RANDOM WALK ROUTING IN WIRELESS SENSOR NETWORKS

Milad Kharratzadeh

McGill University, ECE Department
milad.kharratzadeh@mail.mcgill.ca

ABSTRACT

In this report, we present a survey of routing techniques in the Wireless Sensor Networks (WSNs), which are based on random walk. WSNs consist of small nodes which are capable of sensing, computation, and communication. These nodes can use up their limited supply of energy performing the computations and wireless communication. Different techniques handle this constraint in different ways (e.g. considering nodes employ duty cycling, or just defining two state (on and off) and assigning a probability to each state). Also, routing protocols might differ depending on the network architecture (e.g. grid or random geometric graphs). First, we review two techniques of random walk routing in WSNs with grid topology which only consider the load balancing over all possible paths from source to destination (with all nodes always on). Then we bring randomness into the previous model by considering on and off states for nodes and assigning probabilities to each, to model the power constraint. Then, we will study the Lukewarm Potato Protocol which is more general and considers a random geometric graph topology, and also employs a duty cycling. Finally, we will try to improve lukewarm potato protocol and compare all the mentioned algorithms.

Index Terms— Wireless Sensor Network, Routing, Random Walk

1. INTRODUCTION

Wireless Sensor Networks (WSNs) consist of a set of sensors that are spatially distributed in order to cooperatively monitor physical or environmental conditions. Advances in processor, memory, and radio technology make it possible to have cheap small sensor nodes which are capable of significant computation and transmitting information in a wireless environment [1]. So, we can use these small nodes to build new networks with a high number of nodes. Since each node has a limited sensing range, with large set of sensors, sensing is best distributed and coordinated. Also, large networks are highly scalable, redundant, and robust in case of node failures and/or environment changes [2]. On the other hand, the problem with these networks is that we do not have centralized control over the whole network anymore. Besides, because of power constraints for nodes of the network, only short-ranged communication is preferred among nodes (i.e. nodes only communicate with a few neighbors), and therefore applications for these networks (including routing) must be distributed [3, 4]. For example in routing, each node can only use local information in choosing the output port to send data to. The areas in which these WSNs are used include environmental monitoring [5], danger alarm, and medical analysis [6].

In general, nodes in a WSN, periodically sense and generate data that needs to be sent to a special node in the network, called sink or gateway node. Since, the sink node is not usually in the communication range of this node, the data need to be send through some intermediate nodes (multi-hop communication) using some routing protocols. The sensed data are classified into three types: large-size data, mid-size data, and small-size data [1]. By small-size, we mean that the size of data is comparable with the size of data exchanged between neighbors for updating local information (e.g. for routing tables). Large-size data are complex data such as images, and mid-size data are something in between . In routing large-size data, we need a lot of initial negotiation between neighbors to find the exact optimum path to destination before sending data, because not routing on the optimum path may use the resources of the network (e.g. power) much more (because of the large size of data). However, in routing small-size data, other routing protocols (such as random walk routing) that do not have special constraints can be chosen as alternative protocols [1]. We will focus on the routing protocols for small-size data here.

In this report we first study a very simple problem of routing on static grid. We suppose that all the nodes are always working (no on-off states or duty cycling), and only want to balance the loads over all possible paths from source to destination, when sending a huge amount of data. The first method by Hui Tian et. al. [1], performs a totally random walk on the nodes in order to send data from source to sink, by choosing one of the neighbors including or excluding the neighbor from which the data came uniformly at random (two different variations). We will show by rough calculations and also by simulations that this method (in both variations) is not practical in any sense. In the second method, proposed by S.D Servetto and G. Barrenechea [7], in each level, the next hop is chosen such that the random walk doesn't deviate from the shortest path. To achieve load balancing, they define the probabilities of each hop in a way that each node on the same diagonal have the same chance of getting the message generated at source. They also modify their algorithm for the case when the nodes can be on or off with some probability to consider the power constraint.

