A Cross-Environment Study of Routing Protocols for Wireless Multi-hop Networks

Erik Nordström, Per Gunningberg, Christian Rohner and Oskar Wibling

Department of Information Technology, Uppsala University

Abstract—Real world evaluation of network protocols is often a tedious task due to factors such as test execution time, repeatability, and tracking log files for the cause of performance problems. We facilitate the evaluation procedure by using a structured and tool-supported approach, combining real world experiments with simulation and emulation. This method has helped us in analyzing the three ad hoc routing protocols AODV, DSR, and OLSR. In addition to just concluding that the protocols yield a lower total throughput in the real world compared to, e.g., simulation, we isolate problematic time regions and manually inspect the logs at relevant places. We can thus quickly find the origins of problems; one example being the identification of an important cause for TCP's bad performance in ad hoc networks. The neighbor sensing mechanism, commonly implemented using discovery messages, is flawed in a real world situation with heavy traffic load. A second finding is the large real world latencies for DSR in scenarios with intermittent connectivity. The cause of the problem was traced to the aggressive buffering which is done in DSR.

I. INTRODUCTION

In this paper we study the sensitivity of different routing mechanisms when exposed to a real world wireless radio environment. Previous research indicates that ad hoc routing protocols perform well in simulation but break down in different and unpredictable ways in the real world. What design choices of each protocol are responsible for this is not yet well understood. It is clear that sensing is an important factor and often simplistic simulation models are blamed for the discrepancies between simulation and the real world. However, instead of focusing on improving simulations by pursuing better models, we use a novel methodology where we combine simulation, emulation and real world experimentation. This allows us to study each protocols’ sensitivity to different factors such as systems deployment (e.g., integration with hardware and software stacks), radio environment and real world mobility, because each of simulation, emulation and real world offer a different level of abstraction to these factors. To our knowledge, this is the first such large cross-environment study.

Our most important finding is the far-reaching consequences of sensing, in particular neighbor monitoring. Other important, and often inter-dependent factors, are buffering and timing issues. For example, previously unreported is the consequence of aggressive buffering which, in combination with route-flapping, seriously degrades traffic throughput.

We have chosen to study the well known wireless multi-hop routing protocols, AODV [16], DSR [10] and OLSR [5]. They have all been part of the IETF standards track and represent very different design choices and are hence suitable for our purpose. Our work is motivated by the fact that these routing protocols, despite their maturity, are only recently being used in the real world, for example in the increasingly popular area of mesh networks. Although these protocols have been experimented with in such scenarios, there are no side-by-side comparisons and few studies include mobility. Real world side-by-side comparisons are particularly demanding, because they require that experiments are repeatable. Repeatable experiments ensure that inherent differences in the protocols are exposed instead of effects from random factors and that the results are conclusive. Furthermore, current studies [1], [6] do not expose the dynamic properties of the protocols, either because the scenarios are static or the protocols only implement a limited set of features that limit their efficiency in a mobile environment.

We use a novel scenario based approach for our cross-environment study. A set of carefully designed and controlled scenarios have been reconstructed in each environment. Previous work that couples real world and simulation [9], [11] is trace based. Routing protocols are compared in simulation and real world by feeding (e.g., GPS) traces into a simulator.

Also distinguishing from other work is the extent of our comparison. There are 27 combinations of three scenarios, using UDP, Ping and TCP traffic and AODV, DSR and OLSR routing. Each experiment combination...
was repeated 10 times, resulting in 270 independent experiments. The complete traces comprise around 2 Gbytes of data and will be made available for other researchers along with all software.

The paper is outlined as follows. The next section describes our experimental setup and methodology in detail. Section III presents the main results from our comparison through an extensive analysis. Section IV presents related work and Section V concludes the paper with a discussion and future work.

II. EXPERIMENTAL SETUP AND METHODOLOGY

The real world experiments feature people that carry laptops and move according to a scenario choreography that is displayed on the screens. All experiments take place in our building, see Figure 1.

The emulations are run on the same platform as the real world experiments but the nodes are stationary in a room and use MAC filters to emulate the mobility and connectivity changes. In our ns-2 simulations we use the same scenarios translated into a schedule and a commonly used radio propagation model.

