Constrained Random Walks on Random Graphs: Routing Algorithms for Large Scale Wireless Sensor Networks[*]

Sergio D. Servetto
School of Electrical and Computer Engineering
Cornell Univeristy
http://people.ece.cornell.edu/servetto/

Guillermo Barrenechea
Lab. de Communications Audiovisuelles
Ecole Polytechnique Federale de Lausanne
Guillermo.Barrenechea@epfl.ch

ABSTRACT

We consider a routing problem in the context of large scale networks with uncontrolled dynamics. A case of uncontrolled dynamics that has been studied extensively is that of mobile nodes, as this is typically the case in cellular and mobile ad-hoc networks. In this paper however we study routing in the presence of a different type of dynamics: nodes do not move, but instead switch between active and inactive states at random times. Our interest in this case is motivated by the behavior of sensor nodes powered by renewable sources, such as solar cells or ambient vibrations. In this paper we formalize the corresponding routing problem as a problem of constructing suitably constrained random walks on random dynamic graphs. We argue that these random walks should be designed so that their resulting invariant distribution achieves a certain load balancing property, and we give simple distributed algorithms to compute the local parameters for the random walks that achieve the sought behavior. A truly novel feature of our formulation is that the algorithms we obtain are able to route messages along all possible routes between a source and a destination node, *without performing explicit route discovery/repair computations, and without maintaining explicit state information about available routes at the nodes*. To the best of our knowledge, these are the first algorithms that achieve true multipath routing (in a statistical sense), at the complexity of simple stateless operations.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Architecture and Design,Network Protocols; G.3 [**Probability and Statistics**]: Markov Processes, Probabilistic Algorithms

General Terms

Algorithms, Performance, Design, Reliability.

1. INTRODUCTION

[*]Work supported in part by the National Science Foundation under grant CCR 0227676, and by a gift from the Lockheed Martin Corp.

1.1 Networks of Micro-Routers

Wireless networks span a wide spectrum in terms of their functionality (i.e., what they are used for), organization (i.e., how the different components are assembled to form a complete working system), and the technologies used to build them. A long-term project currently under way at Cornell deals with the design and prototyping of networks with the following defining characteristics:

- The nodes operate under severe power constraints, support relatively large data transfer rates, and their number and density is large (e.g., about two dozen per square meter, over a surface of a few square kilometers).
- Once nodes are deployed, their mobility is very limited (if there is any mobility at all). Instead, the main source of uncontrolled dynamics in the network is the temporary failure of individual nodes, typically due to exhaustion of the power source (and for the duration of the "refueling" period).

In this work we refer to the nodes of such a network as *micro-routers*—the network is made up of devices whose functionality is conceptually that of a traditional Cisco router, with the differences that they communicate over a wireless channel, their size and throughput is many orders of magnitude smaller, and they may come equipped with sensors that generate information locally as well. Networks of micro-routers would prove extremely useful in a variety of very relevant scenarios, such as disaster relief operations, military and surveillance applications, cell-size reduction in cellular networks, environmental monitoring, etc.

The development of a working network of micro-routers requires solutions to a number of technical challenges (e.g., routing, flow control, source and channel coding, power control, modem design, hardware, etc.). Among all these, of particular interest in this paper is the *routing* problem, i.e., the problem of moving data among different network locations.

1.2 Complexity and Randomness

Complexity Considerations in Multipath Routing

Implementing a basic packet forwarding service for a network of micro-routers is a challenging problem, for which we are skeptical that experiences drawn from routing in other types of communication networks (such as IP, telephone, mobile ad-hoc, cellular) can be of much help. Instead, we feel radically new approaches to routing are needed in this context.

The high degree of unreliability of the individual micro-routers, combined with large numbers of nodes, strongly calls for *multipath* routing techniques, i.e., techniques in which data flows simultaneously along multiple routes. This is for the simple reason that with many error-prone nodes on any individual route, it is almost a certainty that *some* node along any particular route will fail. However, this is will not be simple to achieve, for two main reasons:

- One is computational complexity: searching a large space of possible routes (derived from having a large number of nodes with high density) may prove computationally prohibitive for low complexity devices like our micro-routers.
- Another is communication complexity: with nodes going up and down all the time, resulting in routes being created and destroyed all the time, the communication overhead required to maintain an accurate picture of even a single route might be prohibitive, let alone doing this for multiple routes simultaneously.

Our main insight presented in this work is that *randomized* algorithms [21] for routing can be used to implement multipath routing, at essentially the cost of having each node implement independent routing decisions plus some minimal overhead. This is illustrated with a simple example next.

Randomized Algorithms

In its simplest possible form, the basic principle on which our routing algorithms are built is as follows. Consider a graph having n nodes with maximum degree bounded by a constant independent of n, each of which stores one bit of information, and let $0 \leq k \leq n$ be a parameter: the goal is to come up with a protocol such that, at the end of its execution, exactly k nodes store 1s, exactly $n - k$ nodes store 0s, and all possible configurations of 1s and 0s are equally likely to occur.

One possible protocol may consist of having a central entity form a list of the $\frac{n!}{k!(n-k)!}$ different possible assignments of 1s and 0s, randomly choose one, and then communicate to each node the value of the bit to store—this solution however has the drawback that it requires the existence of that central entity. To simulate the behavior of the central entity, one could for example create a bucket with k 1s and $n - k$ 0s, and have each of the nodes perform uniform random sampling without replacement from this bucket—now, although the protocol can be implemented in a distributed manner, there is still a substantial overhead in communications to have nodes synchronize access to the shared bucket. Ideally, what we would like is for each node to make a decision about the bit to store *independently* of the decisions made by any other node: in that way, all processing is done locally, and no commmunication with other nodes is required. And it so happens that if we relax only midly the statement of what needs to be accomplished by the protocol, such a solution *is* actually feasible.

Suppose that we are willing to tolerate a fraction ϵ of the time in which the empirical ratio $\frac{k_0}{n}$ (k_0 is the number of 1s in an actual execution of a candidate protocol) satisfies $|\frac{k_0}{n} - \frac{k}{n}| > \epsilon$. But now we observe that *when nodes make independent decisions (e.g., each node tosses a coin that lands on* 1 *with probability* $\frac{k}{n}$*), if the number of nodes* n *is large enough then the Law of Large Numbers [7, Ch. 3] does provide the sought guarantee*. And this is the basic principle we will exploit when setting up our randomized routing strategies. We work under the assumption of large scale networks, and we turn both the size and the unreliability issues that break down stateful routing algorithms to our advantage. Whereas in general it will be difficult to predict the behavior of any individual node, the behavior of a large ensemble of nodes *is* amenable to analysis. Hence, our main goal in this work will be to define routing algorithms executed by a large number of unreliable and loosely coupled components, which give rise to the desired global behavior—efficient routing.

1.3 Random Walks on Random Graphs

There are three main elements in networks of micro-routers that pose serious challenges in the design of routing algorithms: the large number of nodes, the lack of structure in the topology of the network, and the uncontrolled dynamics (ON-OFF transitions) of nodes.

