equals the value of the cost of the node that sent this packet. The cost of a node represents the least amount of power needed to forward packets from this node to the sink. The sink sends ADV packets with cost of 0. Upon receiving an ADV packet from node Y, any node X calculates its own cost as the sum of the cost field in the packet plus the cost to send this packet from Y to X [1]. X sets both its own cost and the cost field in the ADV packet to this sum. Then, X rebroadcasts the ADV packet with the new cost. The cost value increases as we get further from the sink. During this phase each node can determine its neighbors toward the sink. Each node X maintains a list of neighbors toward the sink along with the last cost heard from every neighbor and the physical distance between this neighbor and X.[2] Whenever X gets an ADV from a neighbor whose cost is less than that of X, X adds this neighbor to its *neighbors* list. The *neighbors* list is maintained ordered based on the physical distance from X. The invariant that X has to maintain is that there is no node in the neighbors list whose cost is more than its own cost. This invariant is crucial to avoid routing loops. At any time, X can add any other node M to its *neighbors* list as long as the cost of M is strictly less than that of X. GRAB shares a stripped-down version of this phase with M^{2}RC.

During the data forwarding phase, any forwarding node includes its own cost in the header of data packets. Only nodes with cost less than that of the sender of a packet can forward this packet. This rule along with the way nodes construct their neighbors list ensure that the basic routing mechanism is loop-free. After a node decides it can forward the packet, it has to calculate if this packet still has credit or not. If the packet does not have enough credit, it is only forwarded to the next neighbor on the least-cost path. Otherwise, this node can send the packet to a variable number of neighbors. Each forwarding node keeps, as a part of its routing state, a variable called *branching factor* (henceforth denoted by *bfac*). *bfac* is the number of neighbors to which this node broadcasts packets. To determine the amount of power needed for any transmission, any forwarding node X gets the physical distance between itself and its neighbor numbered *bfac* from its *neighbors* list. Using a certain wave propagation model, X can calculate the amount of power needed to transmit over this distance.

Initially assuming that the routing environment is reliable, all nodes start with a value of 1 for *bfac*, i.e. they forward data packets only to the next neighbor on the least-cost (least-power) path to the sink, even if the packet still has credit. After sending a data packet, X waits for an ACK indicating that the packet it has just sent was correctly received at a node with less cost than that of X (i.e., the packet is received correctly by a node that is closer to the sink).

If X does not get this ACK within a predefined timeout interval, it doubles its *bfac*, resends the packet using the new *bfac*, and waits for an ACK. This process is repeated until either X gets an ACK for the packet or *bfac* grows larger than the number of neighbors of X, which means that X has already sent the packet using the maximum possible power.

Any node can get an ACK either explicitly or implicitly. Referring to Figure 1, when a node X receives a data packet from node Y, X calculates L1, the physical distance between itself and Y. Depending on the value of X's *bfac*, it determines the node to forward the packet to, say node Z, where the distance between X and Z is denoted by L2. If L2 is larger than L1, X knows that by forwarding the packet to Z, Y would be able to receive that transmission and conclude that its packet was received successfully. Hence Y can consider X's transmission to Z an implicit ACK. Otherwise, if L2 is smaller than L1 (Figure 2), Y would not be able to receive X's transmission to Z. In this case, X sends an explicit ACK packet to Y. The ACK packet includes the report number and the cost of X. Upon receiving the ACK packet, Y can consider it an ACK for its packet only if the cost of the sending node (X's cost in this case) is less than Y's cost. Y can perform the same check in the case of implicit ACK since any forwarding node includes its own cost in the header of the packet. Upon receiving an ACK, Y would increment a special counter of the number of reports that were recently acknowledged. When this counter exceeds a certain threshold, Y can assume that the routing conditions got better. Hence it can decrement its *bfac* by 1 (provided it was larger than 1).

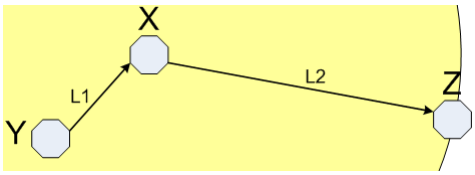

Figure 1: Implicit acknowledgement

This behavior constitutes a Multiplicative Increase/Additive Decrease (MIAD) controller on the *bfac* variable. As mentioned above, controlling the *bfac* of

[1] We assume that nodes can estimate the amount of power needed to send packets based on measured signal-to-noise ratio (SNR).

[2] Nodes can estimate the physical distance between one another knowing the amount of power a packet was sent with and the measured power level with which the same packet was received along with a signal propagation model.

