Reactive Routing Overhead in Networks with Unreliable Nodes

Nianjun Zhou
zhoun@rpi.edu

Huaming Wu
wuhm@rpi.edu

Alhussein A. Abouzeid
abouzeid@ecse.rpi.edu

Department of Electrical, Computer and Systems Engineering
Rensselaer Polytechnic Institute
110 Eighth Street
Troy, New York 12180, USA

ABSTRACT

This paper presents a new mathematical and simulative framework for quantifying the overhead of a broad class of reactive routing protocols, such as DSR and AODV, in wireless variable topology (ad-hoc) networks. We focus on situations where the nodes are stationary but unreliable, as is common in the case of sensor networks. We explicitly model the application-level traffic in terms of the statistical description of the number of hops between a source and a destination. The sensor network is modelled by an unreliable regular Manhattan (i.e. degree four) grid, and expressions for various components of the routing overhead are derived. Results are compared against *ns-2* simulations for regular and random topologies, which corroborate the essential characteristics of the analytical results. One of the key insights that can be drawn from the mathematical results of this paper is that it is possible to design *infinitely scalable* reactive routing protocols for variable topology networks by judicious engineering of the traffic patterns to satisfy the conditions presented in this paper.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing protocols*; I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling methodologies*

General Terms

Performance, Theory, Algorithms, Design

Keywords

Sensor networks, ad hoc networks, reactive routing, routing overhead, AODV, DSR, reliability

1. INTRODUCTION

Several classes of routing protocols for variable topology wireless (ad-hoc) networks have been proposed in the literature. In this paper, we focus on the class of *reactive* routing protocols such as AODV [1] and DSR [2], where paths are maintained only when needed. We focus on situations where topology changes because of node failure rather than node movement. Our objective is to mathematically characterize the scalability properties of these protocols under different traffic patterns. We primarily focus on *routing overhead* as a measure of scalability. To our knowledge, there exist no evaluations (whether analytical or simulative) of reactive routing overhead for *unreliable* networks, and hence this is a new contribution.

In this paper, we focus on the inter-dependence between the traffic pattern and the routing overhead, by deriving quantitative measures of the impact of the communication traffic patterns on the performance of routing protocols. One important application of this result is in the problem of sink-placement i.e. the problem of placing special nodes in the sensor grid for collecting information from "near-by" sensors, such that the average routing overhead is bounded below a desired value.

As a first analytic treatment of this problem, this paper uses a *Manhattan* grid model of a sensor network due to the simplicity of the topology. A Manhattan grid has a discrete and regular topology with fixed degree per node. We show, with the aid of simulations that the analytic results regarding the scalability of the network holds even for the case of random network topologies.

We view the two different models (regular vs. random) as abstractions of two different practical scenarios. Regular (Manhattan) grid is an abstraction of regular networks, where we can control or assign the location and wireless coverage of nodes. Random topology is an abstraction of irregular and random networks, where we cannot control and assign the locations of the nodes.

It is important to note that this work does not attempt to model or compare between specific reactive routing protocols - rather, to capture the essential behavior and scalability limits of this class of protocols by deriving lower bounds on the overhead. Extensive simulation-based comparisons between various reactive routing protocols can be found in the literature (e.g. [3]). The simulation results in this paper are not intended to provide an exact match with our analysis - rather, they are provided as a support of the conclusions regarding overhead scalability. The *ns-2* [4] simulations naturally deviate from the assumptions made in the analysis, and are intended to reflect that the results are indeed reasonable, even though we do not model the complex behaviors of MAC layer or route caching. Route caching helps reduce overheads but may introduce other side effects (e.g. [5])and MAC layer collisions may inadvertently reduce flooding overheads.

This paper is organized as follows. Section 2 presents the simplified models of the routing algorithms, traffic and topology. These are used in Section 3 for the mathematical analysis of routing overhead. Section 4 provides *ns-2* simulations of the Manhattan model. Section 5 introduces a model for generating random topologies and checks the validity of our analytically-derived conclusions by conducting *ns-2* simulations of random topologies. Section 6 concludes the paper highlighting the practical implications of this work and several possible venues for future extensions.

2. NETWORK MODEL

This section presents the description of a generic reactive routing protocol, and introduces the notation used throughout the rest of the paper.

Since we are interested in modeling a variable topology with node failures, we assume that nodes fail, or equivalently, turn to an "OFF" state, randomly. However, we assume that most nodes are "ON" most of the time. In other words, we are interested in modeling the overheads in path maintenance due to node failure, but we assume during our analysis that given a node failure, all other nodes have not failed. This necessitates this assumption. This is true if the probability that a node is OFF is much less than the probability that a node is ON. Examples of a practical scenario where this may arise is in the case where the nodes are powered using renewable sources of energy (e.g. the wind or the sun). Another example is when nodes are replaced soon after they fail, either manually or through the activation of stand-by nodes.

2.1 Traffic Model

We define a *new path request* or simply a *new session* as one that is associated by the arrival of a new application-level session request at a node i with some destination $j \neq i$ for which node i doesn't already have a path. If the destination is the same as an existing session, a "new path request" will not be generated, and hence we do not count this as a *new session.* New path requests are independent. Throughout the paper, we refer to a *routing-layer sessions* simply by *sessions* or *active paths* interchangeably. Also, we interchangeably use a routing *packet* and routing *message.*

Let $r_{i,j}$ denote the distance, in number of hops, between two nodes i and j. We assume that a node i will need to communicate and hence maintain a (routing-layer) session with node j with probability $p_{i,j}$ given by

$$p_{i,j} = \frac{c}{r_{i,j}^k} \tag{1}$$

where $0 < c < 1$. Because of the symmetry, it will be more convenient to drop the subscripts from the above equation when it is understood. We thus let $p(r)$ denote the probability that two nodes (i and j) will communicate i.e.

$$p(r) = \frac{c}{r^k} \tag{2}$$

where r is the distance between the node initiating the new path request and the destination node.

A discussion of the choice of the shape of the above distribution is necessary. Up to our knowledge, there are no similar empirically derived traffic statistics for wireless ad-hoc or sensor networks, and, in the lack of such statistics, the assumption is necessary. In the case of sensor networks, and because sensor networks are generally designed with a specific overall objective rather than providing connectivity between end-points [6], we argue that the traffic pattern is under the control of the designer. In either cases, we would like to understand the effect of the traffic pattern on the routing overheads, and hence the scalability, of reactive routing protocols.

There are several intuitive reasons for choosing such a traffic pattern. It seems quite intuitive that it is impossible to design very large scalable sensor networks if all nodes need to communicate with each other with equal probability, since the overheads will be huge (as our analysis shows below). In any case, our approach applies to any traffic pattern that is identical and independent for all nodes.

As a final note, we deliberately did not choose the form $\frac{c}{k^r}$ although this latter from may have proved easier to handle mathematically. The reason is that $\frac{c}{r^k}$ allows us to analyze the effects of r and k more closely than $\frac{c}{k^r}$ since the former decays polynomially in r while the latter decays exponentially in r.

2.2 Routing Protocols Model

We describe below a generic reactive routing protocol, which we believe captures the essential behavior of many designs and implementations of routing protocols including DSR and AODV relevant to our analysis. This analysis, and hence the generic protocol below, does not consider caching mechanisms. The effects of caching mechanisms will be observed later in the paper through simulations.

The overhead of reactive routing protocols can be associated with two operations: *route discovery* and *route maintenance*, both of which are described below.

2.2.1 Route Discovery

Route discovery is the mechanism initiated by a node i upon the arrival of a "new path request" (defined earlier) in order to discover a new path to a node j. Similar to other reactive routing protocols, our generic protocol uses a "flooding" technique for control packets. Node i floods the network with Route Request (RREQ) packets. The RREQ packet is initiated by some value called Time to Live (TTL) in the header of the packet. Each node forwards an RREQ packet only once, and decrements the TTL upon each transmission. If the initial TTL value is large enough, an RREQ packet arrives to j (we assume the network is connected). Since we don't use route cache, we assume that only the ultimate destination j can reply to a route request. Node j sends out a Route Reply packet (RREP) to i along the reverse path. Finally node i gets a shortest path source route to node j (in the case of source routing protocols like DSR) or entries in the routing tables are established at the nodes along a path between i and j (in the case of distance vector protocols like AODV).

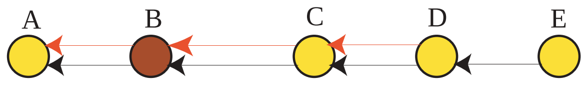

Figure 1: Node failure example. Two sessions (paths) exist, one from D to A and another from E to A. Node C detects that node B has failed and hence notifies D and E by sending RERR packets.

2.2.2 *Route Maintenance*

Route maintenance is the mechanism by which a node i is notified that a link along an active path has broke such that it can no longer reach the destination node j through that route. Upon reception of a notification of route failure, node i can initiate route discovery again to find a new route for the remaining packets destined to j.