- To deal with the size issue, we are primarily interested in *decentralized* algorithms: that is, algorithms which operate based only on local information, and possibly on information carried by a packet as it moves across the network. In this way, the complexity of these algorithms is independent of the size of the network.
- Routing decisions at each node are based on information the nodes have about the state of other nodes in the network. To deal with the fact that the network does have some uncontrolled dynamics (such as frequent failure of nodes), we are primarily interested in routing algorithms whose dependence of a local decision on the state of other nodes decays with the distance separating these nodes. In this way, we can effectively limit the scope of local updates.
- To deal with the problem of uncontrolled network dynamics, we are primarily interested in routing algorithms capable of taking advantage of the vast number of routes (derived from the size of the network) that would typically exist between any two nodes. However, because of the size as well, it may prove computationally unfeasible to explicitly maintain state information at the nodes describing all these paths—in many cases, such computations involve NP-hard problems [15]. Therefore, our algorithms should take advantage of multiple paths *without* an explicit listing of them.

We see therefore that the main characteristic of interest to us is full decentralization—decentralized computations, involving only local information.

The behavior of large-scale, complex systems has been the object of study in different branches of science for a long time. Among these, the physical sciences provide many examples: formation of crystals, ferromagnetic properties of materials, statistical descriptions of gases, etc. Kelly discusses the notion that fundamental physical/economic concepts such as energy and price can provide useful insights into the design of routing schemes for communication networks [14]. Among these examples, Kelly considers one which essentially inspired most of the work reported in this paper: modeling of interacting particle systems using random walks. At a microscopic level, the behavior of a particle can be described in terms of its position and speed, and random walk models are typically used. At a macroscopic level, that same behavior can be described in terms of quantities such as temperature / pressure / voltage / etc. (depending on the type of particles under consideration), and fluid dynamics equations are typically used. Both descriptions are equivalent, although at different levels of abstraction.

Now, there are many similarities between the motion of particles and routing. If we identify particles with data packets, and the network with the medium in which particles move, then the routing problem becomes a problem of how to “push” a particle from one location in the medium to another. Therefore, inspired by this analogy, we have chosen to formalize the routing problem as a problem of constructing suitably constrained random walks on our graphs. And the main challenge here is to do so under the above specified decentralization constraints.

To construct the desired random walks, we need to address the following issues:

- We need to specify a desired stationary distribution for the random walk to be defined. Ideally, we would like two "macroscopic" properties to hold. First, we would like packets to visit only those nodes which lie on "short" routes between their source and destination nodes, to ensure low delay. Then, subject to this constraint, we would also like the number of packets that visit a node to be independent of the particular node visited—by spreading out the load evenly over a large number of nodes, the impact of node failures is minimized.

- We also need to define a *distributed* algorithm for computing the local parameters of a random walk that results in the desired stationary load distribution—these are local rules which, under the given decentralization constraints, yield the desired macroscopic behavior.

- To make routing performance independent of the size of the network, we require that the algorithm to compute node labels use only: (a) local state information; (b) state information maintained by one-hop neighbors; (c) state information carried by each packet.

In essence, our approach consists of defining random walks with a drift (so that packets move from the source to the destination), and whose parameters can be computed under the given decentralization constraints.

1.4 Related Work

The literature on routing is extensive and spans several disciplines, so we will not attempt to be thorough in this compilation of related work. Instead, we will present a summary of work that most directly influenced ours, while at the same time attempting to keep our list of references at least representative of existing ideas.

Routing using multiple paths has a long history in the context of high-speed networks [17], where it has been proposed as a way of reducing queueing delays in a manner analogous to adaptive routing [18], of dealing with transmission errors [6], and of dealing with system failures [1, 2]. More recently, although a single path ends up being used for routing, parallel multiple route computations were proposed as a mechanism to provide Quality of Service (QoS) in ad-hoc networks [5]—this paper also provides a comprehensive literature survey related to QoS routing.

Recent work by Ganesan et al. proposed *energy-efficient* routing algorithms for sensor networks based on multiple routes, as a means to combat the unreliability of individual sensors [8]. That work provided much inspiration for our work presented here, although there are substantial differences worth pointing out:

- One is that the type of devices assumed in that paper have a "finite lifetime" (they are powered by chemical batteries that will eventually run out) [26], whereas our micro-routers have "infinite lifetimes": we envision them being powered by renewable sources such as ambient vibrations [13], solar cells, or even new ones being the subject of current research (`http://www.darpa.mil/mto/solicitations/BAA01-09/S/pip.pdf`). Therefore energy efficiency and low power operation is important for us to maximize the number of bits that a node can process/transmit while alive. But when batteries run out, the node goes into replenishment mode and eventually it comes back up to life—in this sense, our network never dies, and so the idea that "each bit transmitted brings the network one step closer to death" (that seems to be pervasive in much of the previous work on energy efficiency for sensor networks) is much less of a concern in our setup.

- Ganesan et al. propose two different multipath schemes: one in which multiple disjoint paths are maintained, another in which paths are not disjoint, but interleaved. Yet among all these paths (disjoint or braided), there is one path that is considered "primary", whereas the other ones are maintained as backups to deal with node failures. In the context of a different network though, a similar idea of maintaining backup routes had also been proposed in [20]. In our work, we do away *entirely* with the concept of maintaining route information. There exist multiple routes between source and destination, but each node is completely unaware of this, each node simply randomly chooses one of its neighbors to forward a packet—how to compute locally the pdf that each node has to sample is the core of our technical contribution.

In work of a similar nature, Ganesan, Krishnamachari et al. also study the behavior of "epidemic" (typically, flooding) algorithms in large-scale multi-hop wireless networks [9]. While certainly having data flowing across multiple links in parallel, the focus of that work is on understanding how these algorithms behave in large networks.

Another important body of related work deals with routing problems in mobile ad-hoc networks (e.g., [12, 24, 25]). In this context, routing along multiple paths has also been studied (e.g., [22, 23]), although not as extensively as single-path routing.

Our interest in random graphs, and more specifically in connections between random graphs and routing problem, was sparked by the work of Kleinberg on the algorithmic aspects of the small world phenomenon [16]. Among other results, of interest to us is that in that work it is shown how, for one specific family of random graphs closely related to those considered in this work, there exist fully *decentralized* algorithms of the type we seek to construct here, that can be very efficient at routing messages. Strogatz presents an interesting overview on complex networks [30], and Watts provides an accessible introduction to small worlds [31].

Gupta and Kumar present results on the transport capacity of wireless networks [10]. Scaglione and Servetto interpret those results in terms of the capacity of flows on graphs, and use that formulation to obtain bounds on the rate/distortion function of the whole network, to ensure that a broadcast problem of interest in that work admits a solution [28]. Hajek presents results on how long will it take for a particle undergoing brownian motion with a state-dependent drift to hit a particular spot [11].

Complexity management techniques for the problem of providing fair bandwidth allocations in large networks have been proposed in [29]. And although they do not deal with routing problems specifically, the line of thinking presented in that paper did have a strong influence on our own thinking, by pointing out problems with network algorithms involving complex state and by suggesting approaches to deal with them.