Then, we will review The Lukewarm Potato Protocol proposed by R. Beraldi, R. Baldoni, and R. Prakash [8]. In this random walk routing algorithm nodes employ duty cycling (i.e nodes keep their radio on for only a limited amount of time in each time cycle). Duty cycling is a widely used approach for preserving energy and prolonging the life of a sensor network. In brief, The Lukewarm Potato Protocol, tries to find a tradeoff between routing on the shortest path (which is not available all the time, and so we must wait for it maybe in each hop) and just forwarding out data on the first available outgoing link (hot potato routing). We will review their algorithm and analytical results about the hitting time (total time to send data from source to sink). We also propose some changes in the lukewarm potato protocol to make it work better.

Finally, we will compare the mentioned algorithms and explain their similarities and differences, both in modeling the network and the performance of the proposed routing algorithm.

2. MODELING THE NETWORK AND POWER CONSTRAINT

If all the nodes on the network, keep their transceivers on all the time, then the message can be routed on the shortest path, which can be easily obtained by e.g. methods for constructing minimum spanning trees. This way, the shortest path is always available because all nodes are on all the time. So, the shortest path from source to sink is the fastest path for sending data. However, as we mentioned before, in WSNs we have serious power constraints and thus, keeping the radio always on, will drain the energy supply of sensor nodes, soon. Load balancing, on-off states for nodes, and duty cycling are three major ways of handling power constraint.

In load balancing we naïvely assume that all the nodes are always on and we only need to balance the load (messages) over all possible routes. In other words, we want to design our random walk in a way that all the possible paths from source to destination have the same chance of carrying any message that is generated at source. In grid topology, load balancing is equivalent to having the same probability of receiving the generated messages for all the nodes on the same diagonal, i.e. all the nodes which have the same distance from the source.

Another approach in modeling the power constraint is considering On and Off states for nodes. In this approach, nodes take each of the two states with some probability. Duty cycling is another widely used approach to reduce energy consumption and prolong the life of a WSN. With duty cycling, the active time slots can be scheduled such that within any duration of time, the active periods of neighbors overlap long enough to enable the complete transmission of at least a message.

It is clear that in the second and third approaches, because the next hop is not always available, the nodes need to be able to buffer data. Also, we will have higher end to end delay with respect to the static structure, because of the times that the node must keep the data until the next hop becomes active and ready to get the message.

There are also, different ways to model the network topology. In practical applications of WSNs, two fundamental issues are coverage and connectivity. In coverage, we like to be able to sense the whole physical space of interest. In connectivity, we want all the nodes to be able to communicate with each other either directly or through intermediate nodes. There are two type of topologies of WSNs: regular topologies, and random topologies. There are three major type of regular topologies: hexagon, triangle, and square. Fig. 1 shows these three types of regular topologies. It is shown in [Tian, 10], that the hexagon topology has the best coverage, while the triangle topology has the best reliability (connectivity). The grid topology is something in between and has both good reliability and coverage.

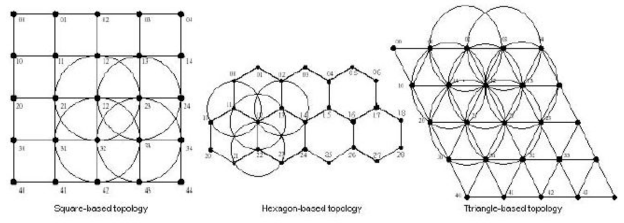


Fig. 1. Regular Topologies (Reproduced from [1])

In Random Geometric Graph (RGG) model, the nodes are scattered unitary at random in a unitary square area, and two nodes are connected to each other if they are in each other's communication range. This topology is more realistic than grid, but it has shown in [9] that the grid can be used as an approximating topology of RGG.

In the following sections we will introduce random walk routing algorithms that modeled the problem using the approaches mentioned in this section.

3. RANDOM WALKS ON REGULAR AND STATIC GRAPHS

In this section we only consider the network with the regular $N \times N$ grid topology shown in Fig. 2. In this topology each sensor has four neighbors (except for the boundary nodes which has either three or two). In this section we assume that the source and the sink are at $[0, 0]$ and $[N-1, N-1]$ respectively (top-left and down-right of the Fig. 2, respectively). In the following, we will study some random walk routing algorithms on the mentioned topology.