We use the same protocol implementations that run natively in real world, for all three environments. This contrasts to previous studies which have relied on implementations that use emulation or translation layers to be able to run simulator code in the real world [18], [14], [7]. These approaches suffer from considerable overhead and sometimes require specific scheduling between real time and simulator time which increases the uncertainty in the results and the conclusions.

The mobility scenarios comprise four nodes and up to three hops. This choice of scale is a conscious decision as it allows multi-hop topologies, while making repeatable mobility patterns achievable. We defer increasing the scale of the scenarios until protocol problems in the smaller scenarios are solved and the protocols achieve reasonable performance there. Working with larger scenarios is considerably more challenging for in depth analysis and repeatability, and would be extremely time consuming.

A. Scenario Descriptions

Our comparisons comprise three mobility scenarios: End node swap, Relay node swap and Roaming node. They are choreographed to test different aspects of the routing protocols. Their simplicity makes it feasible to recreate them in the simulation and emulation environments. Figure 2 depicts logical overviews of the scenarios. Positions A, B, C and D correspond to the physical locations in Figure 1. At these positions, nodes only have connectivity to their adjacent neighbors. In all experiments, there is only one traffic stream between nodes 3 and 0, consisting of either UDP packets, Ping messages or TCP segments. The scenarios are constructed so that there is always connectivity between the source node (3) and the destination node (0) over one or more hops. Nodes move at normal walking speed. We measured it to about 1.3 m/s. Each scenario has a warm-up phase and a cool-down phase of at least 10 seconds, during which the routing protocols have time to converge (in the case of OLSR) or, in the case of cool-down, to deliver delayed data packets before the experiment ends. The traffic streams start some time after the warm-up phase, depending on scenario.

These scenarios are selected since they stress the ability of the routing protocols to adapt to different situations as discussed below.

![Fig. 2. Logical overview of the three scenarios: (a) End node swap, (b) Relay node swap, and (c) Roaming node. The dotted lines indicate movement and the times indicate when nodes start to move and when they pass the waypoints: A, B, C, and D.](image-url)
The **End node swap** scenario (Figure 2 a) aims to test a routing protocol’s ability to adapt when both source and destination move and the shortest path changes from three hops, through two hops, to one hop and back. At time 31s, data transmission from node 3 to 0 starts. At time 51s, end nodes 0 and 3 start to move toward the other end node’s position (A and D) where they arrive at time 113s. Nodes 1 and 2 are stationary during the course of the scenario. In this scenario we expect to see how quickly a protocol switches to a new and shorter path. Some protocols hang on to an old path until it breaks while other protocols continuously look for alternative and shorter paths.

The **Relay node swap** scenario (Figure 2 b) instead tests how a routing protocol handles mobility among intermediate nodes while the end nodes are stationary. The traffic is initiated between node 3 to node 0 at time 61s, the relay nodes 1 and 2 start to change positions at time 81s. When they meet in the middle, our node placement allows, depending on the current connectivity, a two hop route between end nodes 0 and 3 using either one of the relay nodes as an intermediary. The relay nodes reach their destinations at time 101s.

The **Roaming node** scenario (Figure 2 c) starts with one hop between nodes 0 and 3, in contrast to the other scenarios. Also, there is no movement among potential relay nodes. Instead, node 3 roams the network, moving from position A to position D and back during the course of the scenario. All other nodes are stationary and only forward traffic. When initiating the traffic at time 26s, node 3 starts its movement from position A toward position D. At time 88s, node 3 has reached position D and heads back toward position A, which it reaches at time 150s. The Roaming node scenario aims to mimic, for example, a mesh network where a user (node 3) is mobile and communicates with a gateway (node 0). Here we study the effect of increasing path length and the route optimization behavior when node 3 moves back.

### B. Coupling the Real World, Emulation and Simulation

Table I lists a number of factors in which the real world, emulation and simulation differ for our experiments. We know from previous work that the radio is an important factor for explaining performance discrepancies between the platforms [11]. In order to study its impact we try to control the other factors. Mobility is handled with choreography and scenarios. Hardware and software are identical while protocol logic is varied with the routing protocols, but otherwise the same between the platforms. Through this harmonization we can use simulation and emulation as a baseline for the protocols and gradually expose the radio factors of the real world.

