Whirlpool Routing for Mobility

Jung Woo Lee
Stanford University
Stanford, CA USA
jungwoo.lee@stanford.edu

Branislav Kusy
CSIRO ICT Centre
Brisbane, Australia
branislav.kusy@gmail.com

Tahir Azim
Stanford University
Stanford, CA USA
tazim@cs.stanford.edu

Basem Shihada
KAUST
Thuwal, Saudi Arabia
basem.shihada@kaust.edu.sa

Philip Levis
Stanford University
Stanford, CA USA
pal@cs.stanford.edu

ABSTRACT

We present the Whirlpool Routing Protocol (WARP), which efficiently routes data to a node moving within a static mesh. The key insight in WARP's design is that data traffic can use an existing routing gradient to efficiently probe the topology, repair the routing gradient, and communicate these repairs to nearby nodes.

Using simulation, controlled testbeds, and real mobility experiments, we find that using the data plane for topology maintenance is highly effective due to the incremental nature of mobility updates. WARP leverages the fact that converging flows at a destination make the destination have the region of highest traffic. We provide a theoretical basis for WARP's behavior, defining an "update area" in which the topology must adjust when a destination moves. As long as packets arrive at a destination before it moves outside of the update area, WARP can repair the topology using the data plane. Compared to existing protocols, such as DYMO and HYPER, WARP's packet drop rate is up to 90% lower while sending up to 90% fewer packets.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Mobile Routing, Sensor Network, Collection Protocol

1. INTRODUCTION

Unlike their wired brethren, wireless nodes can move freely and untethered. This flexibility introduces significant complications to networking. While the past decade of research on mobile routing has generated a large number of elegant and intellectually interesting protocols and proposals [13, 25, 17], making them work well in real networks has remained frustratingly difficult [8]. In practice, the most notable successful deployments of mobile ad-hoc protocols involve no mobility at all! Examples include variants of dynamic source routing (DSR) in MIT's Roofnet [8] and Meraki [4], as well as extensions to OLSR in Athens [1], Berlin [2], and Leipzig [3].

This paper seeks to take a next step towards bridging the gulf between algorithms and practice in mobile wireless routing. It examines a limited form of mobility: routing data to a mobile node as it moves through a static wireless mesh infrastructure. Towards this end, it presents the Whirlpool Routing Protocol (WARP).

A broad range of wireless sensor network applications motivate WARP's design considerations. Vineyard workers [7], biologists [23], geologists [30], engineers [18], and tracking robots [28] all use sensor data in real-time while operating in remote and sparsely populated areas. These applications rely on self-built infrastructures, which need to minimize cost and power requirements: routing to a stationary sink and relying on an IP infrastructure to communicate to a mobile user is unattractive. Consequently, timely data collection to a mobile sink is an important system service, driven by interests in both the scientific community and industry.

The key insight behind WARP is that, when a destination moves, the existing distance vector tree can quickly find its new location. While a particular one-hop neighbor may no longer have connectivity, at least some other such neighbors still do. WARP uses the existing topology to search around the destination's old location for nodes that still have routes: when it finds such a node, it quickly repairs the local topology.

WARP builds on recent work in efficient distance vector protocols [15] and the long history of research on local repair [9, 20, 27]. Specifically, WARP extends these algorithms by speculatively routing data packets when it detects the destination has moved rather than triggering a discovery operation. Speculative routing sends packets along a spiral trajectory around the last known position of the destination. These spirals are based on the existing distance vector gradient. They require neither geographic information nor controlling how a node moves, yet can quickly and efficiently find routes. When a destination is stationary, WARP defaults to its underlying distance vector protocol, imposing no overhead.

The rest of this paper presents WARP's design (Section 2), provides theoretical bounds on its supported mobility (Section 3), describes a TinyOS-based wireless sensor network implementation (Section 4), and evaluates the implementation in simulation, controlled testbeds, and real mobility experiments (Section 6). The paper concludes with a discussion of WARP's relationship to the long history of mobile wireless routing research (Section 7) and future work (Section 8).

MobiHoc'10, September 20–24, 2010, Chicago, Illinois, USA.

2. THE WHIRLPOOL ALGORITHM

This section overviews WARP's design. It describes four mechanisms WARP uses for routing efficiently to mobile destination: fast mobility detection, speculative routing, local repair, and in-band signaling. It walks through an example of WARP's operation.

2.1 Overview

As a routing protocol, WARP's responsibility is to determine a packet's next hop to a destination. It provides a best-effort datagram service. WARP is a distance vector protocol: nodes maintain a measure of "distance" to a destination. This builds a routing tree around a destination. In the context of this paper, we only consider a single destination, as this is the typical sensor network traffic pattern. Generalizing WARP to multiple destinations simply involves using its mechanisms on each destination's distance vector.

When a destination is stationary, WARP operates as a standard routing protocol. Nodes estimate link costs, compute route costs, and send packets along the minimum cost path to a destination. WARP is a reactive protocol, as it changes how a distance vector protocol reacts when a node detects a destination has moved. Rather than flood route requests or emit other control traffic, WARP uses data packets to probe the network topology, using *whirlpool routing* to quickly and efficiently find the destination.

By default, a WARP destination sends frequent beacons to advertise its presence to nearby nodes. However, as this can have a high cost, WARP suppresses these beacons when unnecessary. Whenever a destination receives a data packet, it suppresses its next beacons. When there is active data traffic, WARP uses this traffic to validate the topology. In the absence of data traffic or routing failure, WARP falls back on rapid beacons.

These *spiral packets* use the existing routing gradient to "whirlpool" around the old location of the destination in order to pass near its new location. When a destination overhears a spiral packet, it immediately responds with a beacon. Receivers hearing a beacon exit the whirlpooling state and quickly inform their neighbors that they have a new route, making nearby nodes reconfigure their routing tables and stop whirlpooling as well. This simple epidemic of route updates causes nodes in the update area – the area whose distance vectors need updates – come to quickly reach a consistent view of the topology.

WARP's spirals require neither global nor geographic information, as the existing distance vector tree contains useful network topology information. Figure 1 shows an example of how this works. Intuitively, sending packets to nodes with a similar cost in the tree forms a path around the old position. This path might not be a circuit, but nonetheless it tries to cover the space around the prior position, maximizing the chances of covering the update area and finding the new position.

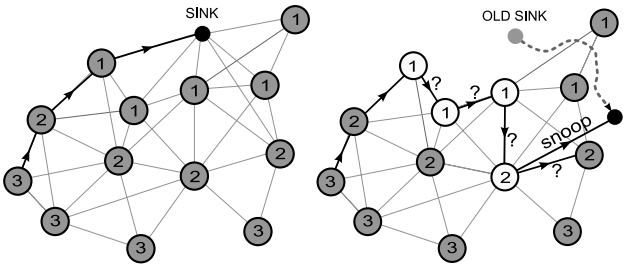
2.2 Fast Mobility Detection

WARP uses link layer reliability to quickly and reactively detect mobility. When a link to the destination fails, a node assumes the destination has moved and begins speculative routing.