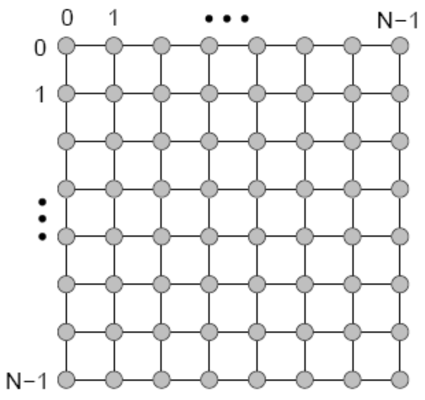


Fig. 2. Grid Topology (Reproduced from [7])

3.1. Totally Random Walk Routing

The method in this section is proposed by Hui Tian et. al. in 2005 [1]. They considered two kinds of routing schemes with random walk. In the first one, a node selects one of its neighbors uniformly at random, with equal probabilities, (**Case 1**), and sends data to it. In the other one, a node selects a neighbor node uniformly at random among all its neighbors except the one from which the data came (**Case 2**). Because of the random nature of the algorithm, all the routes from source to sink in the network have the same chance of carrying the signal and therefore the load is balanced well.

Case 1: In their analysis for performance of the algorithm [1], H. Tian et al assumed that the probability of forwarding the packet in either right or wrong direction is $\frac{1}{2}$. If a message is forwarded in right direction, it will get closer to the sink node (right or down in our introduced grid), and if it's forwarded in wrong direction it will get farther (top or left). In their analysis, they ignore the difference between the internal nodes of the grid and the boundary nodes for simplicity [1]. However, as we will show this may lead to wrong results.

It is clear that the shortest path from source to sink includes $2(N-1)$ steps. So if we take exactly i steps in the wrong direction, we must take $2(N-1)+i$ steps in the right direction to reach the destination (i steps to cancel the wrong steps and $2(N-1)$ steps to get to the sink) which leads to the total number of $2(N-1)+2i$ steps. Thus, for $i = 0, 1, 2, \ldots$ we have [1]:

$$P_1\{d = 2(N-1)+2i\} = \begin{pmatrix} 2(N-1)+2i \\ i \end{pmatrix} \frac{1}{2^{2(N-1)+2i}}$$

In which d is the total number of hops required to reach destination. Therefore the probability of successful transmission within H hops is:

$$P_1\{d \leqslant H\} = \sum_{i=0}^{\frac{H-2(N-1)}{2}} \begin{pmatrix} 2(N-1)+2i \\ i \end{pmatrix} \frac{1}{2^{2(N-1)+2i}}$$

As a numerical example, for $N = 5$, $P_1\{d \leqslant 50\} = 0.9502$, which means that more than $\%95$ of the packets produced in source, will reach sink within 50 hops. However, as we will show, this result is not the thing that happens in reality. The reason is that their assumption about ignoring the difference between internal and boundary nodes, is not a good assumption and highly affects the results. In the next subsection we will show by rough calculations and also simulations that the real results are different from what H. Tian et al achieved, and the reason is disregarding the effect of boundary nodes. But before that, let us consider case 2, as well.

Case 2: In this case also, they ignore the difference between internal and boundary nodes. Since the message is not forwarded to the node it came from, the probability of forwarding in the correct or wrong direction depends on the previous hop. If the last hop was in the correct/wrong direction then the probability of forwarding in the correct/wrong direction would be $\frac{2}{3}$ and the probability of forwarding in the wrong/correct direction would be $\frac{1}{3}$. Then the average probability of forwarding a packet in correct/wrong direction depends on the number of steps taken in correct/wrong direction. Following the same procedure as case 1, we will have a similar expression for $P_2\{d \leqslant H\}$. For the special case of $N = 5$, the calculation shows $P_1\{d \leqslant 32\} = 1.068$ [1]. They said that this probability became more than one, since the boundary nodes are disregarded. They added that as the size of the grid becomes greater this effect will be less and less, and finally it would be negligible. Then they claimed that for the proposed algorithm in case 2, *all* the packets will arrive within four times of the shortest path (which has length 8 in this case). However, as it was mentioned before the boundary nodes have important effect on the hitting time (number of hops taken to reach destination), and can not easily be disregarded. We will show this by rough calculations and simulation in the following subsection.