<table>
<thead>
<tr>
<th>Environment</th>
<th>Routing Logic</th>
<th>HW Stack</th>
<th>Mobility</th>
<th>Radio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real World</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Emulation</td>
<td>✓</td>
<td>✓</td>
<td>Partial</td>
<td>Partial</td>
</tr>
<tr>
<td>Simulation</td>
<td>✓</td>
<td>✗</td>
<td>Model</td>
<td>Model</td>
</tr>
</tbody>
</table>

**TABLE I**

**RELATIONSHIP BETWEEN THE REAL WORLD, EMULATION AND SIMULATION IN TERMS OF Routing logic, Hardware, Networking stack, Mobility AND Radio environment. A ✓ MEANS THAT THE ELEMENT IS GENUINE AND NOT A MODEL OR, BY RESTRICTIONS, PARTIAL. A ✗ INDICATES ABSENCE OF AN ELEMENT.**

The real world and the emulated experiments use the Ad hoc Protocol Evaluation (APE) testbed [12], which is publicly available. APE is driven by scenario descriptions to ensure that the mobility pattern is repeatable from one experiment to another, modulo the natural variance caused by people moving around with laptops. We have run complementary experiments to quantify this variance, by varying the mobility artificially and by having people interfere. Further, we have compared SNR graphs generated from all test runs performed. The SNR graphs from each scenario have a similar shape which shows that the natural variance is not significant for the experiments in this paper.

When an experiment is conducted, APE reads commands from a scenario file and executes them at the specified time. The scenarios schedule the traffic load and contain instructions to the persons carrying the laptops. As traffic we use either synchronous UDP or Ping traffic (CBR), or a TCP file transfer. APE logs all traffic seen by all nodes during the experiments. By matching the time stamps of all logged packets from all nodes after the experiment, we get a complete global view of the whole experiment. The bulk of our post experiment analysis is based on this information. The logging adds overhead which has some effect on TCP throughput. Our UDP and Ping measurements are not affected by this logging.

All APE computers are identically configured IBM Thinkpad X31 laptops. APE version 0.5, built with Linux kernel 2.6.9 is used for all experiments. The WiFi interfaces are PC-card based Lucent (Orinoco) silver cards supporting the IEEE 802.11b standard. The cards use the Agere Systems Linux driver version 7.18 (March 2005).
which we have updated to support Linux kernel 2.6 and an updated wireless extension API. This driver comes with its own firmware that is dynamically loaded onto the card at initialization. All our experiments are run with the driver set to 11Mbps fixed rate with RTS/CTS turned off. See for example [22] for a motivation to this choice of RTS/CTS setting.

1) Emulation: The emulations use the same HW/SW platform as the real world experiments including the wireless cards. Nodes are stationary and in close proximity, e.g., in the same room and their radios will intentionally interfere with each other. This type of emulation is relative simple, but also quite common and allows comparisons to previous work [7]. The connectivity changes, due to mobility, are emulated using MAC filters by selectively filtering traffic between nodes. The times to enable and disable the filters are extracted from traces generated in the real world experiments. A connectivity change matches the time when the real world signal strengths causes a change of connectivity. We do not introduce any variance in this connectivity time.

This filtering schedule is added to the APE scenario schedule making the connectivity changes completely predictable. The channel quality is high and stable until a change. Besides the predictability the approach eliminates the impact of, for example, gray zones [13] when the signal strength is so weak that the connectivity fluctuates and there are other radio propagation phenomena that degrade the radio channel. The emulation results can therefore, when compared to the real world results, give an indication of the impact of these phenomena on the different routing protocols. Although nodes are stationary in the emulation, it is important to observe that there are still some internal and external radio interferences that impact the experiments. However, our measurements show that this variance is negligible in our context.

2) Simulation: For the simulation we use ns-2 version 2.29. We recreate mobility in ns-2 to match the scenarios from APE. Nodes are configured in a chain topology with logical placement and distances matching those of the real world measurements. The node movement speed is programmed to be between 1.33 ± 0.0125 m/s. This gives a variance in the times when each waypoint is passed of up to two seconds. In the real world experiments, each waypoint is reached within 1-2 seconds of the scripted time.