2.3 Speculative Routing

WARP uses speculative routing to proactively search for a destination's new location. This section first explains speculative routing in terms of an idealized topology, then presents the algorithm WARP uses in practice.

Consider Figure 2, which makes the idealized assumption that radios have a fixed radio range r. In this network, nodes around the old position S forms a set of concentric rings $R_1, R_2, \ldots, R_n$ with radii $r, 2r, \ldots, nr$. The centers of the rings form circles with radii



(a) Normal collection mode. (b) Repairing mode.

Figure 1: When links to a destination break, nodes enter repairing state (white). Repair nodes speculatively route packets along the old gradient (question marks). When the destination overhears a data packet, it immediately initiates repair. Numbers show the routing gradient.

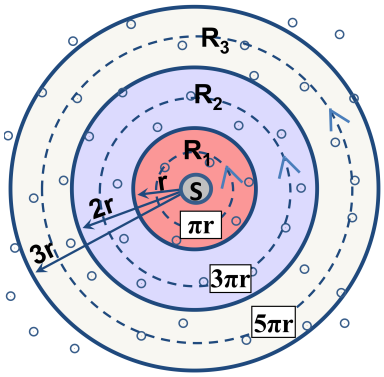


Figure 2: Uniformly distributed random topology. Nodes in the second and third rings route packets to children 3 and 5 times less likely than center nodes.

$\frac{1}{2}r, \frac{3}{2}r, \ldots, \frac{(2n-1)}{2}r$, and circumference $\pi r, 3\pi r, \ldots, (2n-1)\pi r$. These circles approximate spiral loops around the old position. Assuming one-hop distance is r, a space-filling trajectory must take at least $\lceil \pi \rceil = 4$ hops along R_1, $\lceil 3\pi \rceil = 10$ hops along R_2, and in general, $\lceil (2n-1)\pi \rceil$ hops along R_n.

These values are for a perfect space-filling trajectory that moves in a circle along the center of a ring. In practice, nodes do not have sufficient state to construct such a trajectory. In real network topologies, nodes do not have locations of other nodes, so they can not choose a proper next hop to follow the ideal trajectory and there are asymmetric as well as irregular communication links. Consequently, spirals may not be centered at the old location and their radii may not increase at all. WARP overcomes these challenges by increasing the radii probabilistically and by randomly selecting one of the available siblings (or children) to forward the data.

A speculatively routing WARP node picks a random node from the neighbors in its routing table who have similar costs and good link qualities. This is effectively a random walk along the gradient contour, which is less efficient than the perfect space-filling trajectory. Therefore, WARP overestimates the number of hops at the n-th level as $8n$ (space-filling spirals in real networks require $\approx (2n-1)\pi$ hops at the n-th level which $8n$ slightly overestimates). We use the following algorithm for speculative routing, where H is defined as the number of hops that a packet was forwarded in the spiral mode:

```
find n(≥ 1) such that Σ_{k=1}^{n-1} 8k < H < Σ_{k=1}^{n} 8k
rnd = random_integer() mod 8n;
if (rnd == 1)
  send to a child;
else
  send to a sibling;
```

Using this algorithm has an expectation of approximately one WARP node at a given radius forwarding packets to a child. As the destination can snoop on these packets, a space-filling spiral with this property has a high probability of finding the destination. Using a probabilistic and randomized approach means that WARP's behavior may not be theoretically optimal, but makes it robust to unforeseen edge cases and degenerate topologies, thus enabling it to work in practice.

2.4 Local Repair

When a speculatively routing node hears a beacon from the destination, it immediately updates its routing table and stops routing speculatively. In some cases this is a bad decision, as the link to the destination may be poor. However, nodes detect this through the underlying routing protocol's standard mechanisms and adapt to use other routes. Eventually some nodes will have good one-hop routes to the destination. The key property is that the node stops whirlpooling and begins to refine its route.

After a node finds a new route, it forwards data packets along the new route, breaking spirals. These packets inform neighbors that the sender has found a new route: neighbors in turn update their routing tables. Section 4 describes how nodes distinguish distinct mobility events in practice. Incremental updates with route refining make the routing tree to converge the minimum-cost tree over time.

2.5 In-band Signaling

WARP uses data traffic as active topology probes: they detect mobility, find a lost destination, and convey a new route. WARP requires four fields in the packet header: a spiral hopcount, a bit describing whether the packet is spiraling, the parent ID of the transmitter, and the routing gradient cost of the transmitter. As the parent ID (or a next hop node) and cost are typically fields in collection data packet headers, WARP only needs to add the spiraling bit and the spiral hopcount field.

Furthermore, WARP uses data packets to implicitly signal destination locations in-band. As Section 2.1 mentioned, a WARP destination suppresses its beacons if it receives data packets. Nodes overhearing those data packets assume that the destination is not far away, relying on mobility detection through their own data plane as needed. Therefore, a destination sends a beacon only when

- it has not received a data packet within the beacon period or
- it snoops a spiraling packet.

As we show in Section 6, this aggressive suppression causes WARP to send fewer control packets than CTP despite our implementation's use of a beacon timer with relatively small intervals such as 125ms or 2s.

3. MOBILITY SUPPORTED BY WARP

Higher data and beacon rates speed topology repair. This section derives bounds on the data rate required to sustain a routing tree as a function of the destination speed.

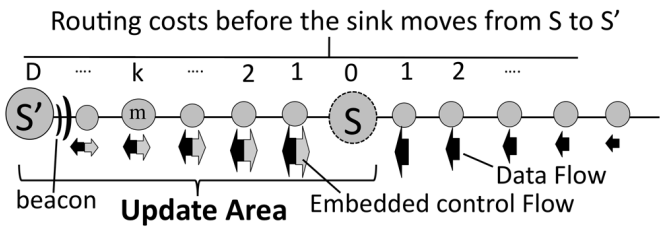


Figure 3: A line topology: the destination moves from S to S'. The update area is between S and S'. The flow of data and embedded control information is indicated by black and gray arrows, respectively. A larger arrow means higher traffic.

We analyze two different topologies under the same scenario: N nodes generate data at a constant rate (p packets/sec) and a destination moves at h hops/sec. One hop is the communication range of a node. This analysis makes two simplifying assumptions. First, it models mobility as a series of discrete steps, as the mobile node stays at each location for a certain time (*wait time* t), jumping to a new location $D = ht$ hops away. Second, it assumes there are no packet losses. We calculate routing cost in terms of hops.

Let the destination change its location from S to S'. A node m needs to be *repaired* if its next hop towards the destination is a node further away from S' than m. The *update area* is a space around the destination containing all nodes that need to be *repaired*. The basic trade-off is between the time required to *repair* nodes inside the *update area* and the mobility of the destination.