3.2. Analyzing Totally Random Walk Routing Algorithm

The mentioned algorithm is exactly the classical random walk on graphs, which is studied extensively with the use of Markov Chains. For example it has been shown that [?] that the hitting time for a chain with length N (which is the time needed to reach one end, starting from the other end, performing a random walk), is $O(N^2)$. Clearly, for the grid (which is the two-dimensional version of chain), the hitting time is even more. Actually, with just considering one dimension of the grid, we can say similarly that starting from the source, the time needed to hit the other side of the grid (not exactly the sink), is $O(N^2)$. Therefore, the message should take at least $O(N^2)$ steps before reaching the sink. This is much more than the bound that Tian, et. al. proposed in their work. In the following, we will show that even for small Ns totally random walk routing is not practical.

We performed two experiment, to see how the totally random walk routing works in both cases. In the first experiment, we run the algorithm 1000 times for $N = 5, \ldots, 70$, and calculate the average number of steps needed to send the message to the sink. As it can be seen from Fig. 3 even for small Ns such as 35, the message takes almost 10000 steps on average, before reaching the destination. So, this figure shows that even for small grid sizes, the totally random walk is impractical. Another point that we can understand from this figure is that the algorithm performs better in case 2, which was mentioned by the authors as well.

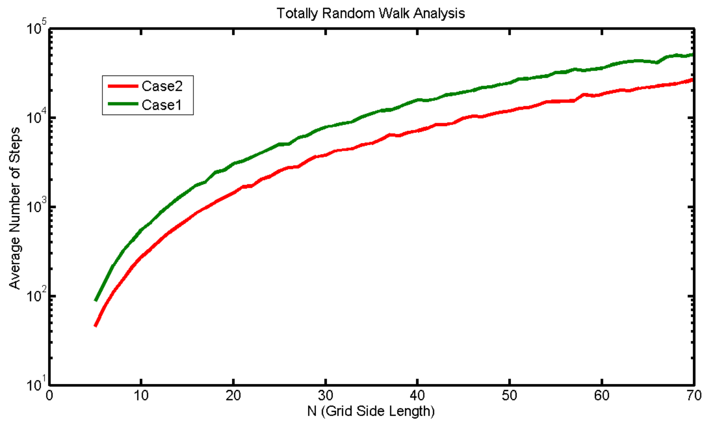


Fig. 3. The average number of steps needed to send the message to the sink in totally random walk routing protocol.

In the next experiment, we calculated the probability of sending the message to the sink within a constant, a, times the shortest path length. What the authors claimed, is that for $a = 4$, the packets will be delivered with probability one. However, it is evident from Fig. 4, as the grid size increases, this probability is going to zero even for $a = 20$.

In summary, as we saw with rough calculations and experimental results, totatlly random walk routing is not practical at all.

3.3. Constrained Random Walk Routing

Constrained Random Walk routing was proposed by Sergio. D. Servetto and Guillermo Barranechea in 2002 [7]. This algorithm also runs on the static grid topology. Here, the nodes only forward to the neighbors on the shortest path, i.e. to one of the right or down neighbors in the grid in Fig. 2. It is evident that the message is always routed on the shortest path, but the load is not necessarily balanced by itself. In this work, the probabilities of forwarding in either direction is adjusted in a way that the load is distributed equally among all paths from source to destination. In other words, if we define a diagonal of the grid as the set of nodes that has the same

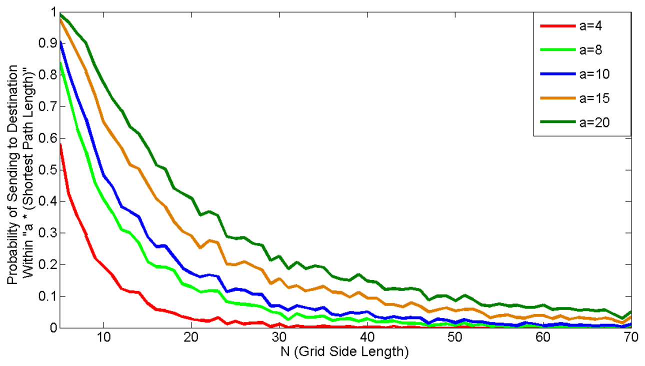


Fig. 4. Probability of forwarding packets to sink in constant, a times the shortest path length in totally random walk routing protocol.