The choice of a radio model to match the actual environment is delicate. We settled on using the standard ns-2 TwoRayGround model to be comparable to other simulation studies and to determine whether this commonly used model can be used to predict the real world performance of our routing protocols. However, to make this simple model match our experimental indoor set-up better, we tuned the WiFi transmission range to 45m. It is slightly longer than the measured average value. The real values vary, of course, much more unpredictably with the actual building layout. We believe that the simulations still provide a convincing reference to the emulation and real world experiments. The chosen parameters for the radio model are listed in Table II.

C. Routing Protocols

The MANET working group [19] intends to standardize one reactive and one proactive protocol based on AODV, DSR, and OLSR. Current candidates are DYMO [3] and OLSRv2 [20]. There are two main reasons why these two protocols are not in our comparison. First, they are not yet as mature, e.g., in terms of implementations. Second, DYMO and OLSRv2 are evolutionary steps from AODV and OLSR, mainly differing in packet header format. Therefore, we anticipate that by comparing AODV, DSR, and OLSR1, there could be valuable input for the design choices of both DYMO and OLSRv2. Furthermore, AODV, OLSR, and DSR represent very different design choices and vary in how dynamic they are. In the following sections we give a brief overview of AODV, DSR, and OLSR focusing on the differentiating aspects and implementation specific details. For more complete descriptions we refer to the literature or respective RFCs. Note that, for brevity, we

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt_</td>
<td>Transmitted signal power (W)</td>
<td>0.031622777</td>
</tr>
<tr>
<td>freq_</td>
<td>Frequency (Hz)</td>
<td>2.472e9</td>
</tr>
<tr>
<td>CSThresh_</td>
<td>Carrier sense threshold (W)</td>
<td>5.011872e-12</td>
</tr>
<tr>
<td>RXThresh_</td>
<td>Receive power threshold (W)</td>
<td>1.45647e-09</td>
</tr>
<tr>
<td>RTSThreshold_</td>
<td>RTS/CTS exchange threshold (bytes)</td>
<td>2000</td>
</tr>
</tbody>
</table>

**TABLE II**

Simulation parameters to mimic an 802.11b WiFi card with a transmit radius of 45 meters (indoors) and RTS/CTS turned off.

1TBRPF [15] is also a MANET protocol, but it is excluded from our comparison because there are no implementations due to intellectual property right (IPR) issues [2].
refer to the protocol names and not the implementations throughout the paper.

1) Ad hoc On-demand Distance Vector Routing: AODV only disseminates routing updates on-demand, when a route to a new destination is needed. The source node floods the network with a broadcast route request (RREQ). Upon reception of this RREQ, the destination or an intermediate node with a route to the destination replies with a unicast route reply (RREP). Forwarding state is configured on intermediate nodes as these request-reply messages traverse the network. The routing tables are soft state and entries time out when packets are no longer forwarded on a route. Because routing updates are not periodic, AODV must monitor links between neighbors to detect link failures, either with periodic HELLO beacons or using link layer feedback. Link layer feedback is often the more efficient, but is only available in the ns-2 simulation. HELLO messages are sent using broadcast and do not guarantee symmetric link connectivity. The AODV implementation used in this evaluation is AODV-UU v0.9.1 [21].

2) Dynamic Source Routing: DSR is also an on-demand protocol and features similar route discovery as AODV. However, forwarding state is not configured on intermediate nodes in the request-reply phase. Instead, routing information is accumulated in control messages as they traverse the network. Each node caches this information and builds its own local view of the network connectivity. End nodes use the information to build complete source routes, listing all nodes from source to destination. A source route is appended to all packets and intermediate nodes only use this source route to forward data. DSR's link monitoring is based on network layer acknowledgements (nACKs), which is a mechanism that periodically exchanges ACK-request – ACK messages over active links to monitor connectivity. The timeout value for a link is dynamically determined from link RTT measurements, similarly to TCP timeouts. In contrast to AODV, DSR can support automatic route shortening. Using promiscuous mode a node can overhear packets and, by inspecting the source route, discern whether an optimization can be performed and then notify the sender. We use the DSR-UU implementation v0.2 [21]. DSR-UU implements link monitoring using nACKs and alternatively link layer feedback in ns-2. DSR ACK requests are piggybacked on data if possible, but ACKs require an extra transmission. DSR-UU reserves 50 bytes of space for the variable length DSR header in each data packet, effectively reducing the optimal amount of data in each packet. This fixed size reduces implementation complexity, but also reduces the capacity.