If an intermediate node C finds the link to its next-hop destination B broken (by some MAC layer method), C initiates a Route Error (RERR) message. We describe two route notification mechanisms, one mimics source-route routing while the other mimics distance vector routing:

(a) In *source-route notification*, C sends an RERR message to *each source node* that has sent a packet routed through the failed link. Each RERR message will travel along *the reverse route* from the node reporting link breakage to the source node. Thus, if any link on a source route is broken, the original source node is notified by an RERR packet. For example, as in Figure 1, there are two source routes. One from node E to node A, another from node D to node A. Suppose now node B fails, node C discovers a broken link and sends two source RERR packets since two routes pass through the link C-B. One RERR packet is from node C to node D, another RERR packet is from node C through D then to node E. In this scenario, node D receives an RERR packet twice, although each RERR reports the same broken link. After receiving the RERR, a new route discovery process must be initiated by the source, if this route is still needed.

(b) In *distance-vector notification*, "C" also sends an RERR message. However, in distance-vector routing, node C doesn't know the source of the packets. Thus the RERR will be forward to the active "next hop" entries in the distance vector routing table. The difference between this and DSR can be indicated in reference to Figure 1, where in this case, node D receives RERR packet only once. In essence, RERR packets follow a spanning tree rooted at the node detecting the failure.

2.3 Notations and Definitions

For the convenience of discussion, we first present the different quantities of interest and their notations. As mentioned earlier, we will omit the subscript identifying nodes due to the symmetry of the network model.

- $N(r)$ denotes the number of nodes located within a distance r from a given node.
- $p_R(r)$ denotes the probability that a pair of communicating nodes is a distance r apart (i.e. this is a conditional probability).
- For a given small positive value $0 < \alpha < 1$, $T(\alpha)$ denotes the time-to-live value necessary for guaranteeing that the packet will reach the destination with a probability at least $1 - \alpha$.
- Z is a random variable that denotes the number of hops that an active session travels. $E[Z]$ denotes the expected value of Z.
- $E[S_i]$ denotes the average number of (routing-layer) sessions (i.e. paths) initiated from a given node.
- $E[S_t]$ denotes the average number of active paths terminating at a given node.
- $E[S_r]$ denotes the average number of sessions passing through a node. Here a pass-through session (relay session) is one that neither terminates nor starts at the given node.
- $E[S]$ denotes the average number of active paths at a given node (i.e. sum of the above three quantities).
- q denotes the probability that a routing packet at a given node is not terminating at the node. In other words, the probability that the session does not terminate at the given node.
- N_{find} denotes the total number of RREQ packets per new route discovery. Every transmission of the same packet is counted as a separate transmission. $E[N_{find}]$ denotes the expected number.
- N_{off} denotes the total number of packets (cost) needed to notify others about a node failure. $E[N_{off}]$ denotes the expected number.
- For a given node, the probability distribution of the number of different active paths terminated at (or initiated from) the node is $p_S(s)$.

3. ANALYSIS FOR MANHATTAN GRID

As we pointed above, we start our research by selecting Manhattan grid topology because of its simplicity and its being as an abstraction of regular networks. Besides, results obtained from approximating the topology of sensor networks by a regular grid will be shown to hold for random network topologies later in this paper.

3.1 Topology Model

Our analysis models the network by a grid (Figure 2). Nodes are located at the intersections of a Manhattan grid. We assume that nodes use an ideal (i.e. no collisions) wireless channel for communication. Each node has a fixed transmission range. Two nodes within the range of each other can communicate directly and are said to be neighbors. Communication links are assumed to be bidirectional. Also, we limit the transmission range such that a node can communicate directly to its immediate neighbors only.

Let (x_i, y_i) denote the coordinates of a node i. We define the distance $r_{i,j}$ between two nodes i and j

$$r_{i,j} = |x_i - x_j| + |y_i - y_j| \tag{3}$$

As we control transmission range to limit the direct communication to the immediate nodes, the distance between

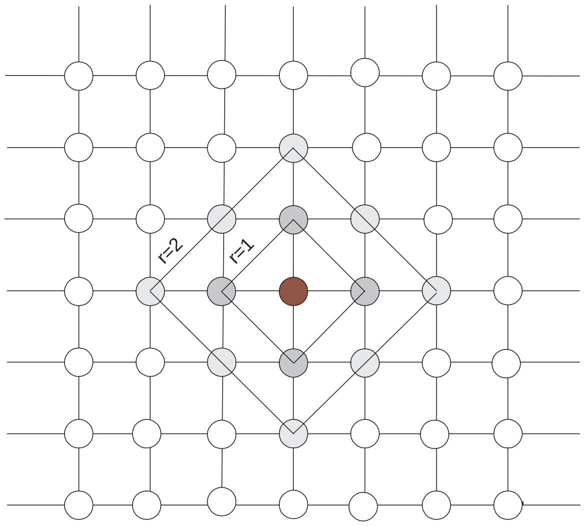

Figure 2: A sketch of a Manhattan grid. The darkest node is the node located at origin point (0,0). Nodes linked together have the same distance to the node at the origin.

two nodes is exactly the number of "hops" along the shortest path between them.

Finally, we will refer frequently to the following summation:[1]

$$f(k) = \sum_{r=1}^{\infty} \frac{1}{r^k} \tag{4}$$

3.2 Some Quantities of Interest

Claim 3.1. *The number of nodes located at a distance r away from a node is*

$$D(r) = 4r \tag{5}$$

Proof. Let O denote the interesting node. Without loss of generality, we assume node O is located at $(0, 0)$. For any given positive integer r, there are a total of $(r+1)$ nodes at a distance r from node O located in each of the four quadrants (Figure 2). Summing over the four quadrants yields $4(r+1)$. Removing the four double counted nodes located at x and y axis, the total number of nodes is $4r$. $\square$

From the above result, we can easily deduce the following result.

Claim 3.2. *The total number of nodes which are with in a distance r from a given node is*

$$N(r) = 2(r^2 + r) \tag{6}$$

Theorem 3.1. *The probability $p_R(r)$ that a pair of communicating nodes is a distance r apart is equal to*

$$p_R(r) = \frac{\frac{1}{r^{k-1}}}{\sum\limits_{r=1}^{\infty} \frac{1}{r^{k-1}}} = \frac{\frac{1}{r^{k-1}}}{f(k-1)}; k > 2 \tag{7}$$

[1]This infinite sum is a famous series known as R-series. The sum of the infinite series has general closed form result when k is an even integer.

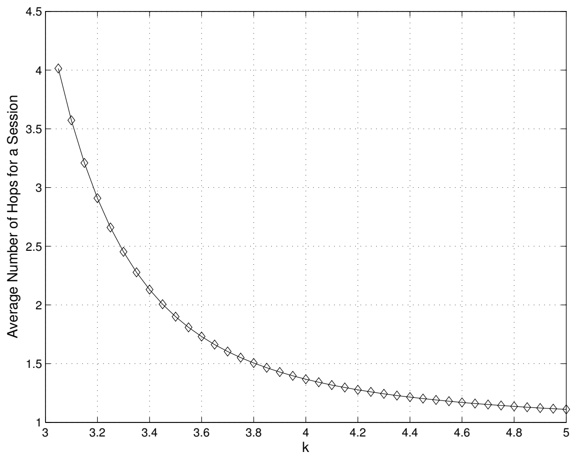

Figure 3: Average number of hops that an active session travels.

Proof. Let O denote the source node. Let r denote a distance from the source node to destination node. There are $D(r)$ nodes at a distance r away from node O. Each node is communicating with node O with a probability $p_R(r)$. Hence,

$$p_R(r) \propto (4r)p(r) = \frac{4c}{r^{(k-1)}} \tag{8}$$

By normalizing the above equation, we prove the theorem. The probability distribution is meaningful only if we have $k > 2$, otherwise it becomes zero. $\square$

It is interesting to know the average number of hops that an active session travels.

Corollary 3.2. *The average number of hops that an active session travels is equal to*

$$E[Z] = \frac{f(k-2)}{f(k-1)}; k > 3 \tag{9}$$

We notice that E[Z] is a function of k (Figure 3).