distance from the source, then for complete load balancing, all the nodes on the same diagonal must have the same probability of receiving the signal that was generated at source. However, the adjusted probabilities of forwarding in either directions depends, depend on the position of the sensor in the network. Therefore, a distributed algorithm for computing lattice coordinates must be exploited before we run the routing algorithm [7].

In this section, we studied random walk routing algorithms in static structures, in which all the nodes are always on and we the algorithms only try to balance the load among all potential routs from source to destination. In the next section, we will study algorithms in dynamic structures where nodes are not active all the time.

4. RANDOM WALK ROUTING ON DYNAMIC TOPOLOGIES

4.1. Revised Constrained Random Walk Routing

This algorithm is almost the same as the one mentioned before, unless in this algortihm nodes take two states, On and Off, with certain probabilities [7]. The algorithm for setting forwarding probabilities must be modified in the case of a grid with possibly missing nodes. At a given node $[i, j]$, it may happen that:

(a) both $[i + 1, j]$ and $[i, j + 1]$ are on: In this case, the network locally looks like the regular static grid, and therefore the probabilities are assigned in the same way

(b) only on of $[i + 1, j]$ and $[i, j + 1]$ are on: Probability one is assigned to the active neighbor.

(c) neither $[i + 1, j]$ nor $[i, j + 1]$ are on: Probability one is assigned to the neighbor, which distance to sink is strictly closer than the current node. If such a node doesn't exist, then the current node judt keeps the message in its buffer until one node becomes available.

In this algorithm [7], inaccurate state information can introduce randomness in transport delay in two forms. First, packets can get delayed at intermediate nodes. This can happen in case (c) mentioned above, if the distant estimates from nodes $[i-1, j]$ and $[i, j-1]$ to sink is greater than from $[i, j]$ itself. In this case, the packet is buffered for a random amount of time, until either one of the nodes $[i+1, j]$ and $[i, j+1]$ becomes on or either $[i-1, j]$ or $[i, j-1]$ find a path to sink with distance less than the distance from $[i, j]$. Another form of random delay can be introduced when the packets get misrouted. This also can happen in case (c) when either $[i-1, j]$ or $[i, j-1]$ is closer to sink than $[i, j]$.

The load distribution induced by a random walk with unadjusted forwarding probabilities (based on fair coin toss) and a random walk with adjusted forwarding probabilities (the mentioned algorithm) are shown in Fig. 5. As it can be seen from the two pictures, the load is balanced better in the random walk exploited adjusted probabilities. Actually, for the random walk based on tossing the fair coin, the load is more concentrated on the diagonal of the grid. However, when the forwarding probabilities are adjusted, the load is distributed very well among all the potential routes from source to destination [7].

4.2. The Lukewarm Potato Protocol

This algorithm was proposed by R. Beraldi, R. Baldoni, and R. Prakash in 2009 [9, 8]. Here, duty cycling is used to reduce energy consumption. The duty cycle of each node is determined by superimposition of two kinds of active slots: random active slots, and scheduled slots. The random time slots are d time slots apart where d is a uniform discrete random variable in the range $1, \ldots, K$. So, K is the maximum possible time distance between two consecutive active time slots. A scheduled active slot is an extra active time slot that the node switch its radio on to communicate with one of its neighbors which is active at the same time. A node can do this, because it is assumed that the nodes can predict the active time slots of their neighbors (local information). The reason of considering the scheduled active time slots is that we can exploit them to have flexible forwarding policies [8].