3.1 One Dimensional Case

We first consider a simple line topology in Figure 3. The update area includes nodes between the new and the old locations of the destination, including the old location. The following theorem derives a bound on the data rate required to support a given speed and wait time of the destination.

THEOREM 1: Let the destination S be at the center of the N nodes as in Figure 3. Given the beacon interval b, the speed of the destination h and wait time t, WARP supports the mobility if the data rate p satisfies the following inequality:

$$p > \frac{2(D-1)}{\{N-2(D-1)\}(t-b)}, \text{ where } D = ht.$$

PROOF: The routing state of the nodes located in the update area is repaired via beacons from the destination and subsequently transmitted data packets. The time that is required to wait for a control or data packet to be transmitted is the inverse of the outgoing control or data rate at a given node. Let P be the aggregate data rate of the network ($P = Np$). Consider node m that is k hops away from S (see Figure 3). Since S was a central node, the aggregate data rate that is pushed towards m is $\frac{P}{2} - kp$ and the outgoing rate at this node is $\frac{P}{2} - (k-1)p$. The time required to repair all nodes in the update area is then

$$\begin{aligned}
&b + \tfrac{1}{\frac{P}{2}-(D-2)p} + \cdots + \tfrac{1}{\frac{P}{2}-2p} + \tfrac{1}{\frac{P}{2}-p} + \tfrac{1}{\frac{P}{2}} \\
&< b + \tfrac{2}{p}\{\tfrac{1}{N} + \tfrac{1}{N-2*1} + \tfrac{1}{N-2*2} + \cdots + \tfrac{1}{N-2*(D-2)}\} \\
&< b + \tfrac{2}{p}\{\tfrac{D-1}{N-2(D-2)}\} < b + \tfrac{2}{p}\{\tfrac{D-1}{N-2(D-1)}\} < t \\
&\Leftrightarrow p > \tfrac{2(ht-1)}{\{N-2(ht-1)\}(t-b)}.
\end{aligned}$$

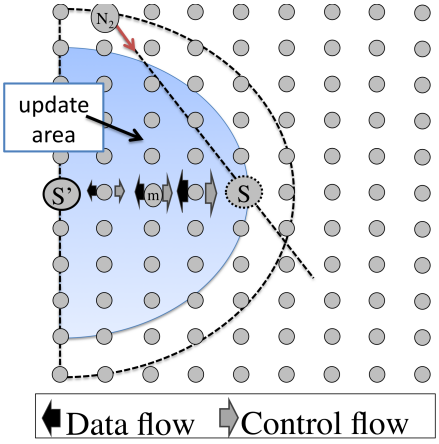


Figure 4: A grid topology. Data and embedded control flows between old and new locations of a destination node (S and S', respectively), represented by black and gray arrows, with sizes proportional to traffic. The gray half-disk denotes the update area.

3.2 Two Dimensional Case

We now consider a two dimensional topology with a uniform node distribution, as shown in Figure 4. We overestimate the update area by a half-disk centered at the new location of a destination with a radius equal to the distance between the new and the old location: For each node (e.g., node N_2) that is outside the update area, the next hop node towards S is closer to S' than N_2. This is because S is inside circle centered at S', with radius $|S'N_2|$ (the dotted half-circle shown in Figure 4).

THEOREM 2: Let the destination S be at the center of the grid topology. Given the beacon interval b, the speed of the destination h and wait time t, WARP supports mobility of S, if the data rate p satisfies the following inequality:

$$p > \frac{2\pi}{t-b}\{\frac{(D-1)^2}{N-\pi(D-1)^2}\},$$

where t, b and D are wait time, beacon interval, and displacement after a pause time(=ht) respectively.

PROOF: The repair time is constrained by the longest path in the update area (see Figure 4). Let node m be k hops away from the destination S and the node density be 1 for simplicity. Total outgoing data rate at node m can be estimated as the total traffic generated outside the k-hop neighborhood of S, divided by the length of the k-hop boundary. The k hop neighborhood has an area of πk^2 and a circumference of $2\pi k$. Given uniform node density, the rate of traffic crossing the k-hop boundary is $P-\pi k^2 p$. Hence, the outgoing traffic rate at m is $\frac{P-\pi k^2 p}{2\pi k}$. The repairing process is similar to 1D case:

$$b + \frac{2\pi(D-1)}{P-\pi(D-1)^2 p} + \frac{2\pi(D-2)}{P-\pi(D-2)^2 p} + \cdots + \frac{2\pi\cdot 1}{P-\pi p}$$
$$< b + \frac{2\pi(D-1)}{P-\pi(D-1)^2 p}(D-1)$$
$$= b + \frac{2\pi}{p}\{\frac{(D-1)^2}{N-\pi(D-1)^2}\} < t$$
$$\Leftrightarrow p > \frac{2\pi}{t-b}\{\frac{(D-1)^2}{N-\pi(D-1)^2}\}.$$

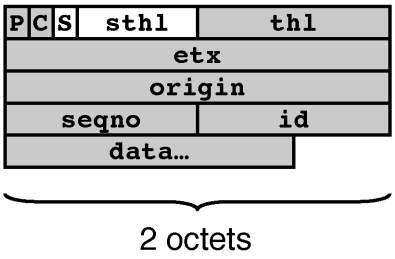


Figure 5: WARP data packet header. Grey fields are standard CTP headers. WARP adds two headers (white) in the reserved portion of a CTP header: the spiral bit S and the spiral time-has-lived STHL, which counts how many hops a spiraling packet has taken.

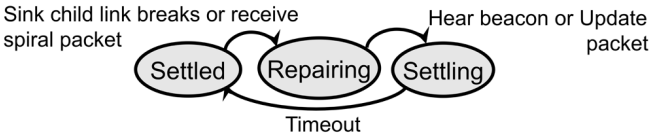


Figure 6: WARP state machine.

4. WARP IMPLEMENTATION

We have implemented WARP as a simple modification to the version of CTP packaged as part of TinyOS 2.1. The CTP data header, shown in Figure 5, has six unused bits in its control field. WARP uses these unused bits for its two additional fields, the "S" bit (spiraling bit) and `STHL` (the spiral hopcount)

For each spiral packet, WARP initializes STHL and increments it on each hop. WARP drops a packet when STHL exceeds a threshold to prevent congestion. To make sure the center of the spiral stays at the last known location of the destination, direct children of the destination initialize STHL to zero, and other nodes to STHL of the most recently forwarded spiral packet. In addition to STHL, WARP uses a single bit, the S (spiral) bit, to distinguish whether a packet is spiraling or not.

4.1 Packet Types

WARP uses STHL and S bit values to distinguish three data packet types: tree, spiral, and update. All are data packets which are sent as unicasts to a next hop. STHL is five bits, so the WARP header takes six bits, which can be placed in the 6 reserved bits of the CTP packet header, shown in Figure 5.