3.3 Time To Live $T(\alpha)$

Claim 3.3. *For a given small positive value $0 < \alpha < 1$,*

$$T(\alpha) = \min\{i| \sum_{r=i+1}^{\infty} \frac{1}{r^{k-1}} \leq \alpha f(k-1)\} \tag{10}$$

Proof. From Theorem 3.1, the probability that the destination node is located at a distance larger than a positive integer i can be calculated from (7) as

$$\frac{\sum\limits_{r=i+1}^{\infty} \frac{1}{r^{k-1}}}{\sum\limits_{r=1}^{\infty} \frac{1}{r^{k-1}}} = \frac{\sum\limits_{r=i+1}^{\infty} \frac{1}{r^{k-1}}}{f(k-1)} \tag{11}$$

which proves the claim. $\square$

3.4 Average Number of Sessions

THEOREM 3.3. *The average number of active paths terminating at a node is*

$$E[S_t] = 4c\sum_{r=1}^{\infty}\frac{1}{r^{(k-1)}} = 4c * f(k-1); k > 2 \tag{12}$$

PROOF. Let o denote the node located at $(0,0)$. Let M denote the set of all the nodes in the network, and M/o denote all the nodes except the node o. Without loss of generality, choose node o as the destination node. Let S denote the number of active paths terminating at node o. Let $X_{i,o}$ be an indicator function which is 1 if there exists an active paths starting at node i and terminating at node o, and is 0 otherwise. Let $p_{i,o}$ denote the probability that node o and i have an active session. Clearly, $Pr[X_{i,o} = 1] = p_{i,o}$. Hence,

$$S_t = \sum_{i\in M/o} X_{i,o} \tag{13}$$

Taking the expectation of both sides yields

$$E[S_t] = \sum_{i\in M/o} E[X_{i,o}] \tag{14}$$

Using the result of Claim 3.1,

$$\begin{aligned} E[S_t] &= \sum_{r=1}^{\infty} 4r\tfrac{c}{r^k} \\ &= 4c\sum_{r=1}^{\infty}\tfrac{1}{r^{(k-1)}} = 4c * f(k-1); k > 2 \end{aligned} \tag{15}$$

$k > 2$ is necessary for the convergence of the above equation. □

COROLLARY 3.4.

$$E[S_i] = E[S_t] = 4c * f(k-1); k > 2 \tag{16}$$

PROOF. It is obvious that each session has to have a source node as well as a destination node. □

LEMMA 3.5.

$$E[S_r] = 4c * (f(k-2) - f(k-1)); k > 3 \tag{17}$$

$$E[S] = 4c * (f(k-2) + f(k-1)); k > 3 \tag{18}$$

PROOF. Let i denote any node in the network, and let r denote the distance from the source node to destination node of a session initiated by node i. This session will pass through $(r-1)$ intermediate nodes. There are total $4r$ nodes having distance r away from node i. The average number of active pass-through sessions observed by other nodes is

$$\begin{aligned} &\sum_{r=1}^{\infty}(r-1)*4r*p(r) = \sum_{r=1}^{\infty}(r-1)*4r*\tfrac{c}{r^k} \\ &= 4c*\sum_{r=1}^{\infty}\left(\tfrac{1}{r^{k-2}} - \tfrac{1}{r^{k-1}}\right) \\ &= 4c*(f(k-2) - f(k-1)) \end{aligned} \tag{19}$$

From symmetry, the average number of active relay sessions for a given node i has to be equal to the average number of active pass-through sessions observed by other nodes but initiated by node i. The right side of the above equation gives the average number of relay sessions for a node.

The average of the total number of sessions at a node is the sum of the averages of pass-through sessions, initiated sessions and terminated sessions at the node. □

The above results indicate that, indeed, the number of sessions passing through a node is larger than the number of sessions generated, on the average. The reason is that some sessions are observed by many intermediate nodes along its communication path.

THEOREM 3.6.

$$q = 1 - \frac{2f(k-1)}{f(k-2)+f(k-1)}; k > 3 \tag{20}$$

PROOF. The above result follows directly from Lemma 3.5. We have

$$\begin{aligned} q &= \tfrac{E[S_r]}{E[S]} \\ &= \tfrac{f(k-2)-f(k-1)}{f(k-2)+f(k-1)} \\ &= 1 - \tfrac{2f(k-1)}{f(k-2)+f(k-1)} \end{aligned} \tag{21}$$

□

For the case of $k \le 3$, we have $q = 1$. On the average, there are much more pass-through (relay) sessions for a node than the number of sessions terminated at or started from the node.

THEOREM 3.7. *The probability distribution of the communication session $p_S(s)$ for a node can be found recursively. If we denote $g_s(n) = \frac{p_s(n)}{p_s(0)}$ for $n > 0$ and assign $g_s(0) = 1$, we have*

$$g_s(n) = g_s(n-1)g_1(1) + \sum_{i=2}^{n}(-1)^{i-1}G_i g_s(n-i) \tag{22}$$

for $n \ge 2$, and

$$p_s(0) = \prod_{a\in M/o}(1 - p_{a,o}) \tag{23}$$

$$G_i = \sum_{a\in M/o}\left[\frac{p_{a,o}}{1-p_{a,o}}\right]^i \tag{24}$$

and

$$p_s(1) = p_s(0)G_1 \tag{25}$$

PROOF. See the Appendix. □

3.5 Route Discovery Overhead

3.5.1 *Route discovery overhead as a function of* $T(\alpha)$

COROLLARY 3.8. *The total number of route request (RREQ) packets associated with a flooding event with TTL value $T(\alpha)$ is*

$$N_{find} = 1 + 2 * T(\alpha) * (T(\alpha) - 1) \tag{26}$$

PROOF. Assume that the route discovery is initiated by a node i. The longest distance each packet travels is bounded by T(α). Let $r_{i,j}$ denote the distance between a node j and node i. Node j with distance $r_{i,j} < T(\alpha)$ will transmit the packet once. Therefore, the total packet retransmissions is the same as the number of the nodes within a circle of radius $T(\alpha)$ from node i. From Claim (3.1),

$$\begin{aligned} N_{find} &= 1 + \sum_{l=1}^{T(\alpha)-1} 4*l \\ &= 1 + 4*\tfrac{T(\alpha)(T(\alpha)-1)}{2} \\ &= 1 + 2*T(\alpha)*(T(\alpha)-1) \end{aligned} \tag{27}$$

□

3.5.2 Lower bound on route discovery overhead

Next we will analyze the average number of routing packets to find a destination node if the source node does not know the location of the destination. We assume that the optimal routing (lower bound) is to inform a node only once, which is implemented by using routing path as clock-wise cyclic route from the lower value distance r to $(r+1)$ after routing through all the nodes with distance r.

Claim 3.4. *The minimum number of routing packets to find the communicating destination located at distance r is*

$$N_{find}(r) = 2r^2 + 0.5 \tag{28}$$

Proof. The overhead of discovery can be separated into two parts: (1) the overhead of looping through the nodes with distance less than r, denoted as $N^1_{find}(r)$, and (2) the overhead for looping through all the nodes with distance r until finding the destination node as $N^2_{find}(r)$.

For $N^1_{find}(r)$ we have,

$$\begin{aligned} N^1_{find} &= \left[\sum_{i=1}^{(r-1)} 4i\right] \\ &= 4\left[\tfrac{r*(r-1)}{2}\right] \\ &= 2r(r-1) = 2r^2 - 2r \end{aligned} \tag{29}$$

For $N^2_{find}(r)$, we have

$$\begin{aligned} N^2_{find} &= \left[\tfrac{1}{4r}\sum_{i=1}^{4r} i\right] = \tfrac{1}{4r}\left[\tfrac{4r*(4r+1)}{2}\right] \\ &= \tfrac{4r+1}{2} = 2r + 0.5 \end{aligned} \tag{30}$$

By adding $N^1_{find}(r)$ and $N^2_{find}(r)$, we prove the claim. □

Theorem 3.9. *The average minimum overhead of finding a new route is*

$$E[N_{find}] = \frac{2f(k-3)}{f(k-1)} + 0.5; k > 4 \tag{31}$$

Proof. For an active session, the distance of source node to destination node is a random variable. The probability distribution of the random variable is given by (7). Using this distribution, the average of N_{find} is

$$\begin{aligned} E[N_{find}] &= \sum_{r=1}^{\infty}(2r^2+0.5)\frac{\frac{1}{r^{(k-1)}}}{f(k-1)} \\ &= 2\sum_{r=1}^{\infty}\frac{\frac{1}{r^{(k-3)}}}{f(k-1)} + 0.5\sum_{r=1}^{\infty}\frac{\frac{1}{r^{(k-1)}}}{f(k-1)} \\ &= \tfrac{2f(k-3)}{f(k-1)} + 0.5 \end{aligned} \tag{32}$$

□

3.6 Shortest Path Analysis

As we mentioned earlier, any node can be turned "ON" and "OFF" randomly. Our routing protocols assume that once a given node turns "OFF", all nodes communicating to/through that node needs to be notified through a route notification procedure. We will compute the minimum overhead required to complete a route notification procedure, on the average. The best reactive routing protocol can do no better than a hypothetical routing protocol that somehow notifies the intended nodes by communicating over the shortest possible paths between the failed node and the intended nodes. Further, it needs only to notify those affected nodes, not all the nodes in the network.

We will therefore need to derive the following two lemmas which provide expressions for some interesting quantities regarding the properties of the shortest paths. Those will be used in the following section to derive the minimum average route failure notification overhead.