In summary. Routing in a network of devices with the characteristics of our micro-routers is a very different problem from more classical routing problems, such as traditional Internet routing [4, 19]. To the best of our knowledge, ours is the first piece of work in which "stateless routing" (i.e., the idea of routing messages without any notion of discovering / maintaining / repairing explicitly described routes) is dealt with.

1.5 Main Contributions and Organization of the Paper

The main contribution presented in this paper is the construction of random walks on one particular family of random graphs that we have chosen as an abstraction for the behavior of a network of micro-routers, with all the desired decentralization properties men-

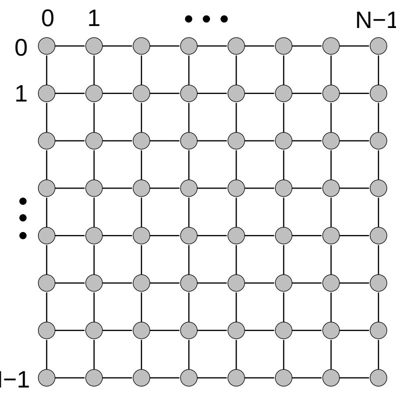


Figure 1: **Cubic grid of size $N \times N$. Packets are injected at the source location $[0,0]$, and must travel hop by hop to the destination node $[N-1, N-1]$. Any interior node $[i,j]$ is connected to 4 neighbors: $[i-1,j]$, $[i,j-1]$, $[i+1,j]$, and $[i,j+1]$; the first two are closer to the source, the latter two are closer to the destination. A completely general random walk on this grid is specified by giving four numbers $\pi_v[i-1,j]$, $\pi_v[i,j-1]$, $\pi_v[i+1,j]$, and $\pi_v[i,j+1]$, for each node $v=[i,j]$ (except at the boundaries, where the number of neighbors is smaller).**

tioned above. Such random walks define a large class of algorithms for each node in the network to execute, to route packets to any destination. At a given node v, let $N(v) = \{u_1 ... u_{n_v}\}$ be the set of neighbors of v. Let $\pi_v = \{p_1 ... p_{n_v}\}$ be real numbers such that $p_i \geq 0$, $\sum p_i = 1$ (a pdf on the neighbors of v). When a packet reaches node v, the next hop is chosen by tossing a die whose i-th face occurs with probability p_i, and the packet is forwarded over the link (v, u_i). By making different assumptions on the topology of the underlying network, on its dynamics, and on constraints imposed on the local pdfs, we are able to explore a large and structured space of possible routing schemes.

The rest of this paper is organized as follows. In Section 2 we formulate and solve analytically for the local parameters of the sought random walks in the context of a static, regular network. Then, we consider increasingly more complex cases that build on this solution. In Section 3, we give routing algorithms for a static network that is obtained as a random perturbation of the regular network considered in Section 2. And in Section 4, we give routing algorithms for networks which result from time-varying perturbations of the regular network of Section 2. Conclusions are future work are discussed in Section 5.

2. RANDOM WALKS ON REGULAR AND STATIC GRAPHS

2.1 Rationale: Solve the Simplest Non-Trivial Problem First

We start by designing suitably constrained random walks for a static graph with a regular structure: a cubic grid, with 4-nearest-neighbor connectivity only. The resulting topology is illustrated in Fig. 1.

There were many reasons that prompted us to start our study of routing algorithms in general random networks with the simple cubic grid: (a) the model is simple enough to allow us to obtain simple closed form expressions for the sought distributions, yet at the same time it is rich enough to allow us to explore issues related to scalability / numbers of nodes; (b) the model is a subset of Kleinberg's model [16];[1] and (c) the constructions of random walks presented here naturally precede the construction of random walks on a more general family of random graphs of interest to us. Observe also that all the constraints described in Section 1 on suitable routing algorithms for our application can be translated into constraints on suitable π_v's. For example:

(c.1) To ensure the avoidance of livelock conditions, we require that if for some destination node d, we have for some $u_i \in N(v)$, that $d(u_i, d) > d(v, d)$, then $\pi_v(u_i) = 0$.

(c.2) To effectively exploit whatever degree of *route diversity* the network provides, we require a certain "load balancing" condition of the stationary distribution π_S induced by the π_v's. Consider two nodes $v_1 = [i_1, j_1]$ and $v_2 = [i_2, j_2]$: we require that if $i_1 + j_1 = i_2 + j_2$, then $\pi_S(v_1) = \pi_S(v_2)$. What this means is that if two nodes are at the same distance from the source, then these nodes must be visited equally often in steady state.

We now proceed to give an algorithm to compute the π_v's.

2.2 Local Parameters of the Random Walk

First we define some notation. G_N is a grid as shown in Fig. 1, of size $N \times N$. The ℓ-th *diagonal* of G_N is the set of all nodes $[i,j] \in G_N$ such that $i + j = \ell$, and is denoted by $D(G_N, \ell)$—to keep the notation simple we will also write $D(\ell)$ for a diagonal, since in this version of the work we will never deal with more than one grid G_N. The *size* of a diagonal is denoted by $|D(\ell)|$. $d_e[i,j]$ is the distance from $[i,j]$ to the nearest node on the boundary of the grid. We also divide the network into two regions:

- In an *expansion* stage, packets move across diagonals with an increasing number of nodes and consequently, the density of packets per node decreases.
- In a *compression* stage, packets move across diagonals with a decreasing number of nodes and consequently, the density of packets per node increases.

These concepts are illustrated in Fig. 2.

Note that other than boundary nodes, for any node $[i,j]$ there are exactly two neighbors on a shortest path from $[i,j]$ to $[N-1, N-1]$; these neighbors have coordinates $[i+1,j]$ and $[i,j+1]$. So, in this particular topology, and under constraint (c.1), a random walk is defined by a single number p, the probability of choosing one of these two links. By convention, we define p to be the probability of forwarding a packet to the neighbor that is closer to the boundary of grid ($1-p$ being the probability of forwarding to the other). And then we have the following result:

For the network of Fig. 1, the value of p at node $[i,j]$ that achieves a uniform distribution on diagonals is

$$p = \frac{|D(i+j)| - d_e[i,j]}{|D(i+j)| + 1} \tag{1}$$

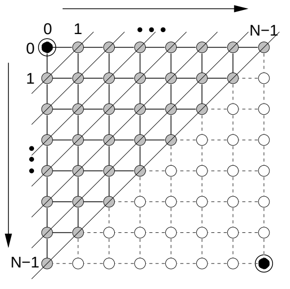
if $[i,j]$ is a node in the expansion stage, and

$$p = \frac{d_e[i,j]}{|D(i+j)| - 1} \tag{2}$$

if $[i,j]$ is a node in the compression stage.

The proof in both cases proceeds by induction on the diagonals. Consider first the expansion stage:

[1]Random walks on small world graphs however are beyond the scope intended for this paper, and will be dealt with elsewhere.



