Asymmetric Routing in Constrained Mobility Networks

Mukundan Sridharan and Anish Arora
Department of Computer Science and Engineering
The Ohio State University
Email: {sridhara, anish}@cse.ohio-state.edu

Abstract

We revisit the age-old question of maintaining routes in the context of a campus area network where mobile devices multi-hop to access one or more resource-rich base stations. In this constrained-mobility and asymmetric environment, we find that as the network density increases, the average time for a mobile device to lose connectivity to all of its neighbors grows much faster than the minimum time to lose connectivity to all of its neighbors. This leads us to consider routing where (a) route update decisions are mobility aware, (b) path maintenance of devices is event-driven rather than periodic, (c) a device can choose from several neighbors to forward its messages, and (d) route repair is offloaded to base station(s). We compare the resulting braided multi-path tree protocol, which we call *Asymmetric mobility-Event driven Routing (AER)*, with multi-path variants of the classic DSDV, OSPF, and Bellman-Ford protocols. Through both analysis and simulation, we find that AER outperforms the others for most of the low mobility region.

I. Introduction

Convergence in mobility devices has combined the capabilities of a cellphone, PDA, music player, video camera, gaming console, GPS and other sensors, and one or more short range wireless radios (such as Bluetooth, WiFi, and 802.15.4). These feature-rich devices are now in a position to support a number of locality-specific applications, such as Buddy Messaging, data collection in sparse Sensor Networks, Virtual Social Networks where mobile users can publish profiles and search for others with similar interests, and Alert Services which informs users about important events or emergencies in the vicinity.

This trend motivates us to explore the desirability and feasibility of implementing locality-specific wireless applications primarily via local mobile-to-mobile multi-hop communications. We consider a network model with dense regions of mobile users in a campus area with office buildings, cafes, terminals, etc. Intra-regionally, the user mobility pattern is typically that users are largely static or they move rarely with low speed. Each region is *asymmetric* in that it one or more non mobile resource-rich nodes that can take on the burden of providing some basic network services. We thus have a multihop wireless network with infrastructure support per region.

In this paper, we revisit the age-old question of maintaining routes in the context of a constrained-mobility and asymmetric environment. Communications between devices and the base station is a primary requirement here as opposed to direct device to device communications, since the latter can be realized indirectly via the base station. Device efficiency, in terms of storage, program size, and energy efficiency is also a primary requirement. This leads to studying whether one of the many routing algorithms used in purely static environments or purely mobile environments is well suited, or an alternative algorithm which exploits the asymmetry between the base station and the devices is better suited.

One finding in this paper is that as the density of devices in a mobile network increases, the average time for a device to lose all of its neighbors increases more quickly than the minimum time for a device to lose all of its neighbors. This has several implications for maintaining a routing structure: First, awareness of device mobility should be exploited to trigger routing message updates, i.e., routing updates are sent based not on the changes in the 'state' of the routing table, but rather on the changes in the actual state of the device, e.g., from mobile to static. Second, synchronized repair of all devices (which depends on the minimum time for a device to lose all of its neighbors) would need to be more frequent than asynchronous repair of each device (which depends on the average time); of course, a tradeoff in synchronized versus asynchronous repair is that the former is more efficient in terms of message cost per device than the latter. And third, choosing more than one parent would substantially decrease the rate of node repair.

We are therefore led to consider a protocol that combines the elements of mobility awareness, asynchronous repair, and multiple parents in an asymmetric setting where the route repairs for each devices are calculated at the base station(s). We call this protocol the *Asymmetric mobility Event-driven Routing (AER)*.

Contributions of the paper. We make a three fold contribution: One, we investigate the stability of braided multi-parent trees under mobility. Two, we propose AER routing that exploits the resource rich base stations and knowledge of mobility. Third, we implement multi-parent adapted versions of the classic OSPF, DSDV, and

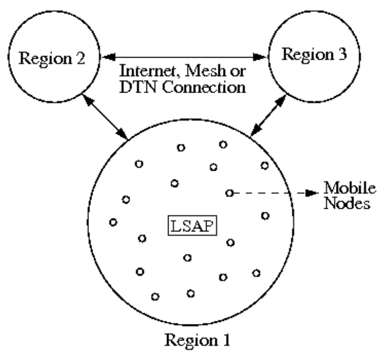

Fig. 1. Campus Wide Mobile Network

Bellman/Ford protocols, for comparison with AER in the constrained mobility regime. We analyze the message and latency costs of all these protocols, and evaluate their performance in simulations as well, to find that AER outperforms the others for most of the constrained mobility region.

Organization of the paper. In Section II, we describe our network and mobility model, and analyze the properties of the mobility model. In Section III we formally define the routing problem, and describe our AER protocol. In Section IV, we describe adaptations of classic protocols and compare them analytically with AER. The protocols are evaluated using simulations and the results are presented in V. In Sections VI and VII, we present the related work and our conclusions.

II. System Model

A. Network Model

University, business, and industrial campuses typically have a number of human and vehicles carrying devices. Users mostly frequent certain 'regions' (e.g., buildings and warehouses) and more rarely move between these dense regions. More precisely, our campus area network is modeled as follows.

Figure 1 shows our campus area network model.

1) It has 'regions' which are densely populated with static or constrained-mobility nodes.
2) Each region has one or more non-mobile devices called the Local Service Access Points (LSAPs). LSAPs are assumed to not be storage space- or energy- constrained.
3) LSAPs of different regions may be inter-connected via the Internet, wireless point-to-point links, a wireless mesh network, or via the nodes moving between the regions (serving as delay-tolerant data carriers), to form a campus-wide network. The type of connection between the LSAPs is transparent to the nodes.
4) Each node knows its unique ID but not necessarily its geographical location in the network.
5) Each node is assumed to have the same transmission radius. A node is a neighbor of another node if the distance between them is at most the transmission radius. We assume that links are symmetric; while this is not always true in practice, with 5-30% links in wireless mesh sensor networks being unidirectional, it is arguably valid for most links.

B. Application Model

LSAPs provide basic networking and application services. A plausible application framework is shown in Fig. 2. The framework supports two basic applications by default, namely the 'Text Messaging App' (TM App) that enables any node to send a message to any other node in the system via the LSAP, and the Buddy Finder, which we described in the previous section and presumably interacts with an 'Active Register' of the nodes in the region maintained in the LSAP. Other applications are optional, both on the LSAPs and the nodes. The messaging layer is a virtual one that multiplexes and demultiplexes application packets.