A **tree packet** is the same as a standard collection packet. A node sends to the next hop which provides the minimum cost route. A tree packet has a cleared S bit and STHL = 0. A **spiral packet** is a packet being sent in a spiral. It has the S bit set and a non-zero STHL. Nodes forward spiral packets according to the algorithm in Section 2.3. An **update packet** is sent up a collection tree like a tree packet. However, it also communicates that the sender has stopped spiraling because it has found a new, valid route. An update packet has a cleared S bit and STHL = 1.

4.2 Routing State

WARP has three states: settled (normal operation), repairing (mobility detected), and settling (destination discovered). Figure 6 shows the WARP state machine. Inheriting from CTP, WARP maintains a neighbor table with routing cost information, and chooses the neighbor with minimum cost as its default parent.

Nodes are in the settled state when the current collection tree is stable. Nodes mark incoming data packets as tree packets and

forward them to their parents. While in the settled state, WARP behaves like a standard collection routing protocol.

Nodes enter the repairing state when they detect that the destination has moved enough to break the existing topology. This occurs when a child of the destination detects its link has broken or when a node is asked to forward a spiral packet. Nodes remain in the repairing state until they hear a beacon from a destination or an update packet. In this state, nodes mark data packets as spiral packets and forward them along spiral trajectories described in Section 2.3.

Nodes enter the settling state from the repairing state when they hear a beacon from a destination or when they overhear an update packet. Nodes in the settling state mark data packets as update packets and send them towards the destination. Nodes transition from the settling state to the settled state after a fixed time interval. This time interval needs to be long enough, so that the nodes in the settling state can update their routing tables with valid routes to a destination. In practice, using the same interval as the destination's beacon interval works well.

Our WARP implementation adds an additional 2.8kB of code over CTP's 5.5kB (a total of 8.3kB) and an additional 33 bytes of RAM over CTP's 1kB.

5. EXPERIMENTAL METHODOLOGY

Our evaluation examines three parameters: the network topology, the destination's speed, and the data rates to the destination. Generally speaking, data rates involve continuous traffic well below saturation. Sensor nodes are deployed with a neighbor count between five and eight. Only the destination moves, at a rate representative of a person's walking or jogging speed. All other nodes, while stationary, have real-time link dynamics from the surrounding environment, such as burstiness and external interference.

5.1 Metrics

Sensor networks are often deployed in remote places and have to rely on a self built infrastructure. Scarcity of resources, particularly energy, demands that network protocols are implemented efficiently with a minimum overhead. Consequently, the two main metrics we use to evaluate WARP are routing cost and reliability. We also measure number of hops that data packets take when delivered to the sink, as a rough estimate of the packet delivery latency.

Reliability. One of the main performance indicators of routing protocols is reliability of packet delivery to its destination. Lossy radio links, network congestion, and mobility of nodes all negatively influence reliability. We measure reliability as a fraction of packets successfully delivered to the destination. A trade-off exists between control packet overhead and reliability, that is, better reliability can be achieved if more control packets are utilized.

Cost. Cost is the number of packets the network transmits divided by the number of unique packets a destination receives. Cost is a direct measure of efficiency of routing algorithms and an estimate of their energy requirements.

PL. Path length counts the number of hops a packet has taken to a destination and is a rough measure of packet latency as well as how much work it took to deliver a packet. Analyzing the PL distribution helps verify the correctness of spiral looping because loops, if not controlled, would lead to very high PLs.

5.2 Comparison Protocols

We compare WARP to two sensor network protocols: CTP and Hyper. The Collection Tree Protocol (CTP) [15], the standard collection protocol used in sensor networks today, serves as the primary point of comparison. As CTP is highly optimized for stationary networks, it allows us to quantify the cost of mobility. Ideally, WARP for a mobile destination should approach the reliability and the cost of CTP for a static sink. CTP also illustrates how even very slow sink speed affects performance of the existing data collection protocols. Anecdotally, other protocols, such as MultihopLQI [29], perform worse than CTP.

We selected Hyper [26] as a representative sensor network mobile routing protocol. Hyper supports fast neighborhood assessment and efficient tree convergence to quickly reconfigure networks. Its two essential mechanisms are a fast link convergence algorithm that estimates link qualities of one hop neighbors of a destination and a fast tree building algorithm that builds a minimum cost tree rooted at the destination. Whenever a destination enters a new region, Hyper triggers the rapid building of a collection tree using the newly estimated connectivity parameters (i.e., the ETX metrics) of the destination's neighbors.

We also tested TYMO, the TinyOS implementation of DYMO, a MANET distance vector routing algorithm [11]. We found that its packet delivery ratio drops below 10% under load or even slight mobility. This is due to the fact that MANET protocols route among possibly all mobile nodes and thus are overly pessimistic. This results in significant routing overhead and long delays in delivering packets to a destination under our mobility and data rate parameters. Correspondingly, while we have experimentally compared WARP with DYMO, DYMO's performance is so poor that it does not represent a fair comparison. While we think it is possible to implement versions of DYMO with good performance, that is itself a subject of a separate research effort.

5.3 Topologies

We evaluate WARP on three topologies:

Mirage Testbed: 60 motes at Intel Research Lab, (Figure 7). A transmission power of -10dBm creates a network 5 to 6 hops across. These experiments use 802.15.4 channel 26, CTP Noe [15], and the standard TinyOS CSMA/CA protocol.

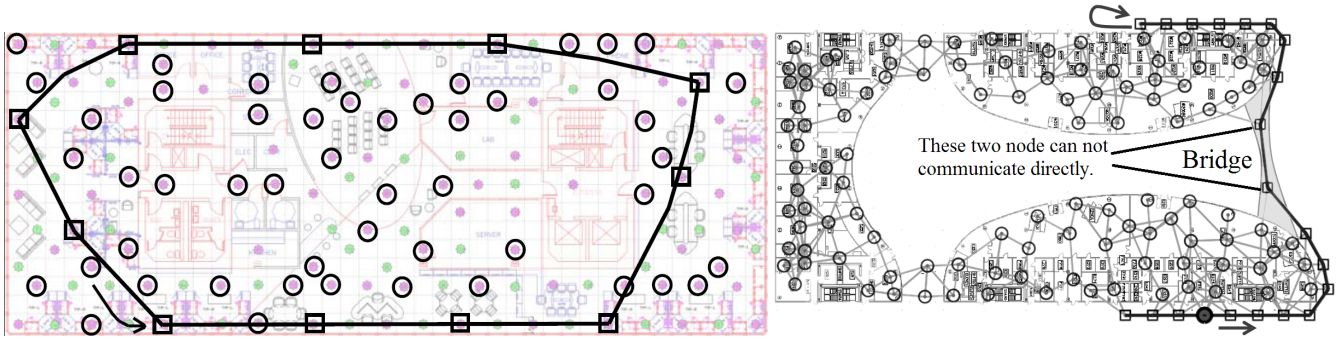
TOSSIM continuous topology: an 8x8 grid topology simulated by TOSSIM, the standard TinyOS simulator [21]. Nodes have a 2-3 grid cell communication range, depending on TOSSIM's node hardware variations.