Let (x_i, y_i) denote the coordinates of a node i. Define

$$\triangle x_{i,j} \triangleq |x_i - x_j| \tag{33}$$

$$\triangle y_{i,j} \triangleq |y_i - y_j| \tag{34}$$

then

$$r_{i,j} = \triangle x_{i,j} + \triangle y_{i,j} \tag{35}$$

Lemma 3.10. *Let $L_{i,j}$ denote the total number of shortest paths between two nodes i and j. Consider the case where $x_i < x_j$ and $y_i < y_j$ (Figure 4). The shortest path between the two nodes will consist of exactly $\triangle x_{i,j}$ horizontal hops to the right and $\triangle y_{i,j}$ upward vertical hops on the grid, in any order. Thus, $L_{i,j}$ is the total number of permutations of hops of the two types. It follows from standard counting methods that*

$$L_{i,j} = \left(\begin{array}{c} r_{i,j} \\ \triangle x_{i,j} \end{array}\right) = \frac{(r_{i,j})!}{(\triangle x_{i,j})!(\triangle y_{i,j})!} \tag{36}$$

Lemma 3.11. *Let $L_{i,j,l}$ denote the number of shortest paths between nodes i and j that pass through node l. For any two arbitrary nodes i and j with $x_i < x_j$ and $y_i < y_j$, we need only to consider the nodes that satisfy $x_i \le x_l \le x_j$ and $y_i \le y_l \le y_j$ since any other node l that does not satisfy this condition cannot possibly lie on any of the shortest paths between nodes i and j. Using the same counting argument as before, the first portion of the paths going through node l must be exactly $\triangle x_{i,l}$ horizontal-right hops and $\triangle y_{i,l}$ vertical-up hops, in any order. For each such choice, there are a number of ways to continue the path to the destination node j, but any of those alternatives must be formed of $\triangle x_{l,j}$ horizontal-right and $\triangle y_{l,j}$ vertical-up hops, in any order. Thus,*

$$L_{i,j,l} = \left(\begin{array}{c} r_{i,l} \\ \triangle x_{i,l} \end{array}\right) \times \left(\begin{array}{c} r_{l,j} \\ \triangle x_{l,j} \end{array}\right) \tag{37}$$

Let $a_{i,j,l}$ denote the probability that node l lies along the shortest route between nodes i and j, as shown in Figure 4. Assume that when nodes i and j need to communicate, any of the shortest paths between these two nodes is discovered (through a RREQ/RREP procedure) with equal [2] probability. Then

$$a_{i,j,l} = \frac{L_{i,j,l}}{L_{i,j}} \tag{38}$$

An example is shown in Figure 4.

Let $p_{i,j}$ denote the probability that there is an active session started from node i and terminated at j. Let $b_{i,j,l}$ denote the probability that node l is routing the communication session from node i to node j. Then

$$b_{i,j,l} = a_{i,j,l}p_{i,j} = p_{i,j}\frac{L_{i,j,l}}{L_{i,j}} \tag{39}$$

[2]This is a reasonable assumption for flooding-based routing protocols.

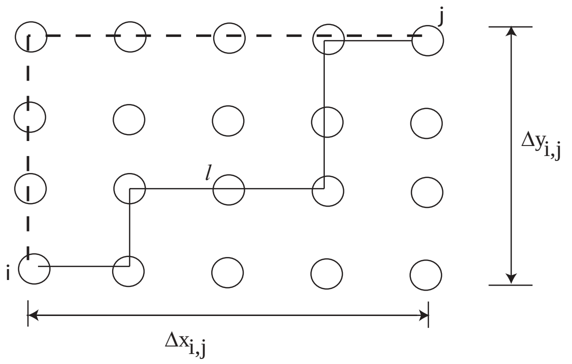

Figure 4: A portion of the Manhattan grid. Two possible shortest paths between nodes i and j are shown, where the dashed path does not go through node l. For this example, $r_{i,j} = 7$, $\triangle x_{i,j} = 4$, $\triangle x_{i,l} = 2$, $\triangle x_{l,j} = 2$. By substituting in (36) and (37), $L_{i,j} = 35$ and $L_{i,j,l} = 18$, which can be verified by enumeration. Hence from (38), $a_{i,j,l} = 0.51$.

3.7 Route Notification Overhead

We presented in Section 2.2 two methods for route notification; source route-based and distance vector-based. In the former, separate messages are sent for each intended node, while in the latter, the same message may be sent along the shared portion of overlapping routes to more than one destination.

Another important feature of distance vector route maintenance is the local-repair feature (e.g. see [1]). If an intermediate node fails, neighboring nodes may attempt to repair (or patch) the path instead of notifying the transmitting. Clearly, this works only for the paths that are not terminating at the intended node.

Thus, we split our analysis to three cases; two extreme cases, and an intermediate case, as follows. The first case uses local repair together with distance vector-based notification. We make the optimistic assumption that local repair will succeed for all active paths except for those terminating at the failed node. The third case is for source route based notification where there is no local repair and all affected nodes (whether having active pass-through paths through the failed node or active paths terminating at the failed node) must be notified, each with an independent RERR packet. The second, intermediate, case uses local repair but with source route notification (i.e. separate messages). We will show that in some cases, the first and second cases are equivalent.

3.7.1 Case 1: Local Repair with Distance-vector based notifications

Let o denote the node located at $(0, 0)$. Let M denote the set of all nodes in the network, and M/o denote all nodes except node o. Without loss of generality, choose node o as the destination node. Node j is another node in M/o. Let $c_{j,o}$ denote the probability that there is an active session with node o started from j and at the same time node j is not a routing node for other active sessions having node o as destination node. Let M_j denote the set of nodes that can have node j as its routing node. From any node $k \in M_j$, there is a path passing through node j that is one of the possible shortest paths from node k to node o. For example, for a node j located at (2,3), M_j will be all the nodes located at $\{(x, y)|x \geq 2, y \geq 3\}$ except $(2, 3)$. Note that the event "there is an active session with node o started from j" is independent from the event "node j is not a routing node for other active sessions having node o as destination node."

$$\begin{aligned} c_{j,o} &= p_{j,o} \left[\prod_{l \in M_j} (1 - b_{l,o,j}) \right] \\ &= p_{j,o} \left[\prod_{l \in M_j} (1 - p_{l,o} a_{l,o,j}) \right] \\ &= p_{j,o} \left[\prod_{l \in M_j} \left(1 - p_{l,o} \frac{L_{l,o,j}}{L_{l,o}} \right) \right] \end{aligned} \tag{40}$$

THEOREM 3.12.

$$E(N_{off}) = \left[\sum_{j \in M/o} (r_{j,o} - 1)\, p_{j,o} \left(\prod_{l \in M_j} (1 - p_{l,o} \frac{L_{l,o,j}}{L_{l,o}} \right) \right] \tag{41}$$

PROOF. Define a random variable $X_{j,o}$ for a give node j as

$$X_{j,o} = \begin{cases} r_{j,o} - 1, & ;\text{if } A \\ 0, & ;\text{otherwise} \end{cases} \tag{42}$$

where A is the event "session starts at j and ends at o, and also j is not a routing node for other active sessions that end at o." Hence,

$$N_{off} = \sum_{j \in M/o} X_{j,o} \tag{43}$$

and taking the expectation of both sides,

$$E[N_{off}] = \sum_{j \in M/o} E[X_{j,o}] \tag{44}$$

and substituting by

$$E[X_{j,o}] = (r_{j,o} - 1) c_{j,o} \tag{45}$$

completes the proof. □

3.7.2 Case 2: Local Repair with Source Route Based Notification

In this case, a RERR packet will be sent to the source of each active session that ends at the failed node, independently (i.e. even if some of the sessions might have some portion of the routes are common). Using the same methodology presented in Theorem 3.3, we have

THEOREM 3.13.

$$E[N_{off}] = 4c(f(k-2) - f(k-1)) \tag{46}$$

3.7.3 Relation between route notification in Cases 1 and 2

Consider a special case where a network in which the needs of communication are "weak" i.e the coefficient c in (2) is very small. If we only preserve the first-order involving c, then

$$c_{i,j} = p_{i,j} = \frac{c}{r_{i,j}^k}; k > 3 \tag{47}$$

and hence

COROLLARY 3.14. *For a network which is low traffic, i.e $c << 1$, route failure notification overhead for both cases 1 and 2 above are identical. Specifically, (41) reduces to (46)*

PROOF. Without loss of generality, we assume a node O is turned off (or failed) at location at $(0,0)$. Because the communication needs are weak, we can ignore the terms of c^n in (41) with $n > 1$. Hence,

$$E[N_{off}] = \sum_{r=1}^{\infty} (r-1) * 4r * \frac{c}{r^k} = 4c(f(k-2) - f(k-1))$$

□

3.7.4 Case 3: No local repair and source route based notification

In this case, a RREP packet will be sent to each source that has an active session passing through or terminating at the failed node.

THEOREM 3.15.

$$E[N_{off}] = 2c(f(k-3) - f(k-2)); k > 4 \qquad (48)$$

PROOF. Observed from any given node m, whether a session passes through or terminates at the node m can be uniquely presented as a tuple (j, m, i). Here, j is the session source node, i is the session destination node. If a node m is turned off, the total number of RERR packets that will be transmitted to node j to notify it about the failure of the node m is $(r_{m,j} - 1)$.

For any given node i and a given session that ends at node i with session length (distance from source node to destination node) r will generate $(r-1)$ tuples described above. Each tuple specifies a pass-through or terminate node m for this session. From the symmetric construction of the network, for a given node m and a given distance r and a given positive integer $l < r$, the average number of sessions with session length r and distance l from pass-through node m to source node is $\frac{4*r*c}{r^k}$. If node m fails, the number of packets to notify node j is $(l-1)$. Finally, we have the following result.

$$\begin{aligned} E[N_{off}] &= \sum_{r=1}^{\infty} \left(\frac{4c}{r^{k-1}}\right) \left(\sum_{l=1}^{r} (l-1)\right) \\ &= \sum_{r=1}^{\infty} \left(\frac{4c}{r^{k-1}}\right) \left(\frac{r(r-1)}{2}\right) \\ &= 2c \sum_{r=1}^{\infty} (r^2 - r) \frac{1}{r^{k-1}} \\ &= 2c(f(k-3) - f(k-2)) \qquad (49) \end{aligned}$$

□

3.8 Scalability of the Manhattan grid

From the previous results, the average total number of active sessions for a node and the average number of packets to notify about the failure of a node are bounded for an infinite Manhattan grid if the coefficient k is larger than 3, and 4. We have the following result.