Several interesting applications can be developed using the query/response model between the mobile user and the LSAP, like a *Buddy Finder* application, where a user can query the LSAP to find if his buddies are also in the region, data collection in *Spare Sensor Networks*, a *Virtual Social Network* application where mobile users could publish 'profiles' and search others with similar 'interests', an *Event/Emergency Alert Service* which alerts users on a campus about important events or emergencies in the region. The application in this case does not need to

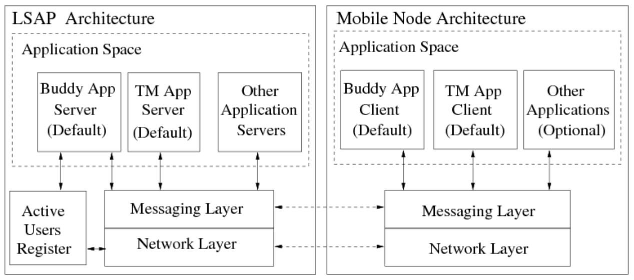

Fig. 2. LSAP - Mobile Node Application Architecture

use the cellular infrastructure, instead it may only use short-range radio communications with LSAP, sensors, and other mobile users in a multi-hop setting. Many of these applications are readily extrapolated to be useful across regions.

C. Mobility Model

Given our focus on mobility inside campus areas, which is often that of humans walking inside or between buildings or that of vehicles moving with strictly limited speeds, the rate of mobility we are concerned with is slower than what is considered in a typical vehicular network. Also, as suggested by previous work on campus area mobility based on WiFi traces [4], only a small percentage of the network is mobile at any given time. Finally, as observed by [8], humans users once in motion move continuously for some time towards some particular destination and, upon reaching there, tend to remain for some time. We refer to this a type of mobility as *Constrained Mobility*.

Based on these characteristics of low speed, rare movement, and relocate-and-pause, we model constrained mobility in a fashion similar to the Random Way Point (RWP) model [9], but with important differences. Formally,

1) Each node is in one two states, ‘static’ or ‘mobile’.
2) A probability of movement, P_m, dictates the transition from the static to the mobile state.
3) Once a node is in the mobile state, it picks a random (normally distributed) destination and moves towards that destination with constant velocity. When it reaches the destination, it enters the static state.

Unlike the RWP model, which uses a velocity that is uniformly distributed between [0, Vmax], we use a constant velocity (humans, for instance, move more or less at a speed of around 1.5 m/s). Also, the pause time in the RWP model is also uniformly distributed, whereas the waiting time in the static state in our model is exponentially distributed with a mean of $1/P_m$. That is, the longer a node is in the static state, the more likely it is to move, which seems somewhat natural in our context. Note also that the number of mobile nodes in our model is also decided by the probability of movement.

D. Analysis of Mobility Model

The RWP model has received the following two criticisms [5], [15]: One, the average speed of the nodes in the network become progressively slower with time as nodes become trapped in low speed states. Thus, time averages of simulations in the model can be misleading. Second, the average neighbor density in the network oscillates periodically as nodes converge and diverge.

Our mobility model does not suffer from these misbehaviors. The first one is avoided trivially by our use of a constant velocity for all nodes, thus the average speed remains the same. As to the second one, both our simulations and user mobility experiments on our campus show that the average neighbor distribution in our model varies randomly. Fig 3 shows the average neighbor density of a node for a 500 second simulation with 100 nodes of the mobility model described in Section II-C. We see from Fig 3 that even though the neighbor density does not stay constant we do not see the typical density waves Fig 4 shows a neighbor density of a mobile node from a actual mobility experiment involving 100 nodes in our university campus. As the figure shows the neighbor density varies rather randomly.

It also seems intuitively plausible that neighborhood density is neither uniformly distributed nor oscillating with a fixed period. For example, the average traffic in buildings is high in the mornings and evenings, and in cafeterias is high at mealtimes.

If all nodes have K neighbors, for $K > 1$, the minimum and average time for a node to lose all neighbors is a direct function of λ_t ($\lambda_t = 1/P_m$), the mean waiting time in the stationary state.

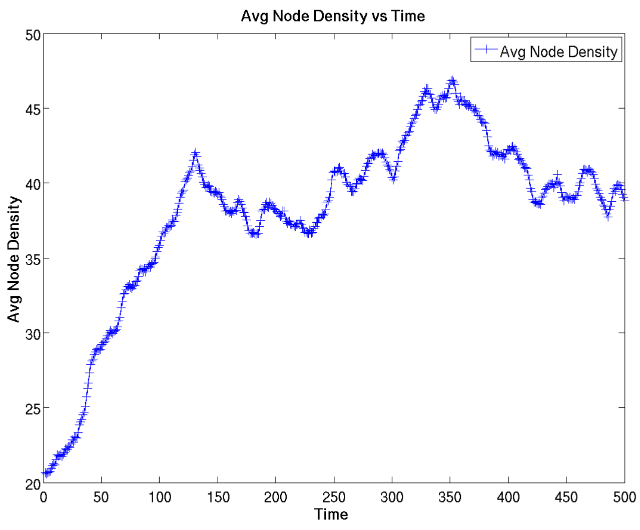

Fig. 3. Average neighbor density of nodes in a simulation with 100 nodes

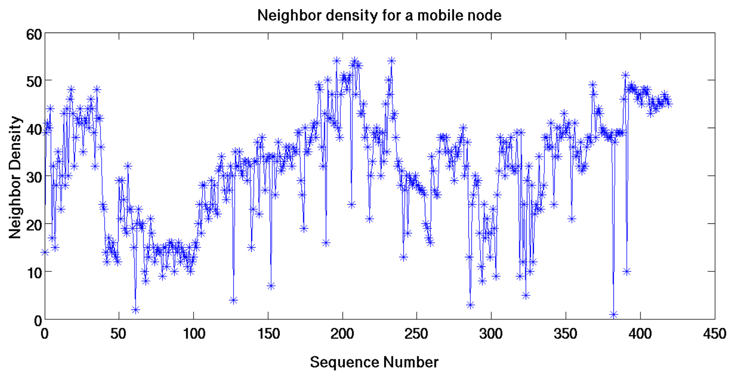

Fig. 4. Experimental neighbor density of typical mobile node from a experiment with 100 mobile nodes

Property 1: *The average time, $t(K)_{avg}$, for nodes to lose all their K neighbors increases linearly with K, while the minimum time, $t(K)_{min}$, increases sublinearly.*

Proof: When a node is in the static state, its waiting time is distributed exponentially, with mean λ_t. Let $X_1..X_K$ be random variables for neighbors $1..K$ that denote their respective times to switch to the mobile state. And let X_0 denote the time for the node itself to switch to the mobile state. Now, $X_0..X_K$ are independent of each other and identically distributed exponential random variables.