3) Optimized Link State Routing: Unlike AODV and DSR, OLSR is a proactive link state protocol similar to OSPF, but with optimizations for ad hoc networks that reduce control traffic overhead and increase reactivity to topological changes. OLSR minimizes control traffic overhead in two ways. First, by using multi-point relays (MPRs) to transmit control messages through the network. Second, by only requiring partial link state information to be flooded.

OLSR relies on HELLO messages to maintain a neighbor set. In a HELLO message, a node announces its link set, neighbor set and MPR set. These messages only reach direct neighbors. In contrast to AODV, OLSR requires a symmetric link to establish connectivity with a neighbor. Actual link states are only propagated throughout the network by MPRs in Topology Control (TC) messages. TC messages contain sufficient link state to build the topology information base and to perform route calculation. Because of the proactive nature of OLSR, the protocol needs time to converge and reacts more slowly to topological changes. We use the OOLLsr implementation v0.99.15 in our experiments [17].

D. Traffic Configuration

All experiments have one data flow between a source node to a sink node consisting of either synchronous UDP packets, Ping or a TCP file transfer session. The CBR rate for UDP and Ping is 20 packets per second while TCP transmits with the highest achievable rate. UDP and Ping have no adaptive mechanisms such as congestion control. Therefore, UDP is used to sample the network connectivity and to measure the route latency. Ping requests are sent to the sink node, which then generates a reply packet for each received request. The request-reply mechanism is used to examine bi-directional connectivity and to measure the round trip times (RTT). TCP is used to study the effect of congestion control and reliable delivery.

We wanted to use the same packet size for all three traffic cases to minimize differences in size induced loss. We settled on 1378 bytes to allow for the DSR header in UDP and Ping. Still, DSR pays a performance penalty for TCP due to this header overhead.

III. EVALUATION

The performance measurements from our cross-environment routing protocol comparison are compiled and presented in Table III. This compilation helps us
understand the performance of the studied routing protocols when moving between evaluation environment on one hand, and traffic types and scenarios on the other. We can study the transient performance and isolate protocol specific causes of performance change from those of the evaluation environment.

A. Measurements and Metrics

The performance measurement results for each protocol, scenario and traffic type in Table III, are averaged over ten measurement series, except for the emulation results. The variance in the emulated UDP and Ping experimentes were negligible under the deterministic connectivity, and are therefore only from one measurement series. The standard deviation and min/max values are also given.

We use the following metrics in our analysis. Delivery Ratio is the fraction of packets delivered out of those generated. Throughput is the number of bytes useful data delivered divided by the time over which data is sent. This is also referred to as Goodput. Latency standard deviation (σ) is the variation in time for a packet traveling from the sender to the receiver. For Ping and TCP we calculate the round trip latency standard deviation. We use the standard deviation instead of the mean for two reasons: First, the standard deviation reflects the stability of routes. Second, in the case of UDP, accurate calculation of the mean is not possible due to the lack of exact time synchronization between the nodes. The Average Hop Count is calculated from the source node to the destination node for UDP traffic. For Ping, the hop counts for the Ping request and the Ping reply are added. Similarly, for TCP the sum of the hop counts for data packet and ACK is used.

There are a number of limitations in our measurements that should be accounted for in the analysis. The emulated TCP transfers are performance limited by node interference. The results are therefore not comparable to those from simulation and real world, but the data is provided for completeness. The real world experiments suffer from the overhead of logging. Therefore, those TCP results should only be compared between the routing protocols and not the test environments. For AODV and DSR we include link layer feedback in some of our simulation results. The majority of simulations in related work use link layer feedback.

B. Performance when Varying the Evaluation Environment

We first establish the impact of the evaluation environment. We expect the results of experiments that are not bandwidth limited to be similar in simulation and emulation. Any discrepancies can with high confidence be attributed to differences in the software stacks of the simulator and the operating system or in event scheduling and processing, caused by how time is modelled in the simulator. In the real world we expect overall lower performance, caused by the radio and mobility.