TOSSIM discontinuous topology: 109 randomly placed nodes in a $9000\ m^2$ region based on the floor plan of a building at Stanford. The building enforces a U-shaped network (Figure 7b). 21 trajectory nodes along a bridge form a synthetic mobility trace (this will be explained in Section 5.4). Nodes at the opposite sides of the bridge cannot communicate directly: crossing the bridge requires all data packets be significantly re-routed (discontinuous topology in this sense).

5.4 Mobility Models

Synthetic mobility. Motion along a predefined trajectory is simulated by having different physical nodes take turns being the destination node. The destination node resides at precisely one trajectory node at any time. After some time, called the *wait time*, the destination discretely jumps to the next trajectory node. The internal state of the destination is also copied to the next trajectory node. This simulated mobility model allows repeatable experiments and does not require physical presence at the testbed.

Real mobility. In this model, a researcher carries a destination node around the Mirage testbed, moving continuously. Unlike the synthetic model, this movement pattern reflects obstacles such as walls, and is independent of the underlying network topology. It



(a) Mirage testbed (b) Discontinuous TOSSIM topology

Figure 7: Data source and trajectory nodes are denoted by circles and rectangles, respectively. A destination node moves along the boundary as depicted by arrows.

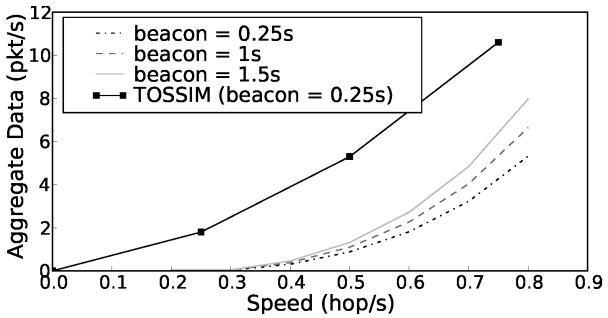


Figure 8: Aggregate data rates P to support mobility h for three different beacon intervals in a grid topology. The top line is for a TOSSIM experiment where data rates giving 80% reliability are plotted.

is a true movement speed, rather than an estimated one from hops per second. Furthermore, the presence of a user introduces a highly dynamic and directional source of signal attenuation.

We test speeds that correspond to slow, regular, and fast walking (0.7 m/s, 1.5 m/s, and 2.9 m/s). In TOSSIM, this translates to 2, 4, 6, and 8 second wait times per hop (0.5, 0.25, 0.17, and 0.125 hop/s). In both simulation and testbed experiments, we let the network initialize for 400 seconds before the destination starts moving. Each node generates 100 data packets at a constant rate. Due to the different number of nodes in simulation and on the testbed, we have slightly different overall data rates: 4.3, 7.1 and 21.3 packets per second (pps) in simulation and 3.3, 5.4 and 16.3 pps in the testbed.

6. EVALUATION

We evaluate WARP in simulation and experimentally. The TOSSIM simulator [21] allows us to validate a simple, controlled network topology as well as a challenging, discontinuous topology a testbed cannot easily create or control. Testbed experiments on the Intel Mirage testbed [12] verify WARP's behavior in a real network and with real mobility tests.

6.1 Theoretical Validation

Figure 8 shows plots for the theoretical bounds described in Section 3, using the grid topology with a 4s wait time and three different beacon intervals. This shows that a higher mobility can be supported by a higher data rate as well as a smaller beacon interval. The TOSSIM simulations verify this trend.

6.2 Synthetic Mobility in Mirage

Figure 9 shows WARP's performance under synthetic mobility in the Mirage testbed. With a stationary destination, WARP defaults to CTP's performance. Across all data rates and even the highest tested speed, WARP's reliability stays above 83%.

Mobility decreases reliability while increasing cost and path length. WARP speculatively routes data packets to a mobile destination, increasing path length. However, WARP adapts rapidly, introducing less than one additional hop on average. Discovering and repairing routes increases WARP's cost under mobility. The cost increase is partially due to longer path lengths and partially due to a lower delivery ratio. Many of the lost data packets are spiral packets which do not find the destination in time. Figure 9(b) is the path length of only delivered packets.

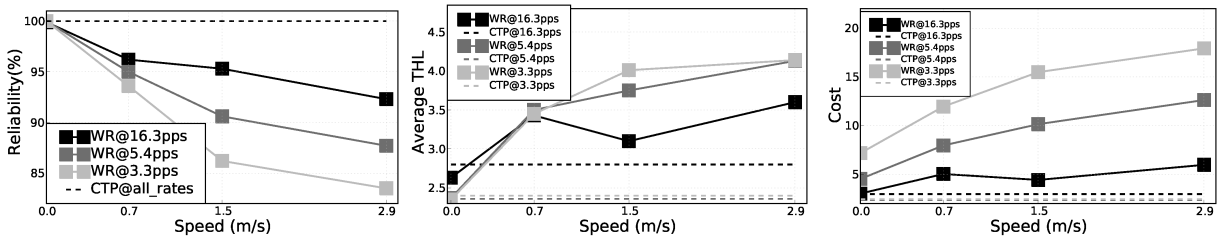
WARP's performance improves with higher data rates. As long as the aggregate data rate prevents the destination node from moving outside the update area without being contacted, WARP can quickly and efficiently deliver packets.

6.3 Simulated Discontinuous Network

We examine how WARP performs when a topology is discontinuous, when successive next hops of the destination node cannot communicate directly. For this experiment, we use the discontinuous topology shown in Figure 7(b). We test three different speeds at an overall data rate of 3 packets/second. WARP performs very well, achieving 93%, 87%, and 82% reliability at speeds of 0.8m/s, 1.6m/s, and 3.3m/s, respectively. The destination node crosses the bridge 15 times for the slowest speed and almost 50 times for the fastest speed.

This experiment is a worst-case situation for WARP: when the destination node moves across the bridge, WARP has to reconfigure the entire network. To measure the effect of this, we approximate throughput by counting how many packets the destination receives in 100ms-time intervals. We low-pass filter this value with a 2.5sec-window for legibility.

Figure 10 plots the throughput when it moves at 0.8 m/s. There are two classes of network reconfiguration events. Long periods of low throughput correspond to bridge crossings that require major topology changes. Short periods of slightly reduced throughput correspond to regular movements of the destination node where local reconfiguration is sufficient. Figure 10 shows the long events as gray areas 1 and 2; the short events are too transient to be visible



(a) Reliability (b) Path length (c) Cost

Figure 9: Mirage experiments. Performance of WARP for a mobile destination is compared to that of CTP with a static sink. For a comparison to CTP with a mobile sink, see Section 6.5.