If we assume that a node will lose all its neighbors as soon as it switches to the mobile state, the time for the node in question to lose all its neighbors is:

$$t(K) \;=\; min(X_0, max(X_1..X_K)) \tag{1}$$

In order to take into account the time taken by nodes to move out of the transmission range R of their neighbor, let D_{avg} be the average distance a node travels to move out of transmission range of a neighbor D_{avg} is of the order of R. To motivate this we observe that if a node move a distance of $2R$ in any direction it would lose all its neighbor and thus the maximum distance a node has to move to lose a parent is bound by $2R$. A node which is at the edge of its neighbors transmission range can lose its neighbor by moving a infinitismally small distance which can be approximated to zero. Thus the minimum distance a node needs to move to lose its neighbor is 0. Thus $0 \;<\; D_{avg} \;<\; 2R$

Now let v the constant node velocity, then, the time to move out of a node's neighborhood is given by D_{avg}/v

and the equation for $t(K)$ becomes:

$$t(K) \;=\; \frac{D_{avg}}{v} \;+\; min(X_0, max(X_1..X_K)) \tag{2}$$

The minimum of two exponential distributions is exponential, finding the maximum of exponential distributions is tougher. Asymptotically $max(X_1..X_K)$ has a constant standard variation that is independent of K and a mean that increases as log(K). We can therefore calculate the mean time using a statistics toolbox, such as Matlab. Figure 5 plots the result of the calculation. It shows the minimum time, defined as the 10 percentile value of $t(K)$, and the average time. The minimum time increases sublinearly (and is fairly constant) while the average time increases linearly with K. For any given value of K, we can directly calculate the second term of the equation from the figure. Thus, for example, for $K = 3$, $t(3)_{avg} = 0.65\lambda_t$ + D_{avg}/v and $t(K)_{min} = 0.1\lambda_t$ + D_{avg}/v. $\square$

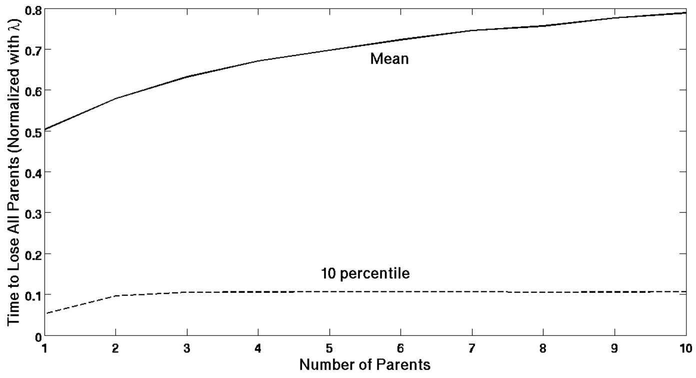

Fig. 5. Average and minimum time to lose all K neighbors

Property 1 is preserved if nodes have different numbers of neighbors, but each node selects exactly K of its neighbors to be its parents in a multi-parent tree; in this case, the property becomes relative to the node losing all its K parents. Further validation of Property 1 is given in Section V via simulations.

III. Asymmetric Routing

We begin by defining routing requirements in asymmetric mobile networks, then discuss the intuition for routing based on multi-parent trees, and finally describe the AER protocol.

A. Problem Definition

The goals of routing in an asymmetric mobile network are:

- For each mobile node, to discover and maintain paths from it to the LSAPs of its region, assuming that such paths exists.
- For each LSAP, to maintain paths to each node in its region, assuming that such paths exists.
- Each LSAP should maintain a registry of active nodes by reliably detecting each node 'join' or 'leave' from its regions network within a reasonable amount of time.

Note that a solution to the problem may require nodes to only maintain (one or more) parents as opposed to the entire paths. And as motivated previously, any-to-any node routing may be accomplished indirectly via LSAPs, which would reduce the program complexity and energy consumption of node routing, and exploit the possibility that most application traffic is bound to and from the LSAP.

B. Intuition for Multi-Parent Routing

Multipath routing algorithms have better path availability than single-path routing. Most extant multipath routing protocols are, however, reactive in nature [10], [11], [14]; this is likely because flooding is the commonly used mechanism for finding multiple paths between two nodes and flooding is easily combined with on-demand route discovery. Distance-Vector Multipath Routing [13] is an exception —it finds all available loop-free shortest paths to the destination— but incurs significant storage and messaging overhead. We also note that multipath algorithms have addressed load balancing and increasing reliability of message delivery (by forwarding on several paths), but our focus is on increased efficiency in the presence of mobility.

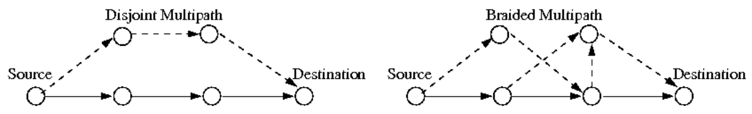

Fig. 6. Difference between Disjoint and Braided Multipath

One implication of Property 1 is that mobility is better tolerated by maintaining more than one parent per node, even if the paths from these parents to the LSAP are not necessarily link disjoint or even of shortest length. On a related note, [6] shows that in a static sensor network setting the braided-multipath approach results in approximately the same path availability as does the disjoint-multipath approach. In the latter, no two paths from a source to a destination share any common links, while in the former for every link on the primary path there exist K-alternate links. Thus, there is no single bottleneck link in the braided multi-path approach whose disconnection results in the non-availability of a path between the source and the destination. Figure 6 illustrates the difference between a disjoint and a braided multipath. A protocol that implements such a tree is also likely to be highly efficient in the usage of storage space. For example, the AER protocol we propose next takes only 2 bytes (ID and hop count) of information for each of the K (typically 3) parents, which means it needs only 6 bytes per LSAP for the routing table.

Our approach to multipaths is to ensure that each node has at least K-parents, of which at least one is a shortest path parent. We build a spanning tree rooted at each LSAP, such that each node in the region has multiple parents which are of lesser or the same hop-count as that node. If routing through the primary parent (one with the lowest hop count) fails, then other parents are explored for availability. This algorithm is easily implemented on a resource constrained mobile device. Although it is designed for any value of K, in practice, we find that there is not substantial improvement when K exceeds 3, so all our evaluations in this paper use 3 parents.

C. Asymmetric Mobility Event-Driven Routing (AER)

To maintain parents at nodes that lead to an LSAP as well as paths at an LSAP to lead to each reachable node in its region, protocol AER implements a centralized form of link-state routing at the LSAP. Each LSAP collects and maintains an up-to-date snapshot of the 'link-states' of all reachable nodes in its region. Once this region snapshot (the so-called 'network-map') is reliably obtained by the LSAP, it calculates the best paths to each node and then sends via source routing to each mobile node a message that contains the set of up to K parents that the node should use. An abstract version of the protocol actions is shown in Figure 7.

```
Actions at mobile node $j$:
Rcv_Parent_Set ($i$, $ParentSet$) $\rightarrow$
    if ($j == i$)
        $j.ParentSet$ := $ParentSet$;

($j.NT.Event = TRUE$) $\rightarrow$
    Send_to_LSAP ($j, j.NT$);

Actions at LSAP $l$:
Rcv_NT_From_node ($i$, $i.NT$) $\rightarrow$
    $l.ActiveRegister$:= $l.ActiveRegister \cup \{i\}$;
    $l.NetworkMap$ := $l.NetworkMap \cup \{i.NT\}$

($l.CorrectionTimer = TRUE$) $\rightarrow$
    for $i \in ActiveRegister$
        if ($Count(i.Parents) = 0$)
            Send_to_node ($i$,$CalculateParentSet(i)$);
```

Fig. 7. AER protocol actions

To maintain connectivity, the LSAP continuously monitors the network map to check if any of the nodes are in danger of losing all their K parents, i.e., if they have (say) only one remaining parent. In this case, the LSAP calculates a new set of parents for each such node and sends to it a parent-set update. Thus the core of the protocol boils down to efficiently collecting a reliable and consistent network-map.