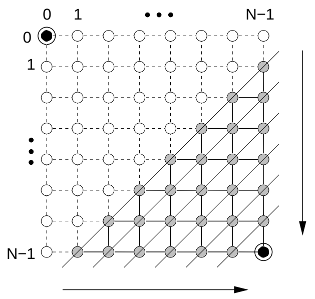


Figure 2: The different stages in the path of a packet from the source $[0,0]$ to the destination $[N-1,N-1]$. In the initial *expansion* stage, the number of nodes among which to spread the packet load increases, and therefore the optimal load per node must decrease. After crossing the longest diagonal (corresponding to nodes with coordinates $i=j$) and entering the *compression* stage, the number of nodes on diagonals starts decreasing, and therefore the load per node must increase.

- The first diagonal corresponds to the source node, and hence $|D(0)| = 1$ and $d_e[0,0] = 0$. It follows from eqn. (1) that $p = \frac{1}{2} = (1-p)$, achieving a uniform packet distribution over second diagonal.
- Let $[i,j]$ be a node located on the diagonal $D(i+j)$. Assume (by induction) that we have already a uniform packet distribution over $D(i+j)$, and so the fractional load supported by each node is $\frac{1}{D(i+j)}$. The next diagonal has $D(i+j)+1$ nodes, and the fractional load we want to achieve is $\frac{1}{D(x,y)+1}$. The situation is depicted in Fig. 2.2. For nodes at distance $0 \le k = d_e[i,j] \le \frac{N}{2}$ from the boundary, their corresponding probability p_k satisfies

$$\frac{1}{|D(i+j)|}\cdot p_0 = \frac{1}{|D(i+j)|+1}$$
$$\frac{1}{|D(i+j)|}\cdot p_1 + \frac{1}{|D(i+j)|}\cdot(1-p_0) = \frac{1}{|D(i+j)|+1}$$
$$\vdots$$
$$\frac{1}{|D(i+j)|}\cdot p_{d_e[i,j]} + \frac{1}{|D(i+j)|}\cdot(1-p_{d_e[i,j]-1}) = \frac{1}{|D(i+j)|+1}.$$

Solving this system of equations, we get

$$p_{d_e[i,j]} = \frac{|D(i+j)| - d_e[i,j]}{|D(i+j)|+1}.$$

Similarly, in the compression stage, suppose we have the load uniformly distributed over a diagonal $D(i+j)$. The fractional load supported by each node is $\frac{1}{|D(i+j)|}$. The next diagonal has $|D(i+j)|-1$ nodes, and the fractional load we want to achieve is $\frac{1}{|D(i+j)|-1}$. This situation is depicted in Fig. 2.2. For nodes at distance $0 \le k = d_e[i,j] \le \frac{N}{2}$ from the boundary, the corresponding probability p_k satisfies

$$\begin{aligned} p_0 &= 0 \\ \frac{1}{|D(i+j)|}\cdot p_1 + \frac{1}{|D(i+j)|}\cdot(1-p_0) &= \frac{1}{|D(i+j)|-1} \\ &\cdots \\ \frac{1}{|D(i+j)|}\cdot p_{d_e[i,j]} + \frac{1}{|D(i+j)|}\cdot(1-p_{d_e[i,j]-1}) &= \frac{1}{|D(i+j)|-1}. \end{aligned}$$

Finally, solving this system of equations yields

$$p_{d_e[i,j]} = \frac{d_e[i,j]}{|D(i+j)|-1}.$$

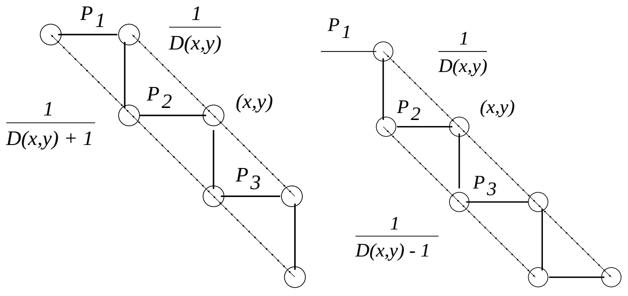


Figure 3: Forwarding probabilities (left: expansion, right: compression).

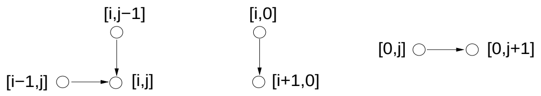
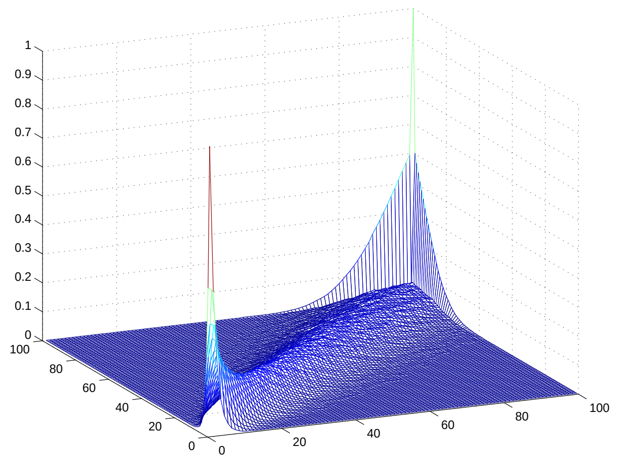


Figure 4: The three possible cases of coordinate formation. If a node v has two neighbors whose distance from the source is smaller than his own, and these nodes have lattice coordinates $[i_1,j_1]$ and $[i_2,j_2]$, then the coordinates of v are $[\max(i_1,i_2),\max(j_1,j_2)]$. If v has only one neighbor with smaller distance to the source, this neighbor must have coordinates of the form either $[i_1,0]$ or $[0,j_1]$, and then the coordinates of v are either $[i_1+1,0]$ or $[0,j_1+1]$. Finally, the decision of which node is $[1,0]$ and which node is $[0,1]$ is made arbitrarily by the source.

2.3 Distributed Computation of the Local Parameters

It is interesting to observe that, should the nodes be aware of their own lattice coordinates, then they could simply plug those coordinates into the definition of the optimal p's above. However, one of the assumptions we make is that nodes do *not* know their lattice coordinates: such coordinates do provide a fair amount of location information, information that is unreasonable to assume nodes like our micro-routers would have "for free". Instead, the most we can assume is that each node comes equipped with a unique identifier (e.g., burned in at fabrication time), and that position information is discovered via communication among the nodes. So our next goal is to give a distributed algorithm for computing lattice coordinates. And to do this, it is instructive to observe how in this particular grid coordinates are constrained to take values as illustrated in Fig. 4.