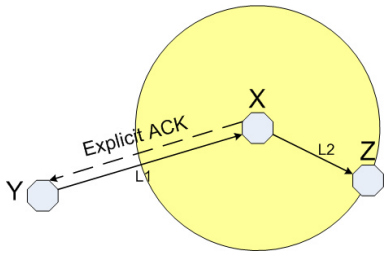

Figure 2: Explicit acknowledgement

each node varies the number of multiple paths available to each node. Thus, M^{2}RC effectively changes the routing toplogy as a function of the current level of reliability in the environment.

This controller is the main contribution of this paper. It is a local controller to react to local changes in the routing environment. The idea of locally reacting to local changes in routing conditions has three advantages:

- Local conditions affect limited parts of the network, while other parts are not affected. Local reaction saves the unaffected parts of the network the overhead of reacting to conditions not directly affecting them. In routing techniques that react globally to local changes, changes in the routing state would be concentrated in the affected area. The nodes in unaffected parts of the network will not show signs of change in the routing state but they still have to consume power while reacting to these conditions;
- Since local reactions are concentrated in a limited area, they would be faster than global reactions which would span the whole network;
- Moreover, local reaction to local changes can be specifically tailored to the routing problem/situation at hand. To achieve the same level of detail using global reaction, if at all possible, a lot of information about the routing condition would have to be passed to a central point (e.g., the sink) where all this information have to be analyzed to modify the routing state in the forwarding nodes in response to these conditions.

M^{2}RC has the following performance optimization measures:

- Whenever a node forwards a data packet, it keeps a record of this data packet in a local cache to avoid forwarding future copies of the same packet. This optimization is shared with GRAB;
- Whenever a node X receives a data packet from a node whose cost is less than X's cost, X checks its local cache for the received packet. If the received packet is not in X's cache, X infers that this is a new packet getting forwarded by another node that is closer to the sink than X. X forwards this new packet in support of the multipath feature;
- When a node X forwards a data packet to node Y, Y gets the packet and send an ACK to X (explicitly or implicitly). If this ACK is lost at X, X would assume that its last transmission was not successfully received at Y, hence, it doubles its *bfac* and re-sends the packet. Later when Y gets the same packet again, it infers that X had lost the last ACK. Y would send an explicit ACK packet to X with a special flag set. This flag tells X that its previous transmission was successful and there was no need to double its *bfac*. When X gets this ACK, it would divide its *bfac* by 2 (i.e., undo the last change to *bfac*).

4. Performance Evaluation and Results

In order to evaluate M^{2}RC we implemented it in ns-2 [4]. We compared M^{2}RC, GRAB, Max-power routing and Min-power routing schemes. We did not include Directed Diffusion as it was shown to be inferior to GRAB in [1]. Our network model is shown in Figure 3. The field size is 2000 meters by 2000 meters with the sink located at (10,1000) and the source located at (1990,1000). The source sends a new report every 5 seconds. The network has 400 nodes uniformly distributed over the field. All nodes have an initial battery power of 150 joules. A simulation runs for 5000 seconds. In our node failure model, in a specified subarea(s) of the sensor field, each node stays up for an exponentially distributed time with an average of 300 seconds, then the node goes to an exponentially distributed (temporary) blackout period with an average that ranges from 60 to 300 seconds. This failure pattern models bursty packet losses in WSN's. The files that specify this failure model were generated offline and then used to subject all routing protocols to the exact same routing conditions. We vary the ratio of blackout-to-uptime to study the performance of the various routing protocols. Our main performance measures are: *throughput*, defined as the percentage of successfully delivered data reports; and *delivery cost*, defined as the power consumed per successfully delivered data report. In the following graphs each point is the average of 20 simulation runs.

Experiment I: The goal of this experiment is to explore the effect of local reactions, which M^{2}RC implements, in a sensor field where node failures are spread over the whole field (i.e., all nodes go through the uptime/blackout cycle described above). This means that potentially many sensors will be off at the same time. In such a "bad" situation, a more aggressive protocol should perform better with respect to throughput. Figure 4 shows the throughput of the four routing schemes. Indeed, GRAB and Max-power achieve higher throughput than M^{2}RC. The reason is that, optimizations employed in M^{2}RC make it less aggressive, which is not very suitable to a persistent failure pattern (note the right-most point in the graph). Min-power routing performs the worst since it does not have any notion of reliable communication. If a packet is sent to a blacked-out neighbor, the packet is lost and the sender does not react to this loss.

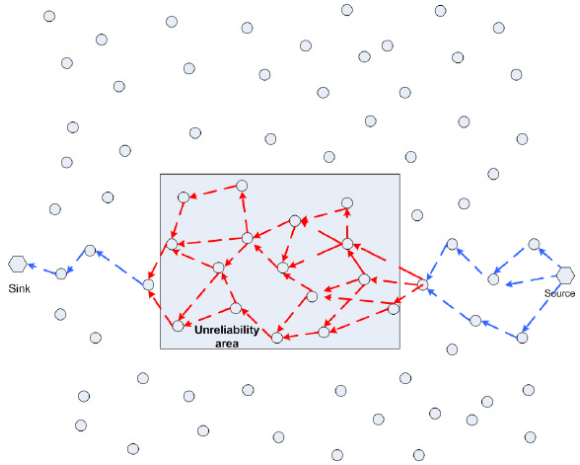

Figure 3: Network model.