The main concept that is used in the lukewarm potato protocol is that routing on the shortest path does not necessarily mean that the message will be delivered to the sink with the minimum possible delay. The fact is that if the routing algorithm sticks to the shortest path (e.g. minimum spanning tree), then it may suffer from large delays because in each step, it may take a long time until the neighbor on the shortest path becomes active. While, routing on another path which is not the shortest one, but the nodes on the path become active pretty soon in each step, may lead to a smaller end to end delay. In brief, they showed that on average, the delay associated with the shortest path is not the minimum one. Rather, longer paths exist with remarkably lower latency [8].

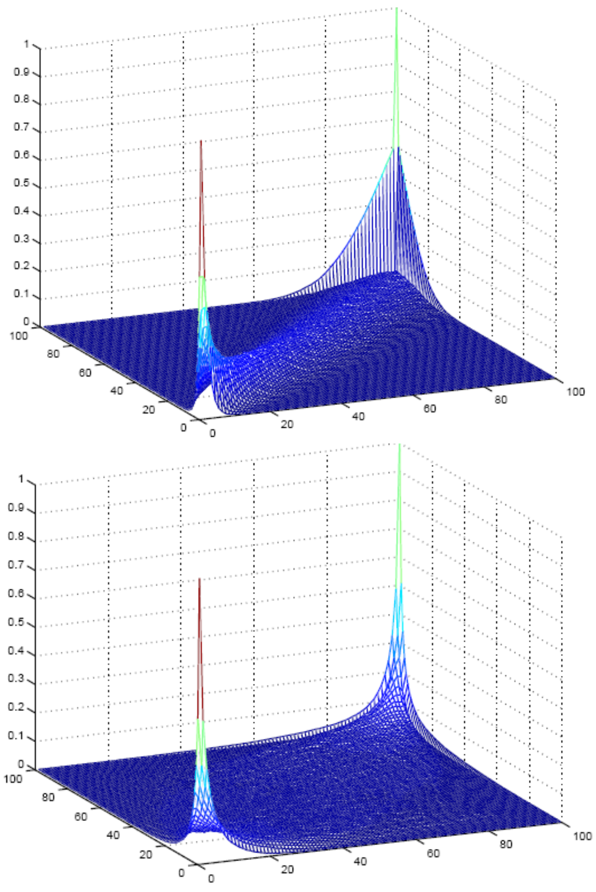


Fig. 5. Load distribution. Top: a random walk with unadjusted probabilities (based on tossing fair coins); bottom: a random walk exploited the proposed algorithm. (Reproduced from [7])

To compare the latencies of the shortest path and fastest path, Beraldi et. al. performed an experiment in which every node flood the message to its neighbors as soon as they become active. The time when the first copy of the message reaches the sink, is the hitting time of the shortest path. The average delay is plotted for both fastest and shortest paths in Fig. 6 as a function of the activity probability.

The mentioned idea is used to propose a protocol which compromises between the shortest path forwarding and hot potato forwarding. Assume that the message is at node F and its parent on the shortest path (e.g. on the minimum spanning tree), is P. Define WT_P, waiting time for node P, as the time that node F should wait for P to be active, after receiving the message.

The lukewark potato routing protocol is as follows [8]:

- If $WT_P < T$, then F sends the message to P (i.e. routing on the shortest path)

- Else, F sends the message to the first neighbor that becomes active (same as hot potato)

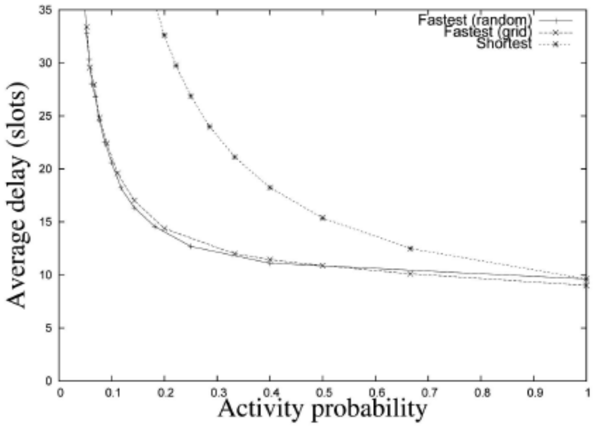
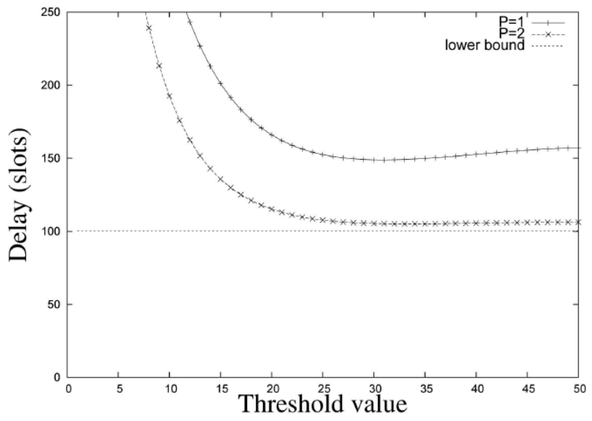