Recall that a snapshot is *consistent* if it reflects the *events* in the same global order in which it happens in the network, where an event is the addition or loss of a link at that node. A simple way of maintaining a consistent snapshot would be to make each node report every single *event* to the LSAP. A more energy-efficient alternative is to exploit the knowledge of mobility at a node, and trigger link updates when the mobility state of a node changes. More precisely, the specific 'mobility events' that will trigger an event-report to the LSAP are:

- If a node changes its state from mobile to static.
- When in a mobile state a node has lost one or more parents.
- If a node in the mobile state, then after every D_{avg} distance (the average distance to move out of a nodes neighborhood) that it travels.

As we show in Theorem 1 below, if nodes report their neighbor table when any of the above events happen, snapshot consistency is maintained.

AER has two exception cases to handle. First, when a node loses its last parent before its new parent-set arrives, AER correctness remains unaffected since the parent update messages will reach the node, assuming there exists a path from the LSAP to the node. But if the node wishes to send a message to the LSAP in this case, it must execute a 'node-join' procedure, where it locally searches for K-parents. This procedure is, however, costly and AER is designed to avoid this from occurring in the common case.

The second case is that of region initialization, where none of the nodes have any parents. In this case, the LSAP can initiate a wave computation (similar to the one in MDSDV protocol we will describe shortly), using which the nodes can initialize their parent-sets. Note that in the scenario where nodes join the region one-by-one, they would end up executing the node-join procedure described above.

Discussion. Recall that we allow nodes with hop-counts equal to that of the node to be its parents while building the multi-path tree. This is motivated for AER by the following simple reason: nodes that are one hop away from the LSAP will have only one path to the LSAP without any sibling-parent (a parent with an equal hop-count) messaging, consequently making the one hop nodes the weakest links in the tree and violating the braided-multipath criteria. But now we have to avoid infinitely long paths on sibling links, so a node is allowed to forward a packet to a sibling-parent only once per hop-count. That is, if a packet travels on sibling link, which is realized by reserving a single bit in the routing header for this purpose, it can be forwarded only to a proper-parent (a parent with a lesser hop-count) in the next hop, thus guaranteeing convergence of routing. This also restricts the maximum number of hops traveled by a packet to twice the length of a shortest path, which helps path reliability.

D. Analysis of AER

Here, we analyze the correctness of AER and show that its snapshots are consistent. Next we analyze its latency constraints, and finally we derive an expression for the routing message cost.

Theorem 1 (Snapshot Consistency): *The snapshot maintained by AER using the neighbor table updates triggered by mobility events is consistent.*

Proof: Recall that we assume that links are symmetric, and that nodes can determine locally whether or not they are moving. For this proof, we also assume that the network has a global clock and that messages are delivered reliably. We first note that, hypothetically, if every change in the network is reported to the LSAP reliably and if the updates are processed in a globally increasing order of time, the snapshot would trivially be consistent.

We next observe that a link changes only when one or more nodes move. The movement of a node j can cause changes in j's neighbor table and also in j's neighbor's tables. In this case, since links are symmetric, it suffices that only j report the changes. Further, as neighbor tables are timestamped with a global clock, it suffices that events at the LSAP be processed strictly in a increasing order of time. An event (i.e., information about a link) that is delivered in a delayed manner can be discarded if a more recent update is available about the event. Since, every event can be conveyed by updates from the mobile nodes alone (albeit with a delay) and every event can be processed in with a globally increasing order of time, the snapshot constrcuted is consistent. $\square$

Even though it might seem at first glance that the assumptions made in the proof are severe, they can be approximated quite reasonably in current network. The first assumption of symmetric links can be approximated by considering only links which are above a certain receiver power level (commonly called inner-band links), thereby making sure that both the nodes see each other. Time synchronization protocols have been shown to work efficiently in wireless networks even in cases where network partition occurs and nodes move [7], so a global clock can be used. The assumption of detecting node movement locally can be accomplished by either continuous monitoring the neighbor tables for changes or by using a sensor, such as an accelerometer.

Lemma 1 (Snapshot Timeliness): *For AER to guarantee that each node has a path to the LSAP, the sum of the maximum times to (i) detect an event (t_d), (ii) report the event to the LSAP (Δ), and (iii) get a fix back from the LSAP (δ) is less the minimum time for a node to lose all its parents ($t(1)_{min}$).*

In AER a mobile node receives a fix when it has lost $K-1$ parents, that is, when it has just one parent. Hence, to guarantee a path at all times, the time for the node to detect the loss of the last but one parent, t_d, the time to report the event, Δ, and the time to receive a fix δ, should be less than the mininum time to lose the last parent $t(1)_{min}$. That is,

$$t_d \; + \; \delta \; + \; \Delta \; < \; t(1)_{min} \tag{3}$$

Δ and δ are network delays in the order of milliseconds and are much smaller than the second term of $t(1)_{min}$ given by Equation 2.

To calculate t_d, note that the mobility of node j can cause node j to lose a parent or j's child to lose a parent (i.e j). If j loses a parent it will detect it immediately. If j is the parent of another node and that link is lost, j will detect this within D_{avg}/v time after it starts moving. Thus, the maximum time for detecting a event, $max(t_d)$, is at most D_{avg}/v.

Analysis of routing cost. We analyze routing cost in terms of the number of messages sent by the entire system per second; we take a system view, rather than a per-node view, since the nodes co-operate in forwarding the routing messages. We assume in the rest of the paper that apart from the routing messages each mobile node broadcasts a heart beat message periodically (once every t_{hb} seconds), to assist in neighborhood discovery process.

We call the time in which each node will require at least one fix as the correction period t_c and measure the routing cost $C(N)$ for fixing each node once; $C(N)/t_c$ thus gives us the routing cost per unit time. Note that t_c is not a protocol parameter, but a measured time, which depends on the mobility rate of the nodes. In otherwords $1/t_c$ gives the frequency at which the protocol collects snapshots

We observe that the average time for each node to have been fixed at least once by LSAP is upper bounded by $t(K-1)_{avg}$, since the protocol fixes the nodes when they have just parent. Additionally, the nodes would take Δ to report the event and another δ to receive the fix. Thus, the total time in which every node receives a fix is

$$t_c \; = \; t(K-1)_{avg} + \Delta + \delta \tag{4}$$

Cost for heartbeat messages: Each of the N nodes send a heartbeat message once every t_{hb}, so the total number of heartbeat messages sent in time t_c is at most $\frac{N * t_c}{t_{hb}}$.

Cost for building the network map: In time t_c, each node would have lost at least $K-1$ parents, so each node would have reported $K-1$ events. On average, each message travels over $H/2$ hops, where H is the maximum hops in the network, thus number of messages sent to build the snapshot is $\frac{(K-1)NH}{2}$.

Cost for fixing the nodes: Each node receives a fix in time t_c, where each message travel over a average of $H/2$ hops. So the total number of 'route fix' messages are $\frac{NH}{2}$.