COROLLARY 3.16. *The Manhattan grid becomes infinitely scalable only if the coefficient k is larger than 4.*

This result, among others, is validated in the following section.

4. NUMERICAL RESULTS AND SIMULATIONS FOR MANHATTAN GRID

To verify our analysis, we use the *ns-2* [4] simulation tool to run a number of simulations described in this section. Since it is impossible to simulate an infinite grid, an issue arises in how to compare the analytical results with simulations. As mentioned in Section 2, our analysis applies to a finite but symmetric grid (i.e. a torus) by replacing the upper limit in each summation in the results by the proper value that depends on the size of the finite grid used. A square grid of size N was used in generating the simulation results. Hence, for generating the numerical results from the analysis, the upper limit used is $\sqrt{N}$ instead of ∞. However, this doesn't take care of the edge effects that will arise in the simulations, which causes deviations from the analytical results, especially for small k.

4.1 Simulation Set-up

We consider five networks of size 49, 121, 225, 361 and 529[3]. Every node is placed at the intersection of a square Manhattan grid. Here, nodes have fixed positions without any movement for the entire simulation. Field sizes are different for different network sizes; 1400m × 1400m field has 49 nodes; 2200m × 2200m for 121 nodes; 3000m × 3000m for 225 nodes; 3800m × 3800m for 361 nodes; 4600m × 4600m for 529 nodes. We use all the simulation components that *ns-2* provides to set up network communication, such as Channel model, Propagation model, Network Interfaces, MAC 802.11, Priority Queue, and Omni Antenna [4].

As the goal of our simulation is to verify our analytical results, the transport protocol is irrelevant, so we use the most simple one to speed up the simulation run time. We choose our traffic sources to be Constant Bit Rate (CBR) sources. A 512 byte data packet is used for all CBR sources. The inter-arrival time of CBR sources is fixed at 4.5 seconds. The CBR agent will be attached to a UDP agent, which is in turn attached to the source node. The source-destination pairs are generated according to (2). Each data point represents an average of at least five runs with identical topology, but different traffic patterns (source-destination pairs).

4.2 Numerical and Simulation Results

Figure 5 shows the results of Theorem 3.3 and generated traffic pattern. The average number of the communicating sessions terminated/initiated at a node is counted. The analysis results are calculated as (12)[4]. It seems the session number per node in simulation increases not as rapidly as theoretical results. This is due to edge effects.

The second experiment verifies the results of Theorem 3.6 in Figure 6. The theory results are according to (20). We use AODV as the routing protocol in the simulation for Figure 6. We counted the total number of sessions initiated or passing through a node. Then the average percentage of forwarding sessions at a node is counted. Figure 7 shows the average total number of sessions from AODV simulations

[3]Slow simulation speed and large memory requirement for the *ns-2* models prevented us from using larger network at this stage.

[4]As pointed out earlier, in all the following numerical analysis results, we summed r from 1 to the network diameter $\sqrt{N}$ instead of to infinity in calculating $f(k)$ since our simulations are for finite networks.

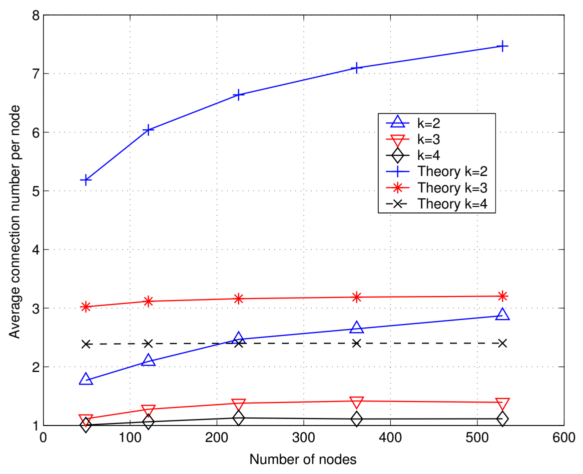

Figure 5: Average number of source-destination sessions per node $E[S_t]$ (12), $c = 0.5$. $k > 2$ for convergence.

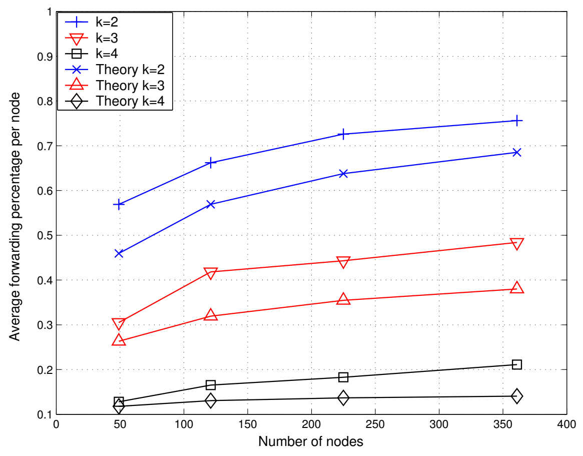

Figure 6: Average forwarding percentage per node q (20), AODV simulations, $c = 0.8$.

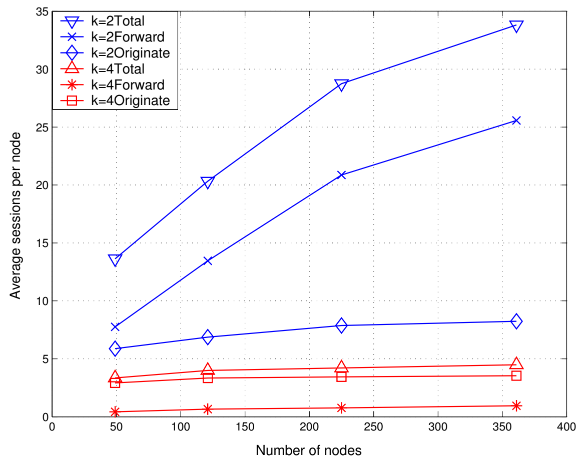

Figure 7: Average number of total (source-destination and relaying) sessions per node $E[S]$ (18), $c = 0.8$, AODV simulations. $k > 3$ for convergence.

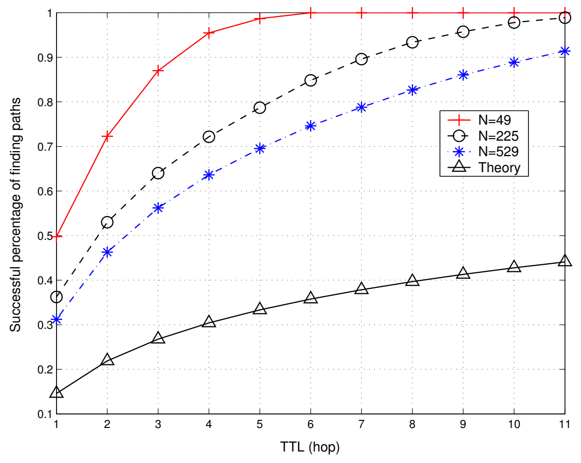

Figure 8: Successful percentage of finding paths as a function of TTL (10), $k = 2$, $c = 0.8$, AODV simulations.

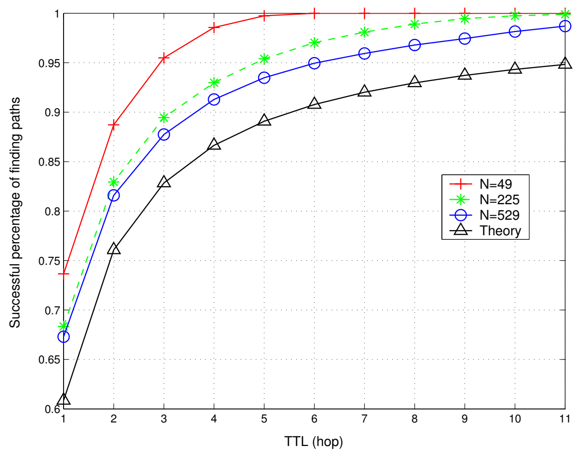

Figure 9: Successful percentage of finding paths as a function of TTL (10), $k = 3$, $c = 0.8$, AODV simulations.

and analysis. The numerical results are calculated according to (18).

Figures 8, 9 and 10 verify Claim 3.3. The numerical results are calculated as $(1 - \alpha)$ with different TTL values in (10). In the simulated scenarios, we set up communicating sessions initiated from the center node. AODV is used as the routing protocol in the simulation about TTL. Parameters of AODV source code (NET_DIAMETER, TTL_INCREMENT and TTL_THRESHOLD) are modified then re-compiled for each simulated scenario. For every session, if the source node can not find a route to destination due to TTL limit, the route request is counted as unsuccessful. We count the total number of unsuccessful route requests. From Figure 8, 9 and 10, we observe that as the network size N increases, simulation results and theoretical results provide an increasingly better match.