We see from Fig. 4 that all we need to recursively compute these coordinates is knowledge of distances between nodes. But this is easily accomplished by computing such distances using the distributed asynchronous version of the Bellman-Ford algorithm [3]—all that is required to perform this computation is knowledge of



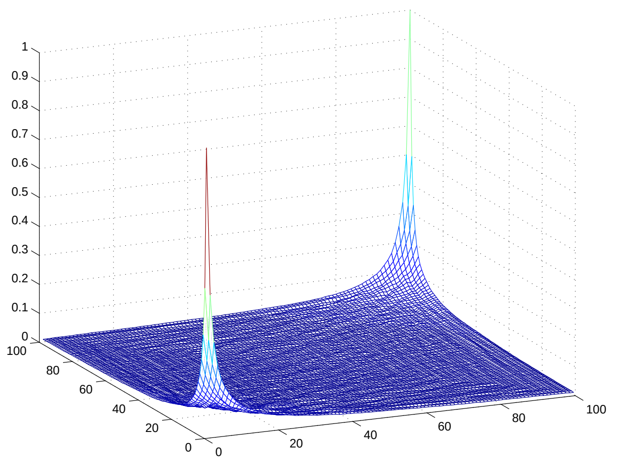


Figure 5: Load distribution. Top: a random walk based on tossing a fair coin among the two feasible neighbors; bottom: a random walk using the local parameters computed above. The simulation consists of 10000 messages transmitted in a network of size 100×100, injected at $[0, 0]$, exiting at $[99, 99]$. The axes in the bottom plane denote network positions (a node $[i, j]$ in the cubic grid is represented by a point (i, j) in this plane), and the vertical axis represents number of packets carried by node $[i, j]$ (normalized such that diagonals sum up to 1).

the source and destination unique identifiers (*not* their coordinates), and local message exchage, as permitted by our assumptions. So, once a node discovers its coordinates, routing is performed by applying eqns. (1,2).

2.4 Simulation Results

For illustration purposes, we compare the load distributions induced by the random walks here computed with the load distributions that would be induced by flipping a fair coin to decide which of the two feasible neighbors on a next hop to pick at each node. These plots are shown in Fig. 5.

If we think of the packets as being particles in a beam directed from the source to the destination, then we see that when using forwarding probabilities which are independent of the network location, the beam is narrowly confined around the main diagonal $i = j$, but as it moves closer to destination the beam becomes more spread out. Furthermore, since there is only one route from nodes of the form $[i, N-1]$ or $[N-1, j]$ to $[N-1, N-1]$, we see how this results in a grossly uneven load of these nodes. With the local parameters defined above, essentially what we do is make the beam wider in the expansion stage (by assigning higher probability to nodes away from the main diagonal), and narrower in the compression stage (by assigning higher probability to nodes close to the main diagonal).

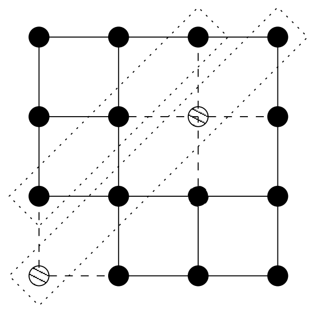


Figure 6: In this irregular topology (with two nodes deleted), if the load on the marked diagonal with three elements is uniform $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, then an uneven load of $(\frac{2}{3}, \frac{1}{3})$ will result for the marked diagonal with 4 elements (two of which are down); but to ensure an even load $(\frac{1}{2}, \frac{1}{2})$ on this second diagonal, an uneven load $(\frac{1}{4}, \frac{1}{4}, \frac{1}{2})$ is required on the first. Therefore, in this particular example, exactly uniform loads across all diagonals simultaneously are not feasible.

3. RANDOM WALKS ON IRREGULAR AND STATIC GRAPHS

3.1 Rationale: Regular Model plus Random Perturbations

The goal for this section is to define graphs which are less structured than the regular mesh considered in Section 2, and still define on these graphs random walks whose stationary distributions achieve the sought load balancing property. And we will do so by introducing random perturbations to the basic model for connectivity considered above: we delete a random subset of nodes from the regular grid. Note however that achieving exact load balancing as defined above (uniform distribution on diagonals) will not be possible in general—this is illustrated in Fig. 6.

So, if we cannot achieve uniform loads across diagonals, what can we achieve? In the context of irregular networks, what are we going to require of the random walks we construct? It turns out that we will still be able to define suitable random walks. This is because, with independent decisions made at each node of what is the next hop to follow, it seems clear that the higher the number of routes between any two nodes, the harder it becomes to predict which particular route a given packet will follow—however, if the number of routes is large, we can still invoke ergodic theorems and say something about the *distribution* of where a packet will lie after n hops. And therefore, by choosing appropriate local parameters for the random walk, we should still be able to control the shape of this distribution, and steer it to one which is, if not exactly, at least approximately uniform across diagonals. How to accomplish this, how to build on this intuition, is what we elaborate on next.

3.2 A Generalization of Lattice Coordinates

We introduce first a generalization of the concept of a lattice coordinate: now we label nodes with pairs of symbols, (s, d): s is the number of routes between the source and the labeled node, d is the number of routes between the labeled node and the destination. We say that two routes are different if they differ in at least one node.[2] And we observe that computation of these labels is again relatively straightforward using a distributed algorithm as discussed in the previous section: generalizing the well known result about combinatorial numbers that $C(n,k) = C(n-1,k) + C(n-1,k-1)$, we recursively compute the number of routes at a node as the sum of the numbers of routes at the two previous nodes.

The notions of expansion and compression stages admit natural generalizations to the case when the cubic grid has some missing nodes, and so do the forwarding probabilities based on these new labels. Consider a node with label (s,d). In the regular grid, this node has two neighbors to which it could forward data, with labels (s_1,d_1) and (s_2,d_2). Then, the probability p of forwarding a packet to the node (s_1,d_1) is defined as:

— Expansion (when $s \le d$):

$$p = \frac{s_2}{s_1+s_2} \tag{3}$$

— Compression (when $s \ge d$):

$$p = \frac{d_1}{d_1+d_2} \tag{4}$$

During the expansion stage, we make the forwarding probability proportional to the number of routes between the node and the source. This is because, if we were successful in spreading the load evenly in earlier stages, then we would expect the load received by any node to increase with the number of routes from the source to that node—more routes mean more ways in which a packet could reach this node. During the compression stage, we make the forwarding probability proportional to the number of routes between the node and the destination. Since nodes can distribute the incoming load between all the available routes toward the destination, we make the supported load proportional to this number of routes.

3.3 Equivalence with Lattice Coordinates in the Regular Grid

An important property of the labels defined above is that, if applied in the context of the regular cubic grid, the computed forwarding probabilities are identical—it is in this sense that we call these labels a generalization of the lattice coordinates. That is, eqns. (3,4) are the same as eqns. (1,2).

Consider the cubic grid shown in Fig. 1. The number of routes toward the source in the expansion stage represents an instance of Pascal's Triangle problem, and hence the number of routes is given by the combinatorial number

$$s_i = \begin{pmatrix} |D(i+j)|-1 \\ d_e[i,j] \end{pmatrix}, \tag{5}$$

as illustrated in Fig. 7.