Total routing cost per second: Adding these three costs and dividing by t_c we get the total routing cost per second cost to be

$$NH(\frac{(K-1)}{2t_c} + \frac{1}{2t_c} + \frac{1}{t_{hb}H})$$

$$= \; NH(\frac{K}{2t_c} + \frac{1}{t_{hb}H}) \tag{5}$$

IV. Comparison with classical protocols

We have implemented asymmetric multipath versions of three well known routing protocols, which we describe here. The three differ primarily in the way they build the node-to-LSAP trees; they use the same approach for building the LSAP-to-node paths and in each nodes periodically sends their neighbor table to the LSAP.

```
Actions at node $j$:
**Rcv_Tree_Refresh**($i.HC$, $i.SeqNo$) $\rightarrow$
    if ($j.SeqNo < i.SeqNo$)
        $j.SeqNo$ := $i.SeqNo$;
        $j.HC$ := $i.HC$+1;
        $j.PrimaryParent$ := $i$;
        $j.OtherParents$ := $NULL$;
    elseif($j.SeqNo == i.SeqNo$ &&
                $j.HC >= i.HC$)
        $j.OtherParents$ := $j.OtherParents \cup \{i\}$;
    fi
    Broadcast_Tree_Refresh($j.HC$, $j.SeqNo$);

Actions at LSAP node $l$:
**(CorrectionTimer==TRUE)** $\rightarrow$
    $l.HC$ :=0; $l.SeqNo$ := $l.SeqNo$ +1;
    Broadcast_Tree_Refresh($j.HC$, $j.SeqNo$);
```

Fig. 8. Actions for Multipath DSDV

A. Multipath DSDV (MDSDV) Protocol

The DSDV [12] protocol implements Bellman-Ford algorithm, but tags route entries with sequence numbers to deal with routing loops and the count-to-infinity problem inherent in the approach. To optimize DSDV for a single destination and for multiple parents, we combine DSDV with wave propagation, initiated at the LSAP. Once every correction period, t_c, the LSAP triggers rebuilding of the tree by broadcasting a special 'tree-refresh' message, which consists of the 'hop-count' to the LSAP and a sequence number. Each node that receives a new 'tree-refresh' message (as determined by the sequence number) sets the source node as its primary parent and rebroadcasts the message with the 'hop-count' incremented by one. Nodes continue to listen for other potential route advertisements with hop counts less than or equal to their current hop count, but with the latest sequence number. Messages with older sequence numbers are discarded. The routing tree construction is essentially a diffusing wave computation initiated at the LSAP periodically during which each node in the network chooses K parents. To guarantee that a path exists from every node to the LSAP at all times, the LSAP should trigger the tree construction before any node lose all its parents.

To construct the LSAP-to-node paths after the tree construction, the nodes send their 'neighbor table' and their parent list to the LSAP, from which the 'network map and 'Active Register' are constructed. Fig 8

Proposition 1 (Correction Period): *To guarantee that a path to the LSAP exists at all time from each reachable nodes, the MDSDV time difference between the end of two consecutive tree-refresh cycles, t_c, should be less than the minimum time for a node to lose **all its parents** minus the time taken to build the tree t_{RT}.* That is,

$$t_c < t(K)_{min} - t_{RT} \tag{6}$$

MDSDV routing cost: Analysis similar to one in Section III-C shows that MDSDV uses $N\frac{t_c}{t_{hb}}$ messages to build node-to-LSAP paths (the tree building messages also serve as heart beat messages) and $\frac{NH}{2}$ messages to build LSAP-to-node paths, resulting in a total per second cost of $NH(\frac{1}{2t_c} + \frac{1}{t_{hb}H})$.

B. Multipath OSPF (MOSPF) Protocol

Recall that in OSPF [2] each node distributes its neighbor table to the entire network periodically and each node builds its paths based on the entire link-state topology. We adapt OSPF so that instead of all nodes building their own routes, the LSAP builds the routes (i.e., selects the K best parents) and distributes them to the mobile nodes. This is very similar to the AER, the difference being that while AER builds its network map in a incremental manner through 'mobility events', MOSPF builds its network map in a periodic and synchronized manner (and hence the snapshots are consistent). Once the network map is built, the LSAP finds nodes that are at risk of losing all parents and proceeds to fix them with a new parent set. Fig 9 shows the actions for Multipath OSPF protocol.

Proposition 2: *To guarantee that a path to the LSAP exists at all time from each reachable nodes, the MOSPF time difference between two consecutive correction cycles, t_c, should be less than the minimum time for a node to*

```
Actions at Mobile node $j$:
**Rcv_Parent_Set**($i$, $ParentSet$) $\rightarrow$
    if ($j == i$)
        $j.ParentSet$ := $ParentSet$;

($j.GloablSnapshotTimer == TRUE$) $\rightarrow$
    Send_to_LSAP($j, j.NT$);

Actions at LSAP node $l$:
**Rcv_NT_From_MN**($i$, $i.NT$) $\rightarrow$
    $l.ActiveRegister$: = $l.ActiveRegister \cup \{i\}$
    $l.NetworkMap$ := $l.NetworkMap \cup \{i.NT\}$

($l.CorrectionTimer == TRUE$) $\rightarrow$
    for $\forall\ i \in ActiveRegister$
        if($Count(i.Parents) < K$)
            Send_to_MN($i$,$CalculateParentSet(i)$);
```

Fig. 9. Actions for Multipath OSPF Protocol

*lose **all its parents** minus the maximum network delay (δ) to fix the node*. That is, $t_c < t(K)_{min} - \delta$

MOSPF routing cost: In time t_c, MOSPF spends $\frac{Nt_c}{t_{hb}}$ for heartbeat messages, $\frac{NH}{2}$ messages for building the network map and $\frac{NHt_c}{2t(K)_{avg}}$ messages for fixing routes of mobile nodes, resulting in a total routing cost per second of $NH(\frac{1}{2t(K)_{avg}} + \frac{1}{2t_c} + \frac{1}{t_{hb}H})$.

C. Multipath Bellman-Ford (MBF) Protocol

```
Actions at node $j$:
**Rcv_grad_adv**($i.grad$, $i.WN$) $\rightarrow$
    $j.NT$ := $j.NT \cup (i.grad, i.WN)$;

**(CorrectionTimer==TRUE)** $\rightarrow$
    $j.grad$ := min($j.NT.grad$)+1;
    $j.WN$ := count($j.NT.WN > 0$);
    if($j.WN == 0$)
        $j.grad$ := $NULL$;
    fi
    Broadcast_grad_adv($j.grad$, $J.WN$);
    j.ParentSet = NULL;
    for $\forall\ i\ \in\ j.NT$
        if ($j.NT.i.grad \leq j.grad$)
            $j.ParentSet = j.ParentSet\ \cup\ i$;
        fi
```

Fig. 10. Actions for Multipath Bellman-Ford Protocol

In order to compare the asymmetric approach with a pure distributed protocol, without focusing attention on optimizations, we implemented a multi-path version of the Bellman-Ford algorithm. In MBF, once every correction period, t_c, each node calculates its gradient, which is the minimum of the gradients of its neighbors with a non-zero 'witness number', incremented by 1. The 'witness-number' of a node is the number of neighbors which have a path to the LSAP and have a gradient less than that of the current node. In a typical connected network, each node should have at least one node with a non-zero witness-number and a gradient less than the current node. After calculating its gradient, the node broadcasts its gradient and witness number. The neighbor with the minimum gradient becomes the primary parent and the node looks for $K-1$ other neighbors with less than or equal gradients. Our use of the witness-number lets us deal with the problem of network disconnection and the 'count-to-infinity' problem and achieves stabilization of the gradient quickly (within max-hop-count rounds). Fig 10 shows the actions for the Multipath Bellman-Ford Protocol.