Fig. 6. Average hitting time as a function of activity probability for shortest and fastest paths (Reproduced from [8])

The key idea of the above algorithm is that if P becomes active before a threshold, node F will wait for P and then sends the message to it. Otherwise, it gets rid of the message as soon as possible, which is the same as hot potato protocol. If the threshold T is set to 0, then the lukewarm potato behaves exactly as hot potato protocol. If T is set to K, the maximum time distance between two active time slots, then the protocol always forward the message on the shortest path. And if T is chosen something in between we have a combination of both.

Experimental results in [8], showed that using lukewarm forwarding is not really beneficial in low-density networks, in which there are not many alternative path from source to sink. However, in hogh-density networks, the protocol provides real improvements. Moreover, if the node has more parents on the shortest path, then clearly it will be routed faster. These result are shown in the Fig. 7. In this figure, P is equal to the number of parents on the shortest path. As we can see in the top picture for a wide range of Ts, the delay remains constant. Therefore, for low-density networks lukewarm potato protocol performance is the same as routing on the shortest path ($T = 50$ here). However, for high-density network, the delay of the lukewarm potato protocol (e.g. $T = 10$) is much more better than routing on the shortest path.



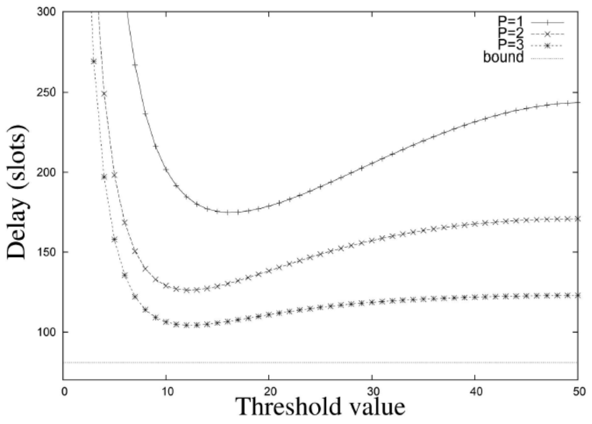


Fig. 7. Average Delay(N=31). Top: Low-Density Network; bottom: High-Density Network. (Reproduced from [8])

4.3. Suggested Modifications

Although the lukewarm potato protocol tried to trade off between the hot potato routing and always routing on the shortest path, this is not all we can do. More specifically, the protocol only considered the parent on the shortest path and if its waiting time is not "too long" the message will be forwarded to that node and otherwise to the first available neighbor. It can be modified if the node considers all its neighbors, and then send the message to the closer neighbor to the sink which becomes active before the threshold T. So, there are more than two choices here and there is a higher chance of getting closer to the sink in each step. Thus, using this approach, the messages will be delivered to sink faster. On the other hand, the nodes need to know their distances from sink. This can be done by a decentralized algorithm at the beginning of the routing protocol.