Figure 5 shows the average power consumed per successful report. M^{2}RC has 15% saving in power per delivered report compared to GRAB when the average blackout time is 60 seconds (the left-most point in the graph). As the average blackout interval increases, M^{2}RC becomes less effective in saving power. The reason is as we mentioned above, an aggressive protocol like GRAB will deliver better performance in such conditions. M^{2}RC also achieves 25-65% saving compared to Max-power and 50-65% saving in power compared to Min-power.

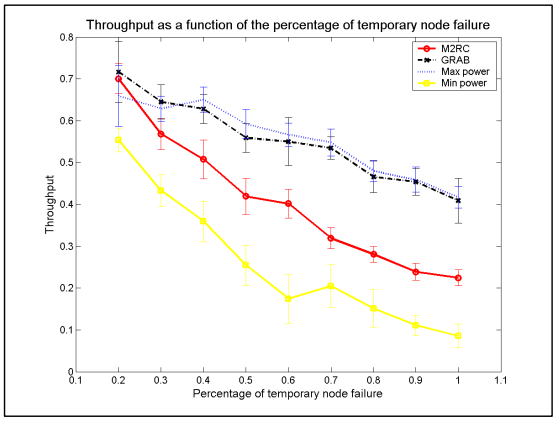

Figure 4: Throughput as a function of average blackout time/uptime

Experiment II: The goal of this experiment is to study the effect of the size and location of the unreliability area on the performance of the routing protocols. Here, the unreliability area is limited to the middle of the sensor field. This new failure model results in less number of nodes in black-out at any given time (compared to Experiment I). Hence, we expect less need for high-powered transmissions and the optimizations implemented in M^{2}RC to pay off more. Figure 6 shows

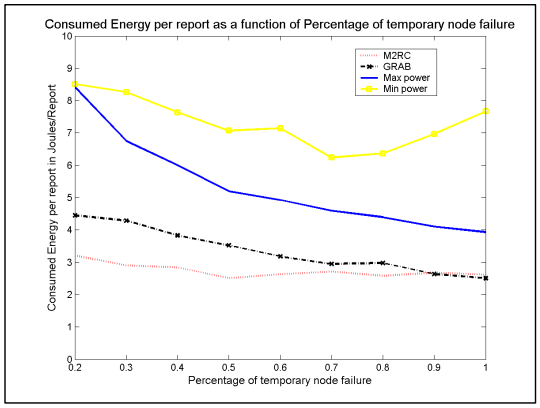

Figure 5: Consumed power per successful report as a function of average blackout time/uptime

the throughput of the four routing schemes. Min-power routing, M^{2}RC and GRAB maintain throughput higher than 85% for all values of blackout times. Max-power, unexpectedly, achieves much less throughput. The reason for that is, with larger regions of alive nodes, and with every node forwarding packets at maximum power, there is potentially a huge number of collisions, which cripple performance. Another reason is that, budget constraints force nodes to drop packets that are out of budget. Using high-powered transmissions consumes the budget early, hence increasing the probability of dropping packets before reaching the sink.

Figure 7 shows the average power consumption per successfully delivered report. M^{2}RC achieves 10-30% saving in power compared to GRAB. The reason for that is the idea of local reaction to local changes in routing conditions. Before and after the unreliability area there is no need to use high-powered transmissions, which helps M^{2}RC save power over GRAB. M^{2}RC only uses higher-powered transmissions when it needs to, i.e., in the unreliability area. M^{2}RC achieves 5-20% saving compared to Min-power, which underscores the benefits of adaptation.

Experiment III: M^{2}RC main contribution is to adapt locally to local changes in routing conditions using an MIAD controller to change the number of target neighbors for each transmission. The motivation behind using an MIAD controller is that it is fast in reconnecting the network in case of partition due to a temporary node failure, and at the same time it converges (albeit slowly/linearly) to the minimum possible value of *bfac* after a number of successful transmissions. The goal of this experiment is to compare different controllers to verify the motivation behind MIAD. To that end, we repeated Experiment II, but this time with two sources instead of one. Figures 8 and 9 show the throughput and the consumed power per successful report, respectively, under different controllers: MIAD, AIAD, MIMD, and AIMD. The results confirm the premise of the MIAD controller—MIAD