Proposition 3: *To guarantee that a path to the LSAP exists at all time from each reachable nodes, the MBF*

Algorithm	Time Constraint
AER	$t_c \ = \ t(K-1)_{avg} \ +\Delta+\delta$
MDSDV	$t_c \ < \ t(K)_{min} - t_{RT}$
MOSPF	$t_c \ < \ t(K)_{min} \ - \ \delta$
MBF	$t_c \ < \ \dfrac{t(K)_{min}}{2}$

TABLE I
SUMMARY OF CORRECTION PERIOD FOR DIFFERENT ALGORITHMS

Algorithm	Cost
AER	$NH(\dfrac{K}{2t_c} + \dfrac{1}{t_{hb}H})$
MDSDV	$NH(\dfrac{1}{2t_c} + \dfrac{1}{t_{hb}H})$
MOSPF	$NH(\dfrac{1}{2t(K)_{avg}} + \dfrac{1}{2t_c} + \dfrac{1}{t_{hb}H})$
MBF	$NH(\dfrac{1}{2t_c} + \dfrac{1}{t_cH})$

TABLE II
SUMMARY OF COST FOR DIFFERENT ALGORITHMS

time difference between two consecutive correction cycles (or gradient messages), t_c, should be less than half the minimum time for a node to ***lose all its parents***. That is, $t_c \ < \ \dfrac{t(K)_{min}}{2}$.

MBF routing cost: In time t_c, MBF expends N messages for building node-to-LSAP paths (which also serve as heartbeat messages) and $\frac{NH}{2}$ messages for building the LSAP-to-node paths, resulting in a total routing cost per second of $NH(\dfrac{1}{2t_c} + \dfrac{1}{t_cH})$.

D. Analytical Comparison of Protocols

Table I summarizes the time constraints for the protocols to guarantee a path from all nodes to the LSAP. Table II summarizes the routing cost per second to satisfy the time constraints; i.e., to guarantee path availability. To illustrate the performance of these protocols, consider K=3 and $N = 30$. We can calculate the routing costs for different probabilities of movement, with values for $t(K)_{min}$ and $t(K)_{avg}$ obtained either via Matlab analysis (as explained in Section II) or obtained via simulation (as explained in Section V); we show later that these two value sets are rather similar. Figure 11 shows the latter costs, again assuming there is no routing loss. We see that AER has the least cost under all mobility conditions and also has the least growth rate. MDSDV and MOSPF have similar performance, with MDSDV being simpler to design. MBF cost grows quickly and has highest cost of the four.

The graph also shows that while AER adapts efficiently to the rate of mobility with the least cost, the routing cost for the other three increases because the correction period t_c need to be adapted to deal with the rate of mobility. If the correction period is not adapted to the mobility rate, the path availability will start decreasing.

V. SIMULATION RESULTS

In this section, we first validate the mobility model results of Section II and then evaluate the performance of the protocols with respect to tree quality and routing cost, via simulations.

A. Simulation Model and Experiment Setup

We have developed a discrete event *Asymmetric Mobile network simulator (AMns)* [1] in Matlab that can:

- Allow different mobility models to be plugged in.
- Simulate message passing, both broadcast and unicast.
- Drop packets based on a link error rate.
- Simulate node failures, both transient and permanent.
- Accurately reproduced simulation results by using the same random seed(s) in initialization.

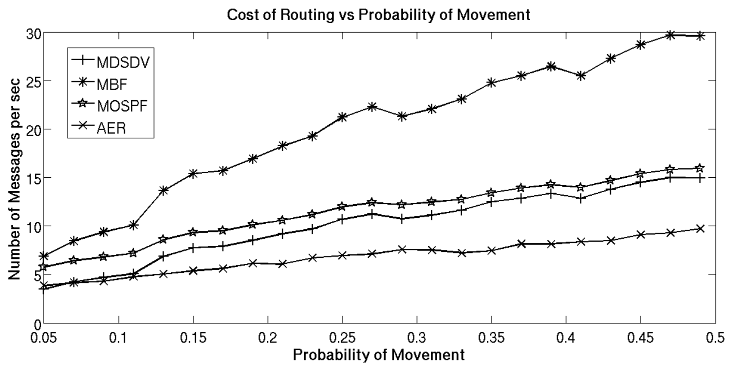

Fig. 11. Cost of routing versus probability of movement

Parameter	Value
Number of nodes, N	30
Radius of transmission, R	20m
Number of parents, K	3
Velocity of movement, v	1.5 m/s

TABLE III
SIMULATION PARAMETERS

Our simulator does not however simulate packet collision.

Using the simulator, we simulated a 100 x 100 m region with an LSAP situated at its center. The mobility model for nodes is the one described in Section II-C. The communication model for nodes is the unit disk model: the unit represents the transmission radius R and the use of the disk implies link symmetry. The default values of parameters are as shown in Table III, unless otherwise stated. Each point in our simulation graphs represents the average of 50 different random instances of simulation.

B. *Validation of the Mobility Model*

To validate Property 1, we conducted a simulation where a multi-parent tree was built using MDSDV at the beginning of the simulation and the tree was then allowed to decay –without ever being rebuilt– as the nodes started moving. We measured the time taken by the first node to disconnect from the tree and the average time taken by node to disconnect, i.e., time taken for 50% of the nodes to disconnect, for different values of N and K. The results are shown in Figure 12.

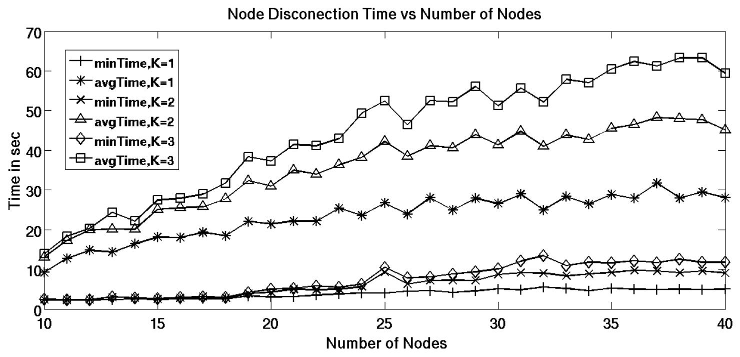

Fig. 12. Minimum and average time for node disconnection versus node density

The figure shows that as K increases the minimum time changes only slightly but the average time increases substantially and particularly for $K = 3$. This is consistent with the analytically derived graph of Fig 5. That graph predicts that the average time starts to flatten out around K=7, while the simulation finds that the value starts to saturate around K=3, which is why our graph is plotted only for K in 1..3 and we use $K = 3$ for comparing the multi-path protocol performance.