To see that these new labels reduce to the standard lattice coordinates in the case of a complete grid, we combine eqns. (3) and (5), to obtain:

$$\begin{aligned} p &= \frac{s_2}{s_1+s_2} = \frac{\begin{pmatrix} |D(i+j)| \\ d_e[i,j]+1 \end{pmatrix}}{\begin{pmatrix} |D(i+j)| \\ d_e[i,j]+1 \end{pmatrix} + \begin{pmatrix} |D(i+j)| \\ d_e[i,j] \end{pmatrix}} \\ &= \frac{|D(i+j)| - d_e[i,j]}{|D(i+j)|+1} \end{aligned} \tag{6}$$

[2]Note that this is much weaker than requiring the routes to be *disjoint*, i.e., that all but the first and last nodes be different.

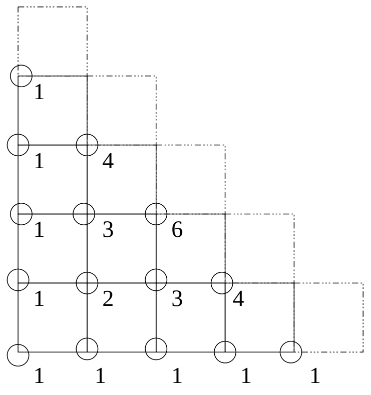


Figure 7: Pascal's Triangle. Each node is labeled with the number of routes to the bottom-left node.

Similarly, for nodes in the compression stage, we get

$$p = \frac{d_1}{d_1+d_2} = \frac{d_e[i,j]}{|D(i+j)|-1} \tag{7}$$

Therefore, labels based on the number of routes is indeed a meaningful generalization of labels based on lattice coordinates, in that the packet forwarding probabilities induced are the same.

3.4 An Algorithm for the Irregular Grid

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,

(b) only one of $[i+1,j]$ and $[i,j+1]$ are on,

(c) or neither $[i+1,j]$ nor $[i,j+1]$ are on.

In case (a), locally the network looks like the regular grid, and therefore we assign probabilities as in the regular case—note that the probabilities themselves are not identical though, since the number of routes available will depend on which nodes are ON and which are OFF. In case (b), we assign probability 1 to the active neighbor. In case (c), we assign probability 1 to a neighbor whose distance to destination is strictly smaller than the distance from the current node.

The basic idea of this algorithm is that we let constraint *(c.1)* (on the avoidance of livelock conditions) dictate our choice of next hop: we deal with perturbations to the basic connectivity model by assigning probability 1 to a neighbor arbitrarily chosen among those closer to destination. The rationale for this choice is that, if the process for deleting nodes is homogeneous (in the sense that the probability of having a node missing is independent of the location of the node, as is the case when nodes are deleted independently), then we expect that load imbalances created forwarding data to a single neighbor in some cases will cancel out. This issue is explored via simulations next.

3.5 Simulation Results

For illustration purposes, we repeat the same experiment performed in Fig. 5. The resulting plots are shown in Fig. 8.

It is interesting to see in Fig. 8 how the proposed algorithm does indeed achieve a marked improvement in terms of load balancing, especially in comparison to the scheme based on tossing a fair coin. Note also that now loads on diagonals are not uniform any more—although these plots suggest that the imbalance is not severe.

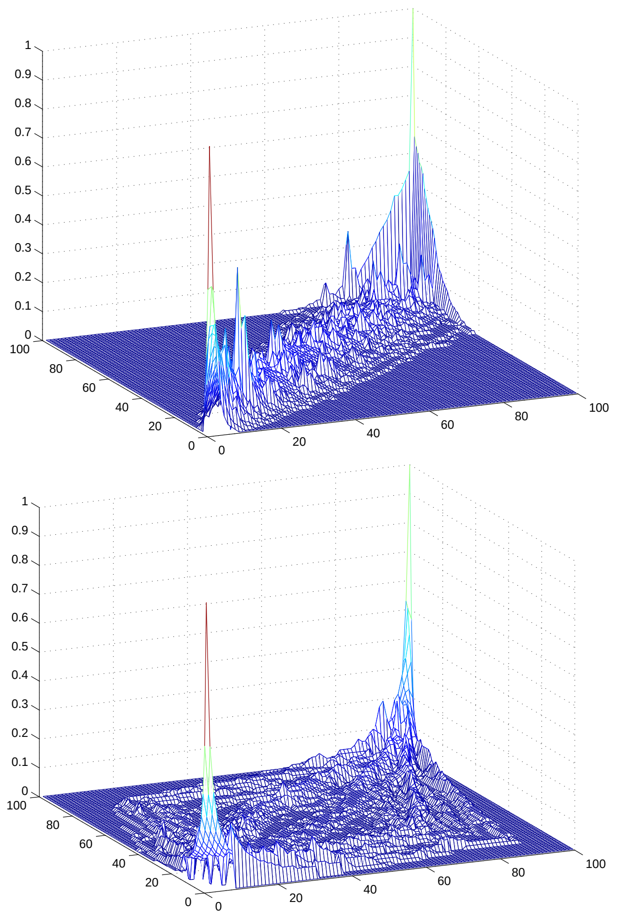


Figure 8: Load distribution. Top: a random walk based on tossing a fair coin among the two feasible neighbors; bottom: a random walk using the local parameters computed above. The simulation consists of 10000 messages transmitted in a network of size 100×100, injected at $[0, 0]$, exiting at $[99, 99]$. The axes in the bottom plane denote network positions (a node $[i, j]$ in the cubic grid is represented by a point (i, j) in this plane), and the vertical axis represents number of packets carried by node $[i, j]$ (normalized such that diagonals sum up to 1). In this simulation, each node is ON with probability 0.95, and OFF with probability 0.05.

4. RANDOM WALKS IN DYNAMIC GRAPHS

4.1 Rationale: Regular Model plus Dynamic Perturbations

We turn our attention finally to the problem that we were interested in right from the start: routing in random *dynamic* graphs. For this purpose, we consider next a time-varying version of the model considered in Section 3: instead of randomly deleting nodes from the cubic grid and leaving them deleted for all times, we take these nodes to switch between ON and OFF states over time, independently from one another, following a Markov rule.

4.2 Dynamic Labels

The mechanics of the labeling method remain almost unchanged from the case of an irregular but static network—the only difference is that when a node changes state, this change will affect the labels of its one-hop neighbors (since the number of routes available to these nodes will change), which in turn may trigger changes to labels of nodes farther apart. What we need to understand in this case is how routing performance is affected by the delays in propagating information about updates of labels, and how sensitive this routing performance is to inaccuracies in the labels. We explore both issues via simulations next.

4.3 Simulation Results

For illustration purposes, we repeat the same experiment performed in Figs. 5 and 8. The resulting plots are shown in Fig. 9.