As mentioned earlier, when a node is turned off, route error packets need to be sent to the nodes which have active sessions terminating at the failed node. Both AODV and DSR are considered. Since default options of DSR in *ns-2*

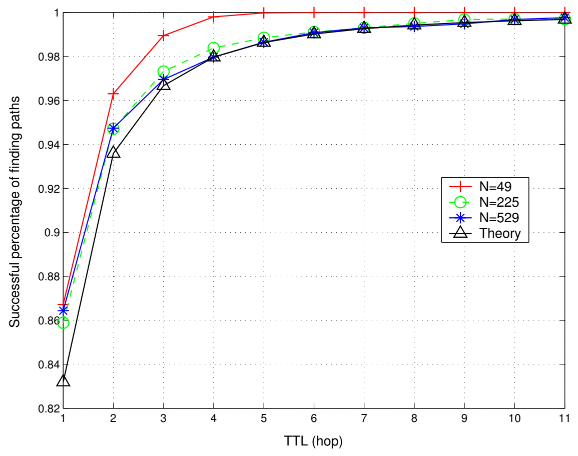

Figure 10: Successful percentage of finding paths as a function of TTL (10), $k = 4$, $c = 0.8$, AODV simulations.

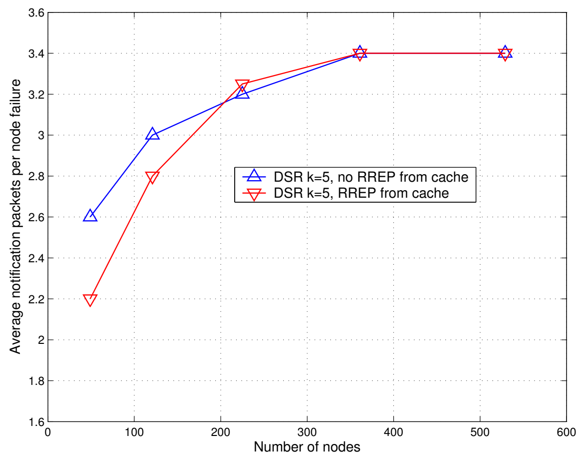

Figure 11: Effect of route cache on RERR for a failed destination, DSR, $k = 5$, $c = 0.5$.

turn on route cache which is not considered in our theoretical analysis, it is interesting to know what is the effect of route caching on scalability of routing protocols. Simulation results show that route caching doesn't have significant effect on the scalability results as we can see from an example simulation result of DSR with and without route caching in Figure 11. But to be consistent with our theoretical analysis which doesn't consider route cache, "Replying to Route Requests using Cached Routes" and "Packet Salvaging" in DSR [2] are turned off in all other simulations.

Figure 12 verifies Theorem 3.12. In the simulated scenarios, we set up communicating sessions whose destination is the center node. After every session has run for a long time (enough for every source node to find a route to destination) to avoid transient effects, the destination node is turned off. Then we count every route error packet sent from neighboring nodes. The theoretical result is given by (46).

Theorem 3.15 derives the average number of packets to notify about the failure of a node to the nodes which have active sessions ending at or passing through the failed node. This time, only DSR is selected as routing protocol for Figure 13 as we mentioned in Section 2.2.2. Theoretical results are obtained from (48), which states that the route failure notification overhead infinitely scale only if $k > 4$.

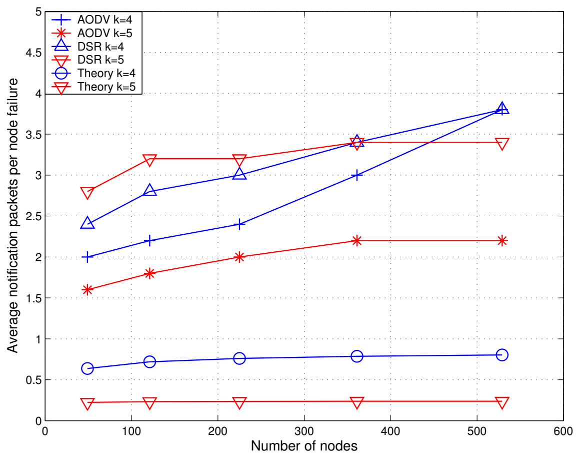

Figure 12: Average route error packets for a failed node as destination (46), $c = 0.5$.

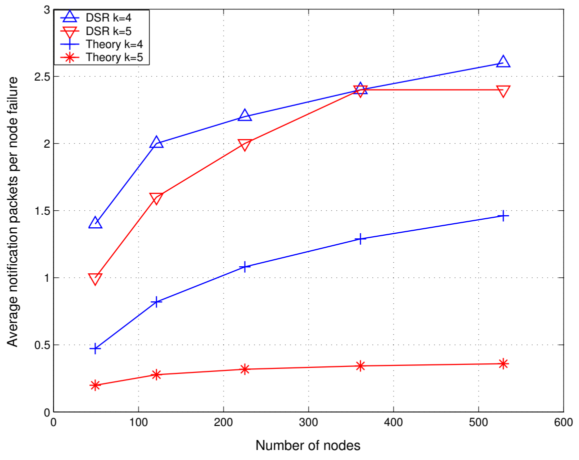

Figure 13: Average route error packets for a failed node as destination or relay (48), $c = 0.5$.

5. SIMULATIONS OF RANDOM NETWORK TOPOLOGIES

Though our analysis is based on the Manhattan grid, we run simulations for random network topologies in order to check the validity of the analysis regarding the scalability as a function of k. We repeat all the simulations for Manhattan grid. Only the different aspects of the simulations are described here.

We still consider five networks of size $N = 49$, 121, 225, 361 and 529 nodes. Unlike the case of Manhattan grid simulations, nodes here are randomly placed in a square area of size A. A random network topology example is shown in Figure 14. Let L denote the side length of the square, then $L = \sqrt{A}$. Let r_0 denote the communication radius of a node, which is the same for all the nodes in the network. Let g denote the average number of nodes within a direct communicate area of a given node (i.e. average degree of a

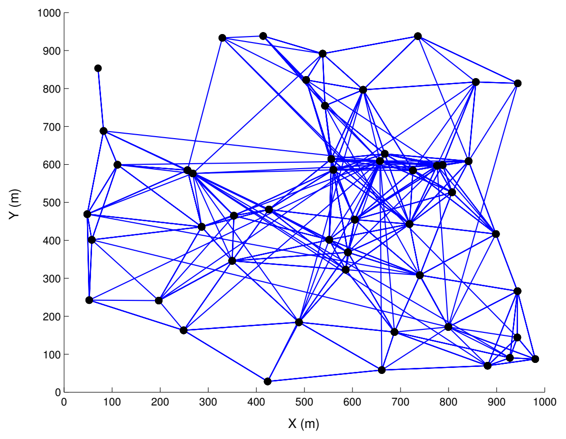

Figure 14: A snapshot of a random network topology with $N = 49$ and $g = 10$. The dark spots represent the nodes.

node). If we use λ to denote the node density for a network size, then we have

$$g = \lambda \pi r_0^2 \tag{50}$$

From [7, 8], we know that this average node degree should be $\Theta(\log N)$ to keep the network asymptotically connected. Considering the largest network size in our simulation, we choose $g = 10$ or $g = 20$. The default value for r_0 in *ns-2* is 250 meters. The side length L of every network size is calculated from

$$L = \lfloor r_0 \sqrt{N\pi/g} \rfloor \tag{51}$$

For every network size, a fully connected topology is generated (we generate a number of topologies, check the connectivity of each using standard methods, and select only those that are connected). Then for every topology, we run Floyd-Warshall [9] algorithm to get the *connectivity matrix* whose elements indicate the number of hops between each node. Similar to the simulations for Manhattan grid, we choose our traffic sources to be Constant Bit Rate (CBR) sources. The source-destination pairs are also chosen according to (2) where the hop numbers between each node are looked up from the connectivity matrix.

Figure 15 shows the generated traffic patterns for $k = 2, 3$ and 4. We draw the curves of the average number of established source/destination sessions over different network sizes. We notice that the curves become flat for $k = 3$ or 4. For $k = 2$, the curve continue increasing as the network size increases. The scalability behavior for different k are consistent with the theoretical result. From, Corollary 3.4, the average number of source/destination sessions becomes infinitely scalable only if $k > 2$.

The second experiment shows the results of q in Figure 16. We use AODV as the routing protocol. We counted the total number of sessions initiated or passing through a node. Then the average percentage of forwarding sessions at a node is counted. From the experiment, we notice that the probability becomes constant for $k = 4$ when the network size is larger than 200 nodes. But for $k = 2$ and 3, the probability continue increasing as the network size increases. Besides, by fixing all other parameters, the change of the probability is not very sensitive to the change of g. For $k = 4$, the probability values are almost the same for $g = 10$ and $g = 20$.