Because our scenarios are constructed with a potential path between source and destination at all times, the protocols should under ideal circumstances provide routes resulting in high delivery ratios. The UDP and Ping traffic types do not exhaust the bandwidth and since the data rate is constant, and packets are not retransmitted, we can easily identify the periods of poor connectivity for each evaluation environment. In Table III, the packet delivery ratios for simulation and emulation indicate that the routing protocols can, with little effort and low loss, handle periods of connectivity changes and multi-hop routes under ideal circumstances. AODV and DSR consistently achieve over 91% UDP/Ping packet delivery ratio for all scenarios in simulation and emulation. OLSR achieves slightly lower ratios at 81-91%, which is expected due to its slower convergence. Since the performance is good in both environments we can, with some confidence, exclude any serious routing logic problems, and the physical hardware and protocol stack as significant performance altering factors.

In the real world, on the other hand, we find packet loss concentrated to the periods of connectivity changes, resulting in a decrease in mean delivery ratio of 6-50% and 7-53%, when comparing UDP and Ping with simulation and emulation respectively. TCP is difficult to compare between environments, but we observe smaller real world differences between the protocols. This is because the performance in the real world is dominated by the throughput achieved during periods of low variance in link quality and stable routes, while in the rest of the scenario TCP stalls. The average hop count is in general lower for TCP than Ping (both are two-way), indicating that the majority of the TCP packets are sent over shorter routes where connectivity is good. Therefore, the variance in the results is also low and any advantage of a particular routing strategy never manifests itself for TCP. Roaming node is the exception, where the frequent route updates allow AODV and DSR to excerpt
their convergence advantage over OLSR. For UDP and Ping, the discerning periods are concentrated to when links are fluctuating and immediately after.

Although it is expected to find packet loss mainly occurring during periods of connectivity change, we are first to quantify the difference in loss between AODV, DSR, and OLSR. How well the protocols adapt, and how they are affected by the real radio environment during these periods, are the deciding factors for their overall performance. This observation provides us with a focus for a more detailed analysis of the differences between the protocols.

### C. Performance of Sensing Mechanisms

The design choices of OLSR, AODV, and DSR represent, in the order listed, an increasing willingness to adapt to changes in the connectivity. This is reflected in the measurements presented in Table III. A higher willingness requires more accurate sensing of surrounding events. The performance in the critical regions, established in the previous section, are for OLSR dominated.
by its slow converge. AODV and DSR are, due to their reactive nature, more sensitive to these regions. While AODV reacts quickly to link breaks, it does not react as forcefully to link establishments. AODV’s strategy can be summarized as being happy with the status quo, as long as there are no interruptions in the packet flows. DSR, on the other hand, constantly evaluates the connectivity by promiscuously listening to overheard routing information. It acts aggressively on this information to optimize its performance, even on a per packet basis. Hence, the sensing needs to be accurate to avoid inappropriate actions. The high variance in DSR’s performance in Table III, particularly for latency, is a result of inaccurate sensing. We now look in detail at the accuracy of the sensing mechanisms of each routing protocol.

1) Network Layer Acknowledgments: One reason for DSR’s considerably higher latency standard deviation compared to the other protocols is that nACKs increase channel contention. Such self-interference has been reported by Draves et al. [6], but in that case for static multi-hop ad hoc networks. The interference is higher for Ping and TCP compared to UDP, because nACK-pairs are sent in both directions on each link. Whilst the ACK request is piggybacked on data, the ACK is not. The interference could be reduced by also piggybacking the ACK, but that is an optimization and only works in case there is traffic in both directions.

The impact of self-interference on latency is evident in both simulation and the real world, but is far more severe in the latter case. Figure 3 illustrates the self-interference in simulation by comparing nACKs to link layer feedback.

The high variance affects DSR’s ability to derive a proper retransmission timeout (RTO), leading to premature route timeouts. Simulation and emulation are less affected by this problem due to a partial or modelled radio channel. There are two other factors that explain premature timeouts. First, by borrowing its RTO calculation from TCP, DSR inherits its inability to derive optimal RTOs from RTT measurements in wireless networks. A difference is also that TCP estimates an end-to-end RTT, whilst DSR estimates a per link RTT that fluctuates more. Second, when a premature link timeout occurs, packets that are either salvaged or buffered during route discovery, will be sent more or less back-to-back when the link is re-established. This causes another type of self-interference, which also increases latency variance.