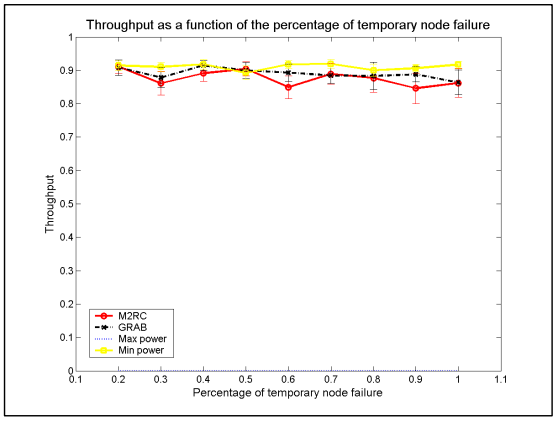

Figure 6: Throughput as a function of average blackout time/uptime

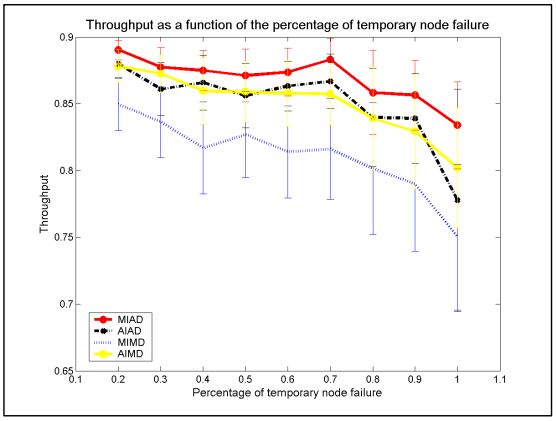

Figure 8: Throughput under different controllers

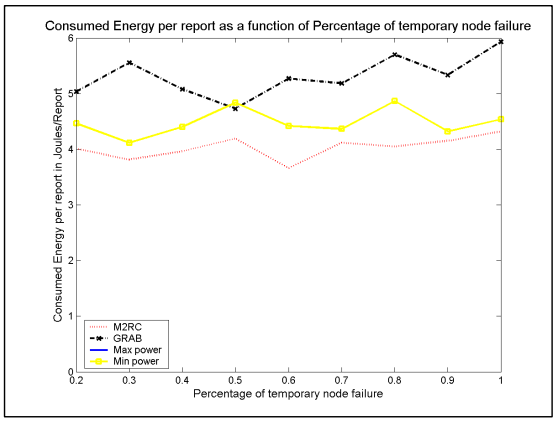

Figure 7: Consumed power per successful report as a function of average blackout time/uptime

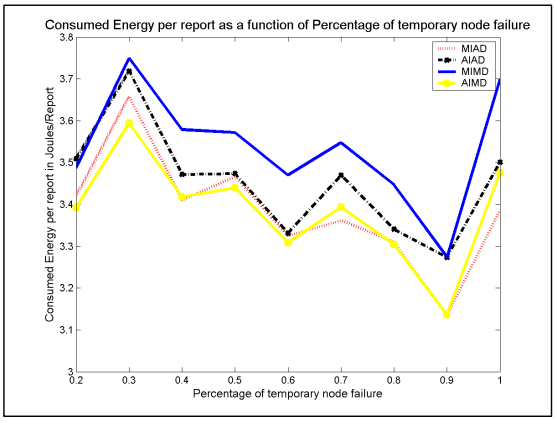

Figure 9: Consumed power per successful report under different controllers

delivers over 83% throughput for all failure/blackout times, with the least amount of consumed power per report.

5. Conclusion

In this paper we presented the design of M^{2}RC, a new routing protocol for WSN's. M^{2}RC is a locally adaptive routing protocol. M^{2}RC adapts to the current level of reliability in the routing environment at every forwarding node using a multiplicative increase/additive decrease (MIAD) controller to adapt the branching factor (i.e. the number of reachable neighbors) of each node. We simulated M^{2}RC in ns-2 and compared it to GRAB, Max-power routing, and Min-power routing. M^{2}RC consistently delivers the best throughput performance using the least amount of consumed power per delivered report.

We are currently investigating the performance of M^{2}RC over a wider range of conditions, node failure and packet loss models. We are also developing an analytical model to better understand the conditions under which M^{2}RC performs better/worse.

6. References

[1] F. Ye, G. Zhong, S. Lu, and L. Zhang. GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks. To appear in *ACM Wireless Networks (WINET)*, Vol. 11, No. 2, March 2005.

[2] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proc. of MobiCOM*, August 2000.

[3] Ning Xu. A Survey of Sensor Network Applications. Computer Science Department, University of Southern California.

[4] The Network Simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[5] Sally Floyd. Congestion Control Principles. RFC 2914, September 2000.

[6] D. Ganesan, R. Govindan, S. Shenker and D. Estrin. Highly Resilient, Energy Efficient Multipath Routing in Wireless Sensor Networks. Mobile Computing and Communications Review (MC2R), Vol 1., No. 2. 2002.