We also repeated the same simulations keeping N constant and varying the probability of movement. Figure 13 shows the average and minimum time to lose all parents (for K=3), along with the values predicted by Equation 2 from Section II-D. From the figure we see that our theoretical and simulation values for $t(K)$ are in agreement, especially for $t(K)_{min}$.

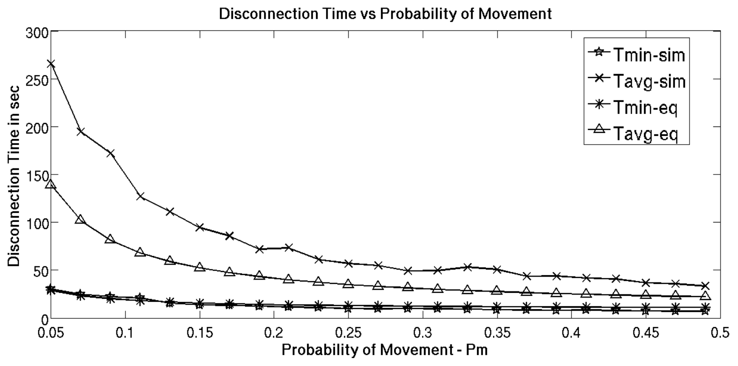

Fig. 13. Comparison of analytical and simulation results for time to lose all parent $t(K)$

Next, we question whether the method used for selecting a parent from among the neighbors affects the disconnection time. We used two different methods to select the parents: (i) random parent selection as in MDSDV and (ii) a location based balanced tree construction, which chooses parents to be as geographically separated as much as possible, the logic being that if the parents are well separated that will offset the movement of child in any direction. Figure 14 shows a plot of minimum time for disconnection for the two algorithms, the random tree being marked as 'RT' and balanced tree being marked as 'BT'. From the figure we see that the performance of the simpler approach random selection, which eschews hard-to-satisfy assumptions of localization, is adequate as compared to balanced trees. What matters more is the choice of K.

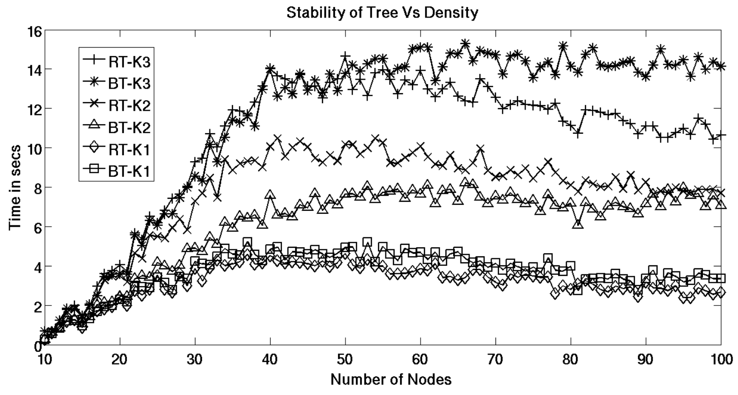

Fig. 14. Minimum time for node disconnection versus node density for a K-parent random tree and balanced tree

C. Quality of Multi-Path Trees

We evaluate the performance of AER in contrast to the multipath versions of DSDV, OSPF and Bellman-Ford algorithms. To compare with a single parent protocol, we also simulate Bellman-Ford (represented with the legend 'BF' in the plots).

Figure 15 shows quality of the multi-path tree, in terms of the path non-availability in the network, i.e., the percentage of nodes without a path to the LSAP, with respect to the correction period. From the figure, we see that AER performs fine overall and is independent of correction time as it is not a periodic protocol. MOSPF is slightly better than AER when the correction period is very short, since it collects better snapshots than AER in this region, and slightly worse as the correction period increases. MDSDV, MBF and BF all show a similar trend, getting adversely affected as the correction period increases. DSDV is affected the worst since it is a single parent protocol.

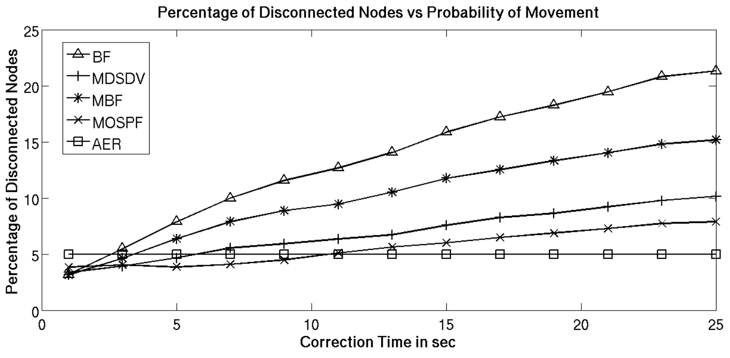

Fig. 15. Quality of the tree versus time period of correction

Figure 16 shows the percentage of node disconnections versus the probability of movement in the static state. From the figure, we see that MBF and BF, which are completely periodic and distributed, perform poorly as the probability of movement increases. MOSPF and MDSDV perform well till the probability of mobility is about 0.35, but then slowly start performing worse; a correction time of 10 seconds seems to satisfy their timing requirements (as given in Table I) till the probability of movement is 0.35 and hence the performance remains constant. AER performance surprisingly improves as the probability of movement increases. The reason for this result is that, as the movement increases, the rate at which snapshots are taken at the LSAP increases and a better snapshot leads to better route updates; of course this improvement comes with an increase in routing cost..

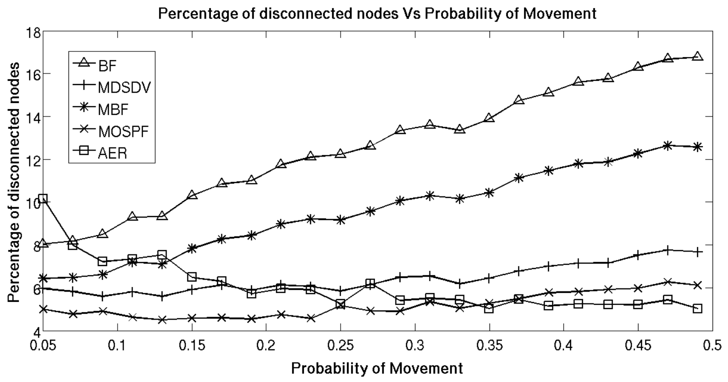

Fig. 16. Quality of the tree versus probability of movement

D. Cost of Routing

Figure 17 shows the cost of routing, i.e., the number of messages send per second in the network for the three algorithms for different values of the correction period, t_c. The probability of movement is kept constant at a value of 0.25. From the figure, we see that all periodic protocols are substantially influenced by the time period. AER being event driven does not show variation with the time period and has the least cost, as was predicted by the cost equations. We also see that the routing cost of the periodic protocols goes down with respect to correction time, albeit with a decrease in the number of connected nodes, as shown in the performance plot in Figure 15. This decrease in cost is a potentially misleading artifact of our metric: as the time period increases, the number of nodes that are disconnected from the LSAP increases and hence the overall cost of the connected nodes sending neighbor table updates over multiple hops to the LSAP goes down.