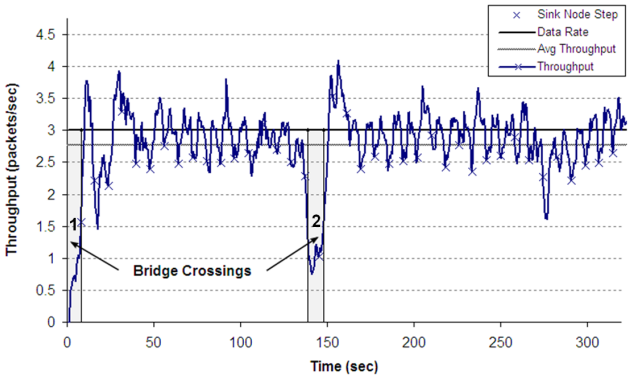


Figure 10: Data throughput at the destination moving along a predefined trajectory, averaged over 15 experiments. Data rate was 3 pps and the average achieved throughput was 2.8 pps (92.6%). Two gray areas in the graph show when the destination node crossed the bridge in two directions.

(they averaged < 0.2sec). Despite large, discontinuous movement of the destination node, WARP achieves good reliability through buffering and retransmission of data packets.

6.3.1 *Real Mobility in Mirage*

Table 1 shows WARP's performance as a researcher carries the destination through the Mirage testbed. Each data point is the result of 3 experiments of 8 minutes. The results in Table 1 show that WARP performs similarly to the synthetic mobility experiments, validating our simulation results. At normal walking speed (1.5 m/s), WARP maintains a reliability of 93% while suffering from only a 10% increase in path length and a 57% increase in cost.

We decompose WARP's cost into control, spiral, and data packets in Table 2. Spiral packets, which actually discover positions of a destination, are a small fraction – between 4% and 7% – of the total load. Using speculative routing, the cost of finding a mobile destination is quite small: WARP spends much more effort optimizing its routing topology than discovering it.

6.4 Individual Mechanisms

Prior sections evaluated WARP's mechanisms together in a complete protocol. This section examines how each of the specific mechanisms contributes to WARP's performance.

6.4.1 *Mobility detection*

We measure how the beacon interval affects WARP's performance. The trade-off is between proactively detecting mobility through beacons or reactively through transmission failures. In the

Speed	Reliability	Path Length	Cost
0m/s	99.2%	3.66	4.59
1m/s	94.4%	3.97	6.19
1.5m/s	93.0%	4.05	7.23

Table 1: Real mobility at 8.1pps in the Mirage testbed. Reliability stays above 90% and cost increases by 55%.

	0m/s	1m/s	1.5m/s
Data	90.8%	83.2%	81.2%
Spiral	1.7%	6.2%	5.8%
Control	7.5%	10.6%	13.0%

Table 2: Cost Breakdown of WARP with real mobility in Mirage experiments: the data rate is 5.4pps, and speeds are in meters/second. Even at high mobility, the control overhead is below 15%.

Interval	Reliability	PL	Cost	Drop	Trigger
125ms	90.6%	3.8	10.1	441	2%
1s	93.6%	4.9	9.2	258	70%
2s	97.6%	5.3	9.1	90	45%

Table 3: Effect of beacon interval under synthetic Mirage mobility (1.5 m/s and 5.4 pps). Drop is spirals lost to TTL, and Trigger is the percentage of beacons that were triggered by a spiral packet.

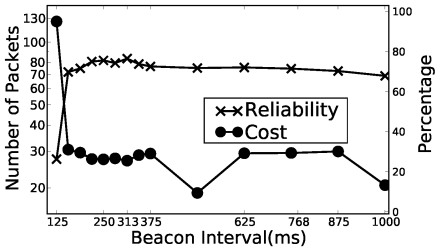


Figure 11: Performance of a simple beaconing scheme with no spiral packets or beacon suppression. The destination moved at 0.5 hops/s and the data rate was 7.1 pps. Reliability is below 70% and the lowest cost is 18.

	Reliability	Queue Drop	Loops/s
No-Spiral	75.6%	34%	23.8
WARP	90.5%	0%	5.1

Table 4: Comparison of WARP with No-Spiral protocol of Fig 11 with 250 ms-beacon interval. Compared to a simple link repair strategy, WARP drops fewer packets and has fewer routing loops.

former case, WARP's routing state is reconfigured rapidly, allowing for lower latency of packet delivery. In the latter, WARP's beacon suppression leads to low routing overhead.

Table 3 shows WARP with three different beacon intervals. A small beacon period reduces path length, but leads to more packet drops. Collisions by frequent beacons cause more false-positive mobility events generating unnecessary spiral packets. As more spiral packets route speculatively, they have lower reliability.

6.4.2 Speculative Routing and Local Repair

We quantify the benefit of speculative routing and local repair by disabling spiraling and beacon suppression. We call this simple link-repair protocol *No-Spiral*.

We first find the best beacon interval for a given topology and workload. Figure 11 shows that, for 7.1 pps and 0.5 hops/s, a 250 ms interval is a good trade-off point between cost and reliability. We compare *No-Spiral* at this beacon rate to WARP in Table 4. *No-Spiral* has lower reliability with many of the losses being due to queue overflows: the protocol is slow in adapting to topology changes and so queues overflow as it retries failed links. WARP's ability to proactively find a destination leads to much better reliability than repairing links through simple beaconing.

6.4.3 Control Traffic Suppression

To evaluate the effect of control traffic suppression on cost, we decompose WARP's cost in the Mirage testbed. As shown in Table 2, the packets can be divided into data, spiral, and control packets. Control packets can be further divided into WARP beacon and CTP beacon packets (WARP does not modify control mechanisms of the underlying protocol). We compare the expected number of beacons, based on the beacon timer, to the number of beacons that the mobile destination actually transmits. WARP suppresses 78-82% of beacons from the destination node, and beacons from the destination node correspond to less than 10% of the total control cost. WARP's control suppression enables it to introduce only a small overhead on top of the control traffic of its underlying routing protocol.

	Reliability	PL	Cost	Ingress
4.3pps	84.2%	12.0	43.7	83
7.1pps	88.7%	13.1	72.1	824
21.3pps	48.8%	10.3	96.0	8146

Table 5: TOSSIM grid with high mobility (2 hops/s). ***Ingress*** **denotes packets dropped due to ingress drops.**

6.5 CTP and Hyper

Figure 12(a) shows reliability of WARP, Hyper, and CTP in the TOSSIM grid topology. We use this topology because its regularity and density is the simplest case for a mobile routing protocol. For example, it does not penalize Hyper for Mirage's quirks.