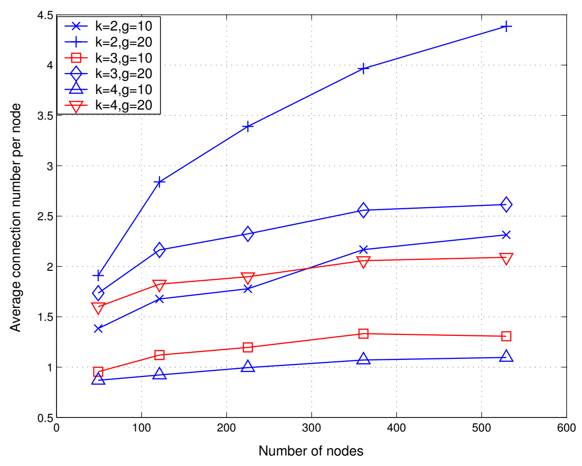

Figure 15: Average number of source-destination sessions per node $E[S_t]$ (12), $c = 0.1$. $k > 2$ for convergence.

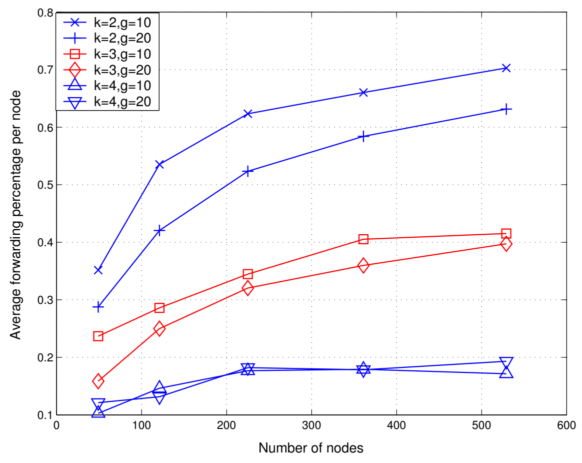

Figure 16: Average forwarding percentage per node q (20), $c = 0.1$.

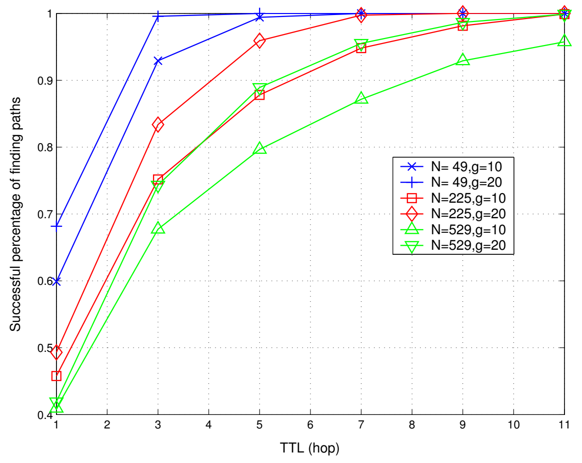

Figure 17: Successful percentage of finding paths as a function of TTL (10), $k = 2$, $c = 0.1$, AODV simulations.

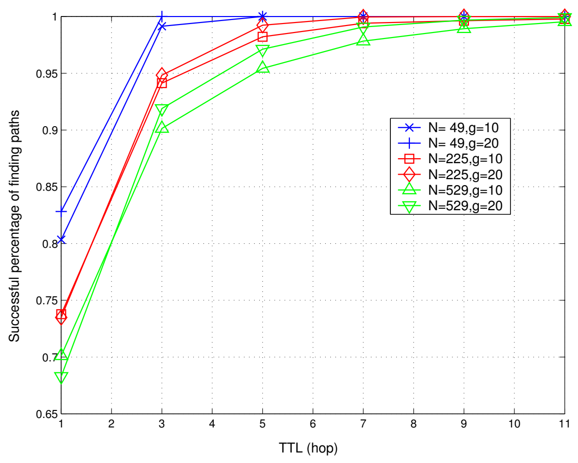

Figure 18: Successful percentage of finding paths as a function of TTL (10), $k = 3$, $c = 0.1$, AODV simulations.

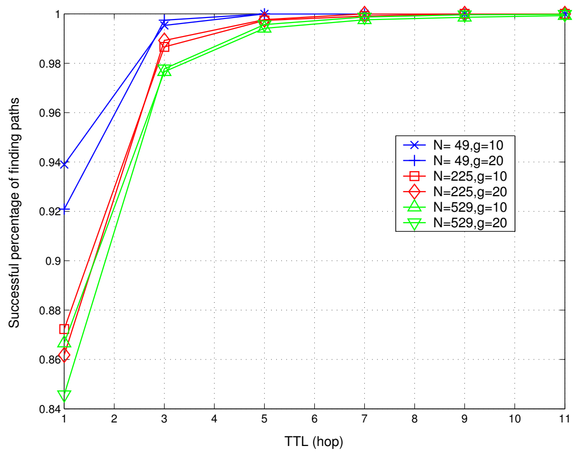

Figure 19: Successful percentage of finding paths as a function of TTL (10), $k = 4$, $c = 0.1$, AODV simulations.

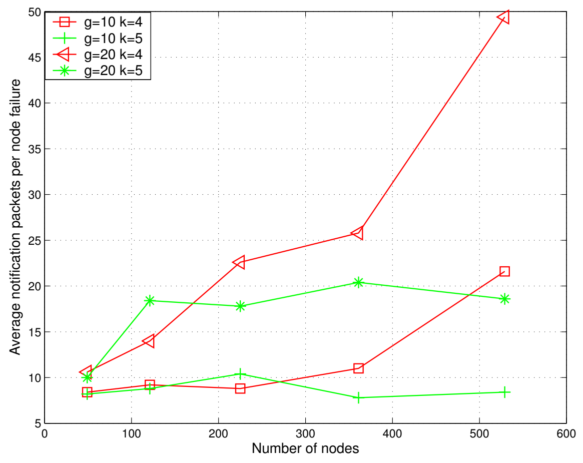

Figure 20: Average number of route error packets for a failed node as destination (46), AODV, $c = 0.5$.

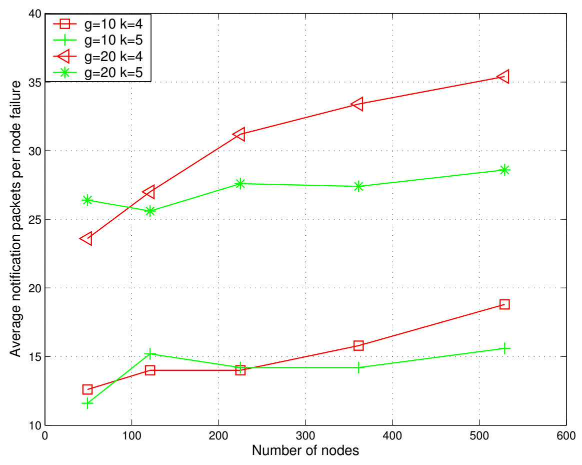

Figure 21: Average number of route error packets for a failed node as destination (46), DSR, $c = 0.5$. $k > 5$ for convergence.

We can use Theorem 3.6 to validate this experiment. For $k > 3$, the probability is given by (20). This is consistent with the experiment results (that the changes of probability is not sensitive to the changes of values of g). From theoretical analysis, q is the probability that a session is a relay session, and is 1 if $k \leq 3$. From our experiment, the probability values of q for $k = 2$ and $k = 3$ continue increasing as the network size increasing, but not increasing fast enough as we expect. One explanation for the slow increase of q could be the edge-effects.

To study the effect of TTL value $T(\alpha)$ on the probability of finding destination node $(1 - \alpha)$, we set up the following simulation scenarios. For each scenario, we initiate our communicating sessions from the node which is nearest to the center of the field. For every session, if the source node can not find a route to destination due to TTL limit, the route request is counted as unsuccessful. We measure all the number of unsuccessful route requests. The y-axis in Figure 17, 18 and 19 is the values of $(1 - \alpha)$. From these figures, we see that the smaller the value of k is, the faster $(1 - \alpha)$ approaches to one. This phenomena is consistent with our theoretical result.

For the simulations of route error notification, we set up communicating sessions whose destination is the node that is the nearest to the center of field. After every session has run for a long time (enough for every source node to find a route to destination) to avoid transient effects, the destination node is turned off. Then we count every route error packet sent from neighboring nodes of that failed node. The simulated results using AODV and DSR are plotted in Figure 20 and Figure 21 respectively.

We also count the average number of packets to notify the failure of a node to the nodes which have active sessions that end at or pass through the failed node. The simulated results of DSR with different traffic parameters are drawn in Figure 22. When a node fails, the failure notification message is not shared for the DSR protocol. The simulation results setup using DSR protocol can be used to compare our theoretical result given by Theorem 3.15. According to the theorem, the average number of root failure packets scale only if $k > 4$. In our simulation scenario, we do see

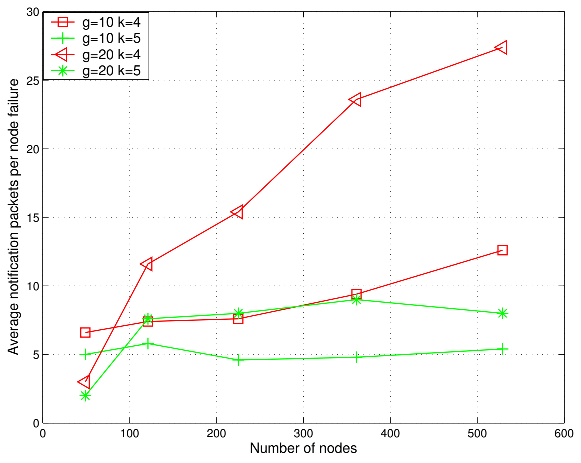

Figure 22: Average number of route error packets for a failed node as destination or relay (48), DSR, $c = 0.5$. $k > 5$ for convergence.

that the average number of packets continues increasing as the size of the network increases for $k = 4$. But for $k = 5$, the average number of packets increases for small network size, and becomes a constant for large network sizes.