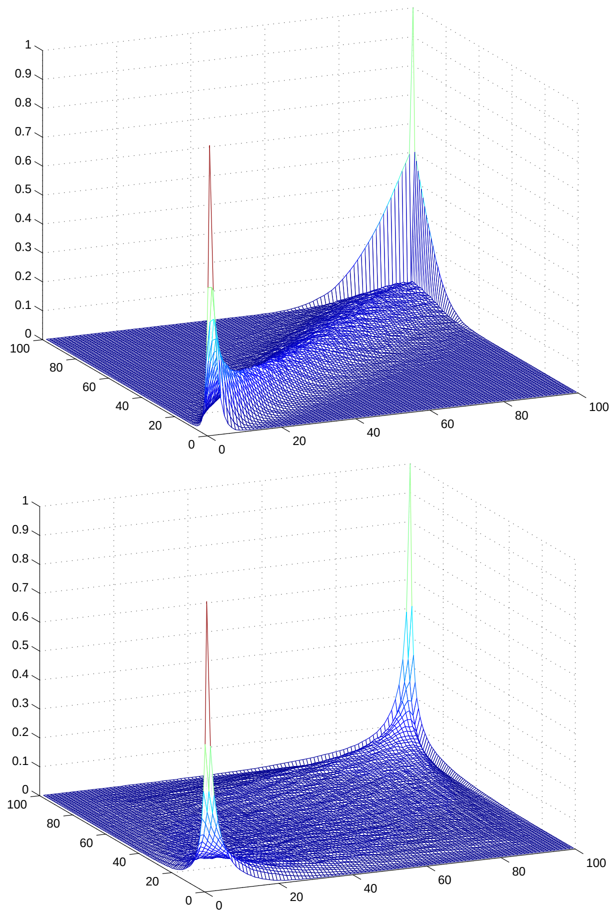


Figure 9: Load distribution. Top: a random walk based on tossing a fair coin among the two feasible neighbors; bottom: a random walk using the local parameters computed above. The simulation consists of 10000 messages transmitted in a network of size 100×100, injected at $[0, 0]$, exiting at $[99, 99]$. The axes in the bottom plane denote network positions (a node $[i, j]$ in the cubic grid is represented by a point (i, j) in this plane), and the vertical axis represents number of packets carried by node $[i, j]$ (normalized such that diagonals sum up to 1). In this simulation, the stationary probability of the OFF state is taken to be 0.05 (that is, in steady state 5% of the nodes are down), and the probability of an ON$\to$OFF transition is 0.025.

Interestingly enough, and at first surprising to us (although rather obviously with the benefit of hindsight), is the fact that the load distributions achieved in the context of a network with uncontrolled dynamics are much closer to uniform than those achieved in an irregular but static—i.e., more predictable—network. Intuitively what is happening in this case is that, because of the ergodicity of the model considered for network dynamics, the load distribution

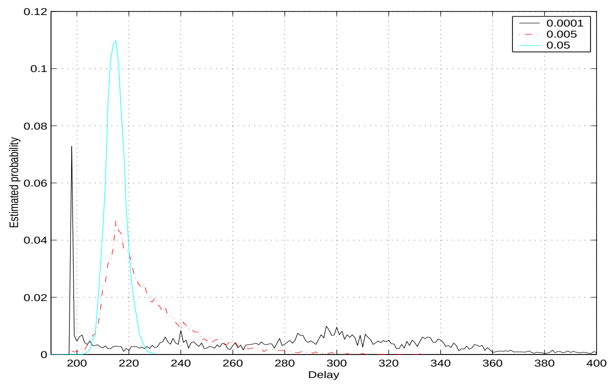


Figure 10: Transmission delay as a function of the variability of the network. **10000 messages are transmitted in a dynamic network of size 100×100. A network with 5% of the nodes in down in steady state, and three different chains with $\Pr(\text{ON} \to \text{OFF}) = 0.0001, 0.005, 0.05$ (the transitions from OFF to ON are adjusted so that in the stationary distribution, the OFF state occurs 5% of the time).**

of the dynamic network is essentially the average of the load distributions of many static networks—and it is this averaging effect what results in smoother, more balanced loads.

Besides load distributions, another important performance indicator is the *delay* distribution: how long does it take for a packet to go from source to destination? In the static case, this question admits a trivial answer: this is exactly the number of hops on a shortest route. But in the context of dynamic networks, this delay becomes random: as nodes go up and down, the information about state transitions needs to propagate throughout the network, and this propagation takes time. Therefore, inaccurate state information can introduce randomness in transport delay in two forms:

- Packets can get delayed at intermediate nodes. This could happen when both $[i+1, j]$ and $[i, j+1]$ are OFF, and the current distance estimates from both $[i-1, j]$ and $[i, j-1]$ to destination are greater than that from $[i, j]$. In this case, a packet at $[i, j]$ waits a random amount of time—until either the map of distances converges (and a new neighbor closer to destination can be identified), or until one of $[i+1, j]$ or $[i, j+1]$ turns ON again.

- Packets can get misrouted. This could happen when both $[i+1, j]$ and $[i, j+1]$ are OFF, and at least one of the current distance estimates from either $[i-1, j]$ or $[i, j-1]$ to destination is smaller than from $[i, j]$. This case cannot occur in the static case: with an accurate map of distances, a node $[i, j]$ satisfying these conditions would never be reached. However, in the dynamic case, this situation could come up for short periods of time, while updates to distance maps propagate.

Different delay distributions are shown in Fig. 10, corresponding to different "degrees of variability" of the network.

It is most interesting to observe in Fig. 10 how networks that are "more predictable" (i.e., in which state transitions are less frequent) induce delay distributions with higher variance than networks that appear to be in a state of flux (i.e., in which state transitions occur more often). Consider the case of $\Pr(\text{ON} \to \text{OFF}) = 0.0001$: in this case, about 7% of the packets make it to destination in the smallest possible number of hops—but conditioned on the delay being slightly higher than optimal, this delay is almost uniformly distributed over the a range that goes almost to twice the minimum time. Alternatively, in the most irregular network considered in these plots ($\Pr(\text{ON} \to \text{OFF}) = 0.05$), the delay distribution is sharply concentrated around a slightly suboptimal value.

We explain this apparent contradiction as follows:

- In a relatively stable network (low $\Pr(\text{ON} \to \text{OFF})$), state transitions are rare effects: in our network of size 100×100, $\Pr(\text{ON} \to \text{OFF}) = 0.0001$ means that on average only one node per time slot undergoes a state transition. If a packet never encounters a node with unaccurate information (i.e., that underwent a state transition and has not had enough time yet to update its local state information), this packet will likely arrive in the minimum number of hops.

- However, if a packet does encounter a node that recently underwent a state transition, it will likely get either delayed at that node, or misrouted, as explained above.

- Now, for how long will this condition persist? Recall the dynamics of our routing algorithm: try to route picking next hops based on the basic model for connectivity, and if none are available, pick a next hop based on distance maps. So, in a stable network, the condition for delaying packets is likely to persist for longer than in a network with nodes going up and down often: with nodes that seldom change state it is necessary to *wait* until the relevant information to update local distance maps arrives, whereas in the network that is in a "state of flux", the time it will take for a neighbor of the form $[i+1, j]$ or $[i, j+1]$ to switch back to ON is likely much smaller than the time it takes for distant updates to arrive.

We intend to make available on the web the simulator we have developed based on which the plots above were generated—this will happen by the time we submit a journal paper on this work.