When the destination is stationary, all three protocols have a reliability above 90%: WARP simply defaults to CTP and Hyper performs very well. Reliability decreases as the destination moves faster. WARP maintains above 89% reliability even at the highest speed. Hyper's reliability drops to below 60% under high mobility.

Hyper's poor performance at high mobility is because it was designed for discrete, rather than continuous mobility. Hyper expects users to rarely switch transmission domains and does not maintain connectivity during motion. In our evaluation, we have a stream of data coming from sensors at all times. This causes Hyper to backlog a significant amount of data for later transmission, congesting the network after the new routing tree becomes available. In contrast, WARP speculatively routes packets to the destination while updating its routing state and does not accumulate many data packets for later delivery. Consequently, Hyper's reliability decreases with higher speeds while WARP maintains high reliability.

The difference in the cost metric shown in Figure 12(c) is even bigger (note that the Y-axis is logarithmic). As the destination changes the transmission domain every few seconds in our case, Hyper incurs an order of magnitude larger routing cost than WARP. Finally, Figure 12(b) shows the average path length for packets delivered to the destination. As these results are only for successfully delivered packets and there are few delivered packets for Hyper at high mobility, the PL of Hyper decreases with higher mobility.

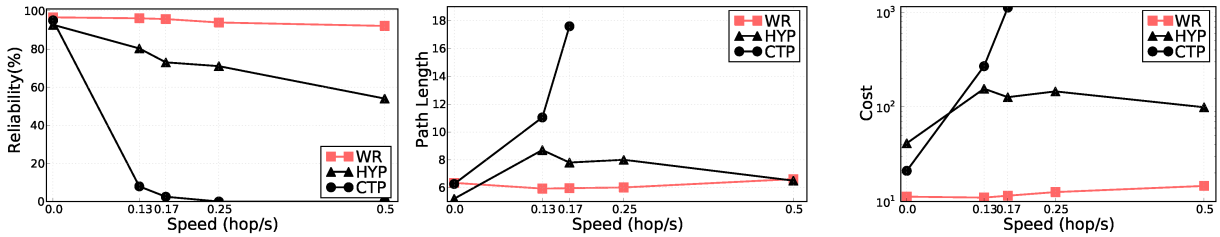
6.6 WARP limitations

Finally, to understand WARP's limitations, we evaluate its performance at very high speeds and data rates.

The results shown in Table 5 illustrate that WARP techniques become inefficient when the destination moves very fast relative to the underlying topology. Both the path length and routing overhead increase dramatically: spiral whirlpool packets fail to find the new location. Consequently, the network topology does not reconfigure and WARP drops data packets due to overflowing queues. If the destination moves so quickly that it exits the update area before the topology can repair, then WARP is unable to operate well.

Previous results showed that WARP's performance improves at higher data rates. However, this is only true as long as the data rate does not saturate the network. If the destination is moving very fast, the high degree of speculative routing increases cost and path length. If the data rate is high, this higher number of transmissions causes nodes to drop data packets before they can distribute signaling information. Higher speeds lower the data rate which WARP can support.

Table 6 examines how mobility rates affect performance at a very high data rate of 64 pps. In this case, path length, cost and reliability remain stable. The high data rate allows WARP to quickly find and repair the topology.



(a) Reliability (b) Path length (c) Cost

Figure 12: TOSSIM grid experiments over varying speeds and at 21.3pps data rate. CTP, Hyper (HYP), and WARP (WR) results are shown.

Speed	Reliability	PL	Cost
0hop/s(CTP)	36.1%	6.36	29.7
0.13hop/s	44.8%	6.56	27.6
0.17hop/s	46.4%	6.84	26.2
0.25hop/s	45.9%	6.85	27.3
0.5hop/s	42.1%	6.85	30.0

Table 6: TOSSIM grid: evaluation of WARP for high data rate (64 pps).

6.7 Summary

WARP is able to route packets to a highly mobile destination, even when its motion is not continuous with the underlying connectivity. In continuous scenarios, simulation and testbed results show WARP's delivery ratios as high as 99.7%, and do not drop below 80%. WARP is able to quickly detect mobility through a combination of proactive (beacons) and reactive (data) mechanisms, but suppresses unnecessary beacons (roughly 80%) to reduce its overhead. WARP's in-band signaling allows data packets to pass the few bits of needed control information, reducing control traffic needs further: WARP's beacons constitute less than 10% of the control cost. WARP's speculative routing quickly finds new routes, while spiral packets constitute only a small portion of the overall data traffic. Finally, collaborative and localized reconfiguration means that once new routes are found, spirals stop quickly.

7. RELATED WORK

Routing in mobile ad-hoc networks (MANETs) has been an active area of research for the past 15 years [25, 24, 13, 11]. These algorithms, however, failed to achieve widespread adoption in real world deployments. The complexity of the MANET problem – every node is mobile – is one reason for this disconnect: instead, MANET protocols are used in static meshes [3, 1, 2]. Perhaps a more significant factor in achieving high performance outside the simpler world of simulation is the basic challenges of wireless by itself [6]. Even though mechanisms that improve reliability of wireless communications are relatively well known, few protocols implement them in practice. Routing in a static mesh is a challenging enough problem by itself which has only begun to be solved effectively in the past few years [8, 15]: fully mobile ad-hoc networks remain an open and difficult problem.

WARP carefully monitors wireless links over time and rapidly adapts to changes in the underlying network topology. It improves reliability of the links by using unicast packet transmissions and link layer acknowledgments and reliability of multi-hop routes by using ETX as opposed to the hopcount metric. WARP limits network congestion by meticulously keeping packet transmission overheads down. It uses data traffic statistics to estimate link qualities, suppresses majority of control beacons, and repairs the routing state locally rather than by an uncontrolled network-wide flood.

Clearly, WARP benefits by considering the narrower case of a mobile node moving through a static mesh. However, in our experience, existing protocols [11] fail entirely in these static mesh experiments with even a limited mobility of the sink. WARP is able to use very simple mechanisms to achieve high packet delivery and low cost in real as well as controlled mobility experiments.

One notable exception of wireless routing protocols used in practice is OLSR [13]. It is used in community wireless mesh networks in Athens [1], Berlin [2], and Leipzig [3]. These are, however, not mobile ad-hoc networks as they consist of stationary nodes [22]. Furthermore, these successful deployments change OLSR to use ETX as the routing metric, rather than hopcount, an extension that is part of the in-progress OLSRv2 specification [5].

WARP's approach of using the datapath to maintain a routing topology is similar to CTP's datapath validation [15]. WARP takes CTP's approach one step further. Where CTP uses data packets to detect when control plane updates are needed, WARP uses them to replace control messages whenever possible through suppression.

WARP combines the ideas from AODV-LL [9] and MicroRouting [14] as a way to achieve both low cost and fast mobility detection. It borrows the idea of link-layer feedback to detect destination mobility from AODV-LL and proactive mobility discovery through beacons from MicroRouting.