6. CONCLUSION

In this paper, we developed a mathematical analysis of the overhead of a broad class of reactive routing protocols. Specifically, we focused on routing overhead associated with (i) route discovery and (ii) route failure notification. The analysis was developed in the context of an unreliable sensor network, modelled by an unreliable Manhattan (i.e. degree 4) grid. Sensitivity of the results was analyzed by comparing them against simulations of regular and random topologies.

Several issues were considered, such as the key differences between distance-vector and source-route based routing protocols regarding the mechanism of route failure notification, and the use of route repair.

Our analytical results point to the key role that the traffic pattern plays in defining the scalability of these protocols. Expressions for various quantities of interest, as well as conditions for scalability were derived and validated via *ns-2* simulations. Although we do not model many aspects of the network, including MAC layer details and route caching details, there is a reasonable match between the analytical results and simulations that do take into account those detailed aspects. Specifically, the simulations validate the infinite scalability results.

In this paper, we analyzed two difference cases of network topologies. Our infinite scalability results should hold for many other topologies and traffic models. Let r_0 and r denote the communication radius of a node, and the physical distance between two nodes, respectively. Let $l = \lceil \frac{r}{r_0} \rceil$ denote the *characteristic distance* between two nodes. Let $p_L(l)$ denote the probability that an active session exists between two nodes having a characteristic distance l. We conjecture that our infinite scalability results still hold if $N(l)$ scales as $\Theta(l^2)$ and $p_L(l)$ scales as $\Theta(l^{-k})$. Here, $N(l)$ is the average number of nodes within a characteristic distance l from a node.

Several avenues of future work remains, including the analysis of the overhead of other types of routing protocols such as hierarchical routing. Further, the results from this work could be applied to aid the design of routing protocols in the future. For example, the results can be applied to limit the size of a cluster-head such that the average routing overhead is bounded by some given constant.

Appendix: Proof of Theorem 3.7

Let o denote the node located at $(0,0)$. Let M denote the set of all the nodes in the network, and M/o denote all the nodes except the node o. Without loss of generality, choose node o as the destination node. We start with the calculation of $p_s(0)$, the probability that no node has an active session terminated at node o. From our model assumption, the events that different nodes communicate with node o are independent. Therefore, the value of the $p_s(0)$ is the product of the probability that any given node is not communicating with node o. Hence we have (23).

Now, for $p_s(1)$, the probability of having only one node is communicating with the node o. Let $p_{a,o}$ denote the probability that there is an active session started from node a and terminated at node o. For a given node $a \in M/o$, the probability that this node is communicating with node o and no other node is communicating with o can be expressed as $p_{a,o} \prod_{b \in M/\{o,a\}} (1 - p_{b,o})$, where $a \in M/o$.

$p_s(1)$ is the sum of the probabilities of all nodes in the set of M/o. We have

$$\begin{aligned} p_s(1) &= \sum_{a \in M/o} \left[p_{a,o} \prod_{b \in M/\{o,a\}} (1 - p_{b,o}) \right] \\ &= \sum_{a \in M/o} \left[\frac{p_{a,o}}{1-p_{a,o}} \prod_{b \in M/o} (1 - p_{b,o}) \right] \\ &= p_s(0) \sum_{a \in M/o} \left[\frac{p_{a,o}}{1-p_{a,o}} \right] \\ &= p_s(0) G_1 \end{aligned} \tag{52}$$

First, we denote $q(a) = \frac{p_{a,o}}{1-p_{a,o}}$. Assume that, we have found the values of $p_s(i)$ and G_i, where $i \in [0, 1, \ldots, n-1]$. From the definition of $p_s(n)$, we have,

$$\begin{aligned} & p_s(n) \\ &= \left(\sum_{a_1,\ldots,a_n \in M/o, a_i \neq a_j} \left[\prod_{i=1}^{n} q(a_i) \right] \right) (p_s(0)) \end{aligned} \tag{53}$$

From there, we have the following equation

$$g_s(n) = \frac{p_s(n)}{p_s(0)} = \left(\sum_{a_1,\ldots,a_n \in \mathrm{M}/o, a_i \neq a_j} \left[\prod_{i=1}^{n} q(a_i) \right] \right) \tag{54}$$

Hence, we can deduce a recursive equation to find $p_s(n)$,

$$\begin{aligned}
&g_s(n-1)g_1(1) \\
&= \sum_{a_1,\ldots,a_{n-1}\in M/o, a_i\neq a_j} \left[\prod_{i=1}^{n-1} q(a_i)\right] \sum_{a\in M/o} q(a) \\
&= \sum_{a_1,\ldots,a_{n-1}\in M/o, a_i\neq a_j, b\in M/o} \left[q(b)\prod_{i=1}^{n-1} q(a_i)\right] \\
&= \sum_{a_1,\ldots,a_n\in M/o, a_i\neq a_j} \left[\prod_{i=1}^{n} q(a_i)\right] + \\
&\sum_{b\in M/o, a_1,\ldots,a_{n-2}\in M/\{o,b\}, a_i\neq a_j} \left[[q(b)]^2\prod_{i=1}^{n-2} q(a_i)\right] \\
&= g_s(n) + \\
&\sum_{b\in M/o, a_1,\ldots,a_{n-2}\in M/\{o,b\}, a_i\neq a_j} \left[[q(b)]^2\prod_{i=1}^{n-2} q(a_i)\right] \\
&= g_s(n) + \\
&\sum_{b\in M/o, a_1,\ldots,a_{n-2}\in M/\{o,b\}, a_i\neq a_j} \left[[q(b)]^2\prod_{i=1}^{n-2} q(a_i)\right]
\end{aligned} \tag{55}$$

For the second term,

$$\begin{aligned}
&\sum_{b\in M/o, a_1,\ldots,a_{n-2}\in M/\{o,b\}, a_i\neq a_j} \left[[q(b)]^2\prod_{i=1}^{n-2} q(a_i)\right] \\
&= \sum_{b\in M/o, a_1,\ldots,a_{n-2}\in M/o, a_i\neq a_j} \left[[q(b)]^2\prod_{i=1}^{n-2} q(a_i)\right] \\
&- \sum_{b\in M/o, a_1,\ldots,a_{n-3}\in \mathrm{M/\{o,b\}}, a_i\neq a_j} \left[[q(b)]^3\prod_{i=1}^{n-2} q(a_i)\right] \\
&= \left(\sum_{b\in M/o,} \left[[q(b)]^2\right]\right)\left(\sum_{a_1,\ldots,a_{n-2}\in M/o, a_i\neq a_j} \left[\prod_{i=1}^{n-2} q(a_i)\right]\right) \\
&- \sum_{b\in M/o, a_1,\ldots,a_{n-3}\in M/\{o,b\}, a_i\neq a_j} \left[[q(b)]^3\prod_{i=1}^{n-3} q(a_i)\right] \\
&= g_s(n-2)G_2 \\
&- \sum_{b\in M/o, a_1,\ldots,a_{n-3}\in M/\{o,b\}, a_i\neq a_j} \left[[q(b)]^3\prod_{i=1}^{n-3} q(a_i)\right]
\end{aligned} \tag{56}$$

By Recursively applying the same deduction to the second term, we prove the theorem.

7. REFERENCES

[1] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc on-demand distance vector (AODV) routing. Internet Draft draft-ietf-manet-aodv-12.txt, Mobile Ad Hoc Networking Working Group, November 4 2002.

[2] D. B. Johnson, D. A. Maltz, Y.-C. Hu, and J. G. Jetcheva. The dynamic source routing protocol for mobile ad hoc networks (DSR). Internet Draft draft-ietf-manet-dsr-07.txt, IETF MANET Working Group, February 21 2002.

[3] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad-hoc network routing protocols. In *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'98)*, pages 85–97, Dallas, Texas, USA, October 1998.

[4] ns-Network Simulator. 1995. http://www.isi.edu/nsnam/ns/.

[5] Y.-C. Hu and D. B. Johnson. Caching strategies in on-demand routing protocols for wireless ad hoc networks. In *Proceedings of the sixth annual international conference on Mobile computing and networking*, pages 231–242, Boston, Massachusetts, United States, 2000.

[6] D. Estrin et al. *Embedded Everywhere: A research agenda for networked systems of embedded computers.* National Research Council, 2001.

[7] P. Gupta and P. R. Kumar. Critical power for asymptotic connectivity in wireless networks. In W. M. McEneaney, G. Yin, Q. Zhang, and Birkhäuser, editors, *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming*, Boston, 1998.

[8] F. Xue and P. R. Kumar. The number of neighbors needed for connectivity of wireless networks. To appear in Wireless Networks.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press, second edition, 2001.