Figure 18 shows routing cost versus the probability of movement. The correction time, t_c, is kept constant at 10 sec (albeit this is not material for AER). From this figure, we again see the trend of decreasing cost at all protocols

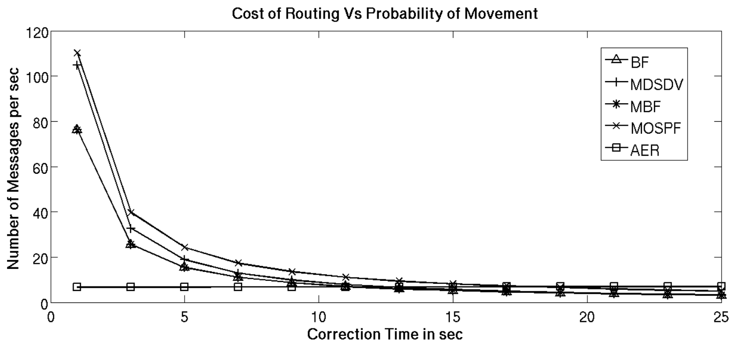

Fig. 17. Cost of routing versus time period of correction

other than AER as the probability of movement increases. The explanation is again that while AER successively adapts to mobility and maintains (actually increases) the level of connectivity, in the other protocols the number of disconnected nodes increase and so the cost artifactually decreases. And although the cost of AER increases with probability of movement, since AER snapshot frequency increases and it fixes more nodes which are in 'imminent' danger of disconnection at higher mobility, this increased cost is still less than the cost of all other protocols and the increased traffic does not overwhelm the availability of bandwidth.

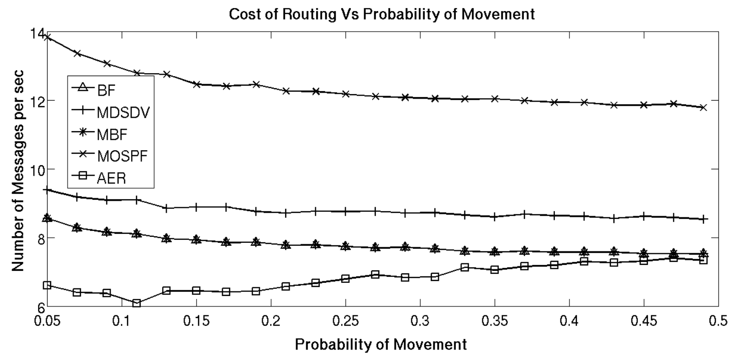

Fig. 18. Cost of routing versus probability of movement

In summary, then, we find that AER has rather low cost compared with the other protocols in the constrained mobility region while guaranteeing equal or connectivity. As the mobility increases, AER maintains high connectivity with a increase in cost proportional to the mobility, but this is still cost-effective in comparison.

VI. Related Work

Many reactive and proactive MANET protocols have been researched, with DSDV, Adhoc On-demand Distance Vector, and Dynamic Source Routing being most popular. One might argue that reactive protocols are preferable if all nodes are mobile with relatively high speeds, as compared to the constrained mobility that we consider in this paper. In contrast, structure maintenance becomes feasible in constrained mobility networks, especially as we can reduce the route maintenance cost by considering the multi-path approach.

Increasingly, attention is being paid to adapt the work on MANET protocols for resource constrained devices, specifically to incorporate simpler protocols with lesser storage and energy requirements. As one data point, DSDV stores several bytes of information per destination in the network and does not scale well to the thousands-of-nodes networks being deployed currently. TinyAODV [3] is one such effort to port the AODV to resource constrained devices such as sensor nodes. It is a minimalist implementation for devices running TinyOS which seems to target static networks, nonetheless the implementation still has thousands of lines of code and requires a significant amount of device space. In comparison, the protocols we considered are considerably more frugal. Each node in our protocols takes 2 bytes (ID and hop count) of information for each of the K (typically 3) parents, which means reserving 6 bytes per LSAP in the routing table. Previous work related to multipath routing has been already discussed in Section III-B.

VII. Conclusion and Future Work

AER performs mobility-aware, asymmetric, multi-path proactive routing. It is well suited to constrained mobility networks, where its spontaneous adaptation to increase in mobility preserves high availability with a sublinear growth in cost. Adaptations of wellknown alternative proactive protocols to accommodate multi-parents do not scale as well for most of the constrained mobility region. This is only partly because our adaptations are simplified to be suitable to resource constrained mobile devices, but the larger reason is that, to avoid increasing node disconnections as networks scale in size, one has to increase the rate of maintenance significantly.

We are currently in the process of deploying the AER protocol on a cellphone based campus area sensor network that is under developement and in the near future, we will be studying the perfomance of protocol in real life deployment.

References

[1] Asymmetric mobile network simulator. http://www.cse.ohio-state.edu/~sridhara/AMns/.

[2] Open shortest path first. http://www.ietf.org/html.charters/ospf-charter.html.

[3] Tinyos implementation of tinyaodv. http://cvs.sourcefourge.net/viewcvs.py/tinyos/tinyos-1.x/contrib/hsn/.

[4] F. Bai, N. Sadagopan, and A. Helmy. Impact: Investigation of mobile-user patterns across university campuses using wlan trace analysis, usc technical report, July 2005.

[5] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[6] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. In *Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'01), October 4-5, 2001, Long Beach, CA, USA*, pages 295–298. ACM, October 2001.

[7] T. Herman and C. Zhang. Stabilizing clock synchronization for wireless sensor networks. In *Symposium on Stabilization, Security and Safety, SSS*, November 2006.

[8] W. Hsu, K. Merchant, H. Shu, C. Hsu, and A. Helmy. Weighted waypoint mobility model and its impact on ad hoc networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(1):59–63, 2005.

[9] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[10] S. Lee and M. Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks. In *Proceedings of the IEEE ICC*, pages 3201–3205, 2001.

[11] M. K. Marina and S. R. Das. On demand multipath distance vector routing in ad hoc networks. In *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, pages 14–23, 2001.

[12] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[13] S. Vutukury and J. J. Garcia-Luna-Aceves. MDVA: A distance-vector multipath routing protocol. In *INFOCOM*, pages 557–564, 2001.

[14] L. Wang, L. Zhang, Y. Shu, and M. Dong. Multipath source routing in wireless ad hoc networks. In *Proceedings of Canadian Conference on Electrical and Computer Engineering, vol. 1,*, pages 295–298. ACM, March 2000.

[15] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful, 2003.