2) HELLO Messages: AODV and OLSR use broadcast HELLO messages to perform link monitoring. Previous work has reported problems with using HELLO messages to determining link connectivity [4], [13]. The main cause is the difference in transmission range between broadcast and unicast. Another consequence of broadcast is that HELLO messages are sensitive to interference and hidden terminals, likely to be frequently occurring during multi-hop configurations. Delayed or lost HELLO messages cause temporary route breaks, leading to route discovery and increasing latency. Although this causes a slight increase in delay and occasional loss for CBR traffic, TCP is affected more severely because it might go into a timeout. In Figure 4 we see that AODV achieves virtually no TCP progress in the beginning of the End node swap scenario. Our corresponding simulation results (see Figure 8) do not reveal any discrepancies between the protocols at this time. To
explain the real world behavior we thus manually inspect our log files during the relevant time period. We find that in seven out of ten runs there are lost HELLO messages between the node pair 3 and 2, the seconds following the start of data traffic at time 31s. The slow start in TCP builds contention, which is emphasized by the adjacent hops. Hello messages collide with transmissions further down the path due to the hidden terminal effect. When TCP starts at one hop as in the Roaming node scenario there is no such interference and TCP proceeds in slow start without interruptions. We have observed similar problems with HELLOs in OLSR, but the effect is less prominent and OLSR seems more resilient, possibly due to its use of link hysteresis. Therefore, OLSR makes steady TCP progress in the beginning of all scenarios.

3) Promiscuous Sensing: The active sensing alone cannot explain the latency we see in the DSR experiments. Figure 5 shows the average RTT in the Relay node swap scenario, which tops out at 10 seconds. After the route switch at 90 seconds, DSR experiences escalating RTTs. The 10 seconds RTT mathes the maximum

timeout value for DSR route discoveries used in our implementation. The question is why does DSR try so hard to discover new routes in the Relay node swap scenario, while AODV and OLSR seem to fare better? For this scenario, DSR is always the worst performer in the real world, while it is often the best in simulation and emulation. Since performance is good in both simulation and emulation it must be the radio and mobility that make this problem manifest itself. To understand the effect of radio and mobility, we plot the source routes of all data packets at the sender and receiver (node 3 and 0, respectively) in Figure 6.

The source routes in simulation are consistent with the scenario and between the source and destination. In the real world, however, node 3 sticks to the non-functioning route 3-1-0 after the relay nodes have swapped positions. Packets are still reaching node 0 because node 1 salvages the packets it receives from node 3 and updates the source route to a functioning one. At the same time node 1 sends route errors to node 3, informing it of the broken route. Node 3 initiates route discovery but soon switches back to the broken 3-1-0 route, because it receives bad routing information. The reason for this is that the DSR’s excessive buffering, both during route discovery and in route maintenance, leads to queue build-ups in the lower layer buffers (e.g., link layer). During a link break, DSR is unable to access lower layer buffers to remove or salvage the packets that now have obsolete source routes using the broken link (e.g., at node 1). When the obsolete packets are transmitted, some nodes will promiscuously sense those packets and add the non-working link state in the source route to their own link cache, because they will not know that the packets are never received. We call this phenomena

link cache poisoning. The poisoning leads to a bad cycle of RERRs, route discoveries, more buffering and hence more poisonous link state. In Figure 7 (left), we were able to recreate this behavior in the simulation by adding a one second delay to some randomly chosen data packets during the simulation run. This simulates queueing delay which is otherwise rarely experienced in

![Fig. 5. Real world Relay node swap round trip latency for Ping traffic. The RTT is averaged over 500 ms windows.](image)

![Fig. 6. Source routes of DSR ping packets in the Relay node swap scenario with Ping traffic. The real world (left) and simulation (right).](image)

![Fig. 7. Link cache poisoning is apparent in simulation when queueing delay is simulated using a 1 second delay on random packets (left). Applying our antidote solves the problem (right).](image)
simulation because of the simple time model and good connectivity.

It is evident that DSR’s promiscuous sensing is too aggressive at using overheard information. In an effort to tune the sensing we modified DSR to only cache links that the data packet containing the source route has already traversed\(^2\). We repeated the simulation with the one second packet delay and found that this antidote mostly solved the link cache poisoning, as seen in Figure 7 (right). However, the efficiency of the antidote in the real world is still unknown and requires further investigation.