5. CONCLUSIONS

5.1 Summary

In this work we presented our work on the design and performance analysis of routing algorithms for large scale wireless sensor networks. First, we argued that complexity considerations make it natural to introduce an element of randomization in the problem formulation, and so we formulated the problem as one of defining suitable random walks on random dynamic graphs. Then we presented random walk constructions in three different cases: a regular and static grid, an irregular but still static grid, and a dynamic grid. The basic approach to constructing these random walks consisted of first defining a simple basic model for connectivity in the network (the regular cubic grid), and then introducing random perturbations to the basic model—solve analytically for the optimal parameters in the basic model, take "greedy shortcuts" around the random perturbations. Properties of the resulting random walks were illustrated via simulations.

5.2 Future Work

There are two lines along which this work could proceed further. One consists of extending the basic model of connectivity considered in this work (the regular cubic grid) to more general percolation models, such as the random networks analyzed by Gupta and Kumar [10], Kleinberg's small world random graph models [16], etc. Although this is certainly a necessary step, we chose to start with the regular cubic grid for the simple reason that the main ideas we wanted to explore, in the case of the cubic grid, could be described using only very elementary mathematics—these models, although certainly much more interesting, require the use of more sophisticated analysis tools. So now that we have a good understanding about how to construct the sought random walks in a simple case, and about their properties, it does make sense to consider the more general (and more interesting) cases.

In the long term, we will study a number of problems on random graphs. One of the aspects we feel is part of the beauty of this work is the existence of a large body of related theory. We intend to:

- Explore connections between our routing problem and diffusion theory [27], since we expect the latter may hold the key to deriving analytical results on the behavior of our routing algorithms in asymptotically large networks.
- Generalize our construction of random walks to random graphs embedded in arbitrary k-dimensional manifolds (instead of the regular grid on a plane).
- Extend our construction to the case involving multiple sources and/or destinations.

5.3 Acknowledgements

The first author would like to thank Raissa D'Souza (for interesting discussions on issues related to this work), and the Institute of Pure and Applied Mathematics (IPAM) of the University of California, Los Angeles (for travel support to attend their program on Large Scale Communication Networks, `http://www.ipam.ucla.edu/programs/cn2002/`).

6. REFERENCES

[1] E. Ayanoglu, C.-L. I, R. D. Gitlin, and J. E. Mazo. Diversity Coding for Self-Healing and Fault-Tolerant Communication Networks. *IEEE Trans. Commun.*, COM-41(11):1677–1686, 1993.

[2] A. Banerjea. On the Use of Dispersity Routing for Fault Tolerant Realtime Channels. *European Trans. Telecommun.*, 8(4):393–407, 1997.

[3] D. Bertsekas and R. Gallager. *Data Networks (2nd ed)*. Prentice Hall, 1992.

[4] U. Black. *IP Routing Protocols (RIP, OSPF, BGP, PNNI & CISCO routing protocols)*. Prentice Hall, 2000.

[5] S. Chen and K. Nahrstedt. Distributed Quality of Service Routing in Ad-Hoc Networks. *IEEE J. Select. Areas Commun.*, 17(8):1488–1505, 1999.

[6] S.-N. Chiou. *Dispersity Routing and Reliability in a Communication Network*. PhD thesis, University of Southern California, 1988.

[7] T. M. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[8] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks. *ACM Mobile Computing and Communications Review*, 5(4):11–29, 2001.

[9] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. B. Wicker. An Empirical Study of Epidemic Algorithms in Large Scale Multihop Wireless Networks. UCLA Computer Science Technical Report UCLA/CSD-TR 02-0013. Submitted for publication.

[10] P. Gupta and P. R. Kumar. The Capacity of Wireless Networks. *IEEE Trans. Inform. Theory*, 46(2):388–404, 2000.

[11] B. Hajek. Minimum Mean Hitting Times of Brownian Motion with Constrained Drift. In *Proc. 27th Conf. Stoch. Proc. App.*, Cambridge, England, 2001.

[12] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In T. Imielinsky and H. Korth, editors, *Mobile Computing*. Kluwer Academic Publishers, 1996.

[13] E. C. Kan, Z. Liu, P. Wang, M. Kim, Y. N. Shen, and G. Pei. Si Fleas: Technology Demonstration of Functional Modules in Submillimeter Autonomous Microsystems. Invited talk, Ninth Foresight Conference on Molecular Nanotechnology, Santa Clara, CA, Nov. 9-11, 2001.

[14] F. P. Kelly. Network Routing. *Phil. Trans. R. Soc. Lond. A*, 337:343–363, 1991.

[15] J. Kleinberg. *Approximation Algorithms for Disjoint Paths Problems*. PhD thesis, Massachusetts Institute of Technology, 1996.

[16] J. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. Technical Report 99-1776, Dept. of Computer Science, Cornell University, October 1999.

[17] N. Maxemchuk. *Dispersity Routing in Store and Forward Networks*. PhD thesis, University of Pennsylvania, 1975.

[18] N. Maxemchuk and M. El Zarki. Routing and Flow Control in High-Speed Networks. *Proc. IEEE*, 78(1):204–221, 1990.

[19] J. M. McQuillan, I. Richer, and E. C. Rosen. The New Routing Algorithm for the ARPANET. *IEEE Trans. Commun.*, COM-28(5):711–719, 1980.

[20] M. Medard, S. G. Finn, R. A. Barry, and R. G. Gallager. Redundant Trees for Preplanned Recovery in Arbitrary Vertex-Redundant or Edge-Redundant Graphs. *IEEE/ACM Trans. Networking*, 7(5):641–652, 1999.

[21] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[22] A. Nasipuri and S. R. Das. On-Demand Multipath Routing for Mobile Ad-Hoc Networks. In *Proc. 8th Int. Conf. Comp. Comm. Networks (IC3N)*, Boston, MA, 1999.

[23] V. D. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proc. IEEE INFOCOM*, Kobe, Japan, 1997.

[24] M. R. Pearlman and Z. J. Haas. Determining the Optimal Configuration for the Zone Routing Protocol. *IEEE J. Select. Areas Commun.*, 17(8):1395–1414, 1999.

[25] C. E. Perkins and E. M. Royer. Ad-Hoc On-Demand Distance Vector Routing. In *Proc. 2nd IEEE Workshop Mobile Comp. Syst. Applic.*, New Orleans, LA, 1999.

[26] G. J. Pottie and W. Kaiser. Wireless Sensor Networks. *Comm. ACM*, 43(5):51–58, 2000.

[27] L. C. G. Rogers and D. Williams. *Diffusions, Markov Processes and Martingales—Volume I: Foundations*. Cambridge University Press, 2000.

[28] A. Scaglione and S. D. Servetto. On the Interdependence of Routing and Data Compression in Multi-Hop Sensor Networks. In *Proc. ACM MobiCom*, Atlanta, GA, 2002.

[29] I. Stoica, S. Shenker, and H. Zhang. Core-Stateless Fair Queueing: a Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks. In *Proc. ACM SIGCOMM*, 1998.

[30] S. H. Strogatz. Exploring Complex Networks. *Nature*, 410:268–276, 2001.

[31] D. J. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton University Press, 1999.