WARP builds on the long history of local recovery research [14, 20, 27, 10]. Like many of these local recovery schemes, WARP detects broken links, retrieves previously cached alternate routes, and invalidates stale routes. Similar to AODV-BR [20], SLR [27], and query localization [10], WARP limits the range over which nodes search for a route. However, WARP differs from these approaches in that it actively probes and incrementally builds new routes without relying on a source or a destination to initiate route recovery.

WARP has intellectual similarities to consensus routing [16], which uses the notion of network-wide consensus to detect inconsistencies and resort to backup routing schemes when needed. But unlike consensus routing, which uses a controlled or periodic consensus mechanism, WARP is entirely decentralized and distributed, as its repairs are local.

In a similar vein, failure carrying packets [19] propose an extension to link-state routing where packets contain routing failure information. Gateways route around failures through a periodic update of consensus on network state.

Finally, Hyper [26] is perhaps the most similar protocol to WARP, as it routes data from a static network to a mobile node. But unlike WARP, which is designed for continuous mobility, HYPER is in-

tended for discrete mobility events. As our evaluation in Section 6 shows, this causes HYPER to have highly degraded performance when a destination's movement is continuous.

8. CONCLUSION AND FUTURE WORK

In this paper, we presented the Whirlpool Ad-hoc Routing Protocol (WARP), a data collection protocol for a mobile destination. WARP maintains high reliability at varying data rates and mobility speeds. WARP achieves these results by rapidly detecting mobility and locally repairing topology using the existing tree topology. The key insight is speculatively sending data packets around the last known location of a destination. WARP carefully monitors wireless links over time and rapidly adapts to changes in the underlying network topology. It improves reliability of the links by using unicast packet transmissions and link layer acknowledgments and reliability of multi-hop routes by using ETX as opposed to hopcount metric. WARP limits network congestion by meticulously keeping packet transmission overheads down. It uses data traffic statistics to estimate link qualities, suppresses majority of control beacons, and repairs the routing state locally rather than by an uncontrolled network-wide flood.

WARP weakens the separation between the control and data planes, using data packets to probe and repair the topology. Consequently, WARP can deliver data and maintain routing state simultaneously, allowing it to aggressively suppress control packets. These techniques cause WARP to remain efficient while improving performance as the data rate increases.

In our future work, we would like to investigate three areas of study. The first is the importance of the beacon period in WARP. Our results indicate that lower beacon periods improve WARP's performance, as fast beacons can reduce the efficacy of whirlpooling. However, this rate depends on the data rate; if there are few packets to whirlpool, then WARP might need more control packets. There is a tension between local topology repair and speculative routing; having WARP automatically adapt its rates accordingly seems beneficial.

The second is the trade-off between data and control traffic in topology maintenance. Data traffic can quickly find new routes and detect broken routes, but can only use entries in a node's neighbor table; WARP might benefit greatly from opportunistic reception, more so than its current spiral packet snooping affords.

Finally, we would like to study how to dynamically adjust the range of local repair. WARP currently uses a fixed range value, even though the optimal value depends on many dynamic factors, such as mobility, the beacon rate and the data rate. We believe that dynamic adjustment and tuning could decrease routing maintenance cost while increasing mobility responsiveness.

9. REFERENCES

[1] Athens Wireless Metropolitan Network. `http://www.awmn.net`.

[2] Berlin Freifunk Network. `http://berlin.freifunk.net`.

[3] Leipzig Freifunk Network. `http://leipzig.freifunk.net`.

[4] Meraki Network. `http://meraki.com`.

[5] OLSRv2 Specification. `http://ietfreport.isoc.org/idref/draft-ietf-manet-olsrv2`.

[6] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *ACM SIGCOMM 2004*, 2004.

[7] R. Beckwith, D. Teibel, and P. Bowen. Unwired wine: sensor networks in vineyards. *Sensors, 2004. Proceedings of IEEE*, pages 561–564 vol.2, Oct. 2004.

[8] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computin g and networking*, 2005.

[9] 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 *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97, New York, NY, USA, 1998. ACM.

[10] R. Castañeda, S. R. Das, and M. K. Marina. Query localization techniques for on-demand routing protocols in ad hoc networks. *Wirel. Netw.*, 8(2/3):137–151, 2002.

[11] I. Chakeres and C. Perkins. Dynamic MANET On-demand (DYMO) Routing. Internet-Draft, draft-ietf-manet-dymo-12.txt, 2008.

[12] B. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. Parkes, J. Shneidman, A. Snoeren, and A. Vahdat. Mirage: A microeconomic resource allocation system for sensornet testbeds. In *Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors (EmNets)*, 2005.

[13] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR). *RFC Editor*, 2003.

[14] S. M. Das, H. Pucha, and Y. C. Hu. Microrouting: A scalable and robust communication paradigm for sparse ad hoc networks. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 12*, page 245.2, Washington, DC, USA, 2005. IEEE Computer Society.

[15] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the Seventh ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2009.

[16] J. P. John, E. Katz-Bassett, A. Krishnamurthy, and T. Anderson. Consensus routing: The internet as a distributed system. In *Fifth USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI)*, 2008.

[17] D. B. Johnson, D. A. Maltz, and J. Broch. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In *In Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5*, pages 139–172. Addison-Wesley, 2001.

[18] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, 2007.

[19] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica. Achieving convergence-free routing using failure-carrying packets. In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, New York, NY, USA, 2007. ACM Press.

[20] S. Lee and M. Gerla. Bakcup routing in ad-hoc networks. In *IEEE Wireless Communications of Networking Conference (WCNC)*, 2000.

[21] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Simulating large wireless sensor networks of tinyos motes. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[22] F. Y. Li, L. Vandoni, G. Zicca, and S. Zanoli. OLSR Mesh Networks for Broadband Access: Enhancements, Implementation and Deployment. In *4th IEEE International Conference on In Circuits and Systems for Communications*, 2008.

[23] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications*, Sept. 2002.

[24] S. Murthy and J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Mobile Networks and Applications*, 1(2):183–197, 1996.

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

[26] T. Schoellhammer, B. Greenstein, and D. Estrin. Hyper: A routing protocol to support mobile users of sensor networks. In *CENS Technical Report*, 2006.

[27] C. Sengul and R. Kravets. Bypass routing: An on-demand local recovery protocol for ad hoc networks. *Ad Hoc Networks*, 4(3):380 – 397, 2006.

[28] C. Sharp, S. Schaffert, A. Woo, N. Sastry, C. Karlof, S. Sastry, and D. Culler. Design and implementation of a sensor network system for vehicle tracking and autonomous interception. In *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN)*, 2005.

[29] TinyOS. MultiHopLQI. `http://www.tinyos.net/tinyos-1.x/tos/lib/MultiHopLQI`, 2004.

[30] G. Werner-Allen, K. Lorincz, M. Welshand, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, 2006.