5. OTHER STATE-OF-THE-ART PROTOCOLS

There are other protocols for random walk routing in wireless sensor networks. A vast majority of them are based on concepts of "query" and "event". Events can be sensor's readings or the computations done in a sensor. Queries can be the request for sensed information or orders to gather more data. In a WSN, a sink can send out queries to different nodes. When a query arrives at a node, the sensed and probably processed data at the node can flow back to the sink, from the same path that query reached the node [10]. When the messages (events) are huge, i.e. the amount of returning data is significant, it is beneficial to try to find the shortest and fastest path from source to sink. This can be done easily by flooding the queries through the whole network [4]. However, when the messages are small, the messages, themselves, can be flooded throughout the network [11]. There are some algorithms such as Rumor Routing [10] that works well between the two extreme cases, i.e. for medium size messages (events).

There are also some protocols that take advantage of the available geographic information to find the best path without flooding [12, 13].

6. COMPARISON AND CONCLUSION

We can compare the mentioned algorithms in two aspects. First, in modeling the power constraint and network topology. Second, in routing performance. In modeling, some of the protocols [1, 7] considered static structures, and some of them [7, 8, 9] considered dynamic structures. And almost all of them considered the grid topology because of its good properties of coverage and connectivity.

Considering performance, we showed that the totally random walk [1] is not practical because of its large hitting time. On the other hand, constrained random walk always routs on the shortest path and also the forwarding probabilities are adjusted such that the load is completely balanced among all routes from source to destination. So, in static structures, this would be the best random walk routing protocol. However, we saw that in this algorithm, nodes need to know their position in the network, and hence we need a decentralized algorithm to do this for us before the actual routing protocol can be run.

In dynamic structures, where nodes might be on or off, the lukewarm potato protocol compromises between the shortest and fastest path and has a good performance in high-density networks. But, as mentioned before this protocol is still a little bit like hot potato in the sense that if the cannot forward to its parent on the shortest path, it will perform hot potato forwarding. Then, we proposed some modifications that the node has more choices than lukewarm potato and can forward to the closer neighbor to the sink that becomes active before a threshold T.

7. REFERENCES

[1] Hui Tian, Hong Shen, and Teruo Matsuzawa, "Randomwalk routing for wireless sensor networks," in *Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies*, Washington, DC, USA, 2005, PDCAT '05, pp. 196–200, IEEE Computer Society.

[2] A. Cerpa and D. Estrin, "Ascent: adaptive self-configuring sensor networks topologies," *Mobile Computing, IEEE Transactions on*, vol. 3, no. 3, pp. 272 – 285, 2004.

[3] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, Seattle, Washington, USA, August 1999, pp. 263–270, ACM.

[4] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks.," in *MOBICOM'00*, 2000, pp. 56–67.

[5] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, may 2001, vol. 4, pp. 2033 –2036 vol.4.

[6] "Medical applications based on wireless sensor networks s. stankovic http://internetjournals.net/journals/tir/2009/july/paper

[7] Sergio D. Servetto and Guillermo Barrenechea, "Constrained random walks on random graphs: routing algorithms for large scale wireless sensor networks," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, New York, NY, USA, 2002, WSNA '02, pp. 12–21, ACM.

[8] R. Beraldi, R. Baldoni, and R. Prakash, "A biased random walk routing protocol for wireless sensor networks: The lukewarm potato protocol," *Mobile Computing, IEEE Transactions on*, vol. 9, no. 11, pp. 1649 –1661, 2010.

[9] R. Beraldi, R. Baldoni, and R. Prakash, "Lukewarm potato forwarding: A biased random walk routing protocol for wireless sensor networks," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on*, 2009, pp. 1 –9.

[10] David Braginsky and Deborah Estrin, "Rumor routing algorthim for sensor networks," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, New York, NY, USA, 2002, WSNA '02, pp. 22–31, ACM.

[11] Fan Ye, Gary Zhong, Songwu Lu, and Lixia Zhang, "Gradient broadcast: a robust data delivery protocol for large scale sensor networks," *Wirel. Netw.*, vol. 11, pp. 285–298, May 2005.

[12] Brad Karp and H. T. Kung, "Gpsr: greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, New York, NY, USA, 2000, MobiCom '00, pp. 243–254, ACM.

[13] Dengfeng Yang, Xueping Li, Rapinder Sawhney, and Xiaorui Wang, "Geographic and energy aware routing in wireless sensor networks," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 4, pp. 61–70, March 2009.