**D. Routing Efficiency**

Routing efficiency is determined by two aspects – the time needed to react on a link break and the ability to optimize to a shorter route when one is available. The routing protocols have different mechanisms to react on changes in the network topology. All three protocols optimize their routes by minimizing the number of hops needed from source to destination. A shorter route in general means less interference in the network, higher bandwidth and lower latency. The protocols are, however, more or less sensitive to the radio environment’s impact on the stability of links during periods of connectivity changes. Aggressive optimization can, for example, result in packet loss due to premature routes.

It is not possible to directly compare routing efficiency in simulation and the real world for only one routing protocol because the timings for the environments are different. However, we can compare simulation and real world by looking at the performance of the protocols relative one another. This gives an understanding of how well they handle the radio environment. In Table III the average hop counts in simulation and the real world are quite consistent. The true meaning of the average hop count cannot be understood unless the configuration of the scenario and the delivery ratio are factored in. For example, OLSR experiences more packet loss during the multi-hop configurations of Roaming node than the other protocols. It has a lower average hop count since the packets successfully sent over one hop are dominant. Average hop count can, despite ambiguity, give an indication of the efficiency of each protocol. In simulation the trend is clear, DSR is the most efficient protocol in terms of shortest path routing. OLSR appears to be slightly better than AODV, but a comparison is difficult because OLSR’s slow convergence has the effect that few packets are sent on routes that only exist for short time periods (< 10s).

DSR is efficient in terms of hop count because it has *automatic route shortening* and therefore evaluates the route in each packet. OLSR’s proactive nature makes it always converge to the shortest routes, but until convergence there is a possibility of non-optimal routing. AODV often uses non-optimal routes because it has no dedicated mechanism for optimization and uses the same route until it breaks. HELLO messages can, however, if enabled act proactively and optimize routes when a node receives a HELLO message from the destination node. HELLO messages sometimes outperform link layer feedback as shown in Figure 8. AODV with link layer feedback never achieves a lower hop count than three as indicated by the constant 3 hop throughput. There is a route break around 110s, but at that time the shorter route from time 60s is no longer available. Therefore, AODV (LLF) always has the same throughput. With HELLO messages, AODV uses the shorter route.

![Fig. 8. TCP time sequence number trace from simulation showing the route optimization behavior in the End node swap scenario.](image-url)

In Figure 9 we plot all the UDP packets sent during the ten runs of the Roaming node scenario. The packets are categorized as successfully received, lost or unoptimally routed over the periods of one hop, two hops and three hops. The figure shows the routing efficiency of each protocol. Packet loss is concentrated to periods of connectivity changes. Note that since packets from all runs are overlayed the loss appears longer and more severe due to time shifts in the loss from one experiment to another. OLSR suffers more loss than AODV and DSR because of its slower convergence. However, although

---

\(^2\)The DSR RFC only lists this restriction for packets containing a RREP extension.
OLSR is slow to converge it is quick to optimize routes. DSR is too quick in optimizing the routes. It often chooses premature and unstable routes, causing route flapping between the longer and shorter route. The route flapping occurs because the automatic route shortening tries to optimize the route as soon as a single packet is promiscuously overheard by a node further down the path. If the new link is not stable yet it will soon time out and the longer route is discovered again. The cycle then starts again. The route request flooding following the route flapping increases overhead and contention in the network. Automatic route shortening sometimes works to DSR’s disadvantage at times a longer but more stable route is selected as shown at time 50 s in Figure 9. The optimization will cause route flapping when it switches back to the shorter but lower quality route. In simulation, the radio model works like a binary switch and proves perfect for DSR’s route shortening because as soon as an optimization can be made, connectivity is perfect.

AODV’s route optimization behavior for Roaming node is visible at time 110 s. The route between node 3 and 0 is not optimized until node 3 receives a HELLO message from node 0, i.e., the destination. The reason some packets take the shorter route is that during some of the ten experiments fluctuations cause a timeout that triggers a route discovery during the period between 110 s and 120 s. The optimal route is then discovered. This is actually a situation when the real world radio environment is an advantage over the stable and predictable model in simulation. At least in a minimal hop count sense.

IV. RELATED WORK

Grey et al. compare in [7] the routing protocols APRL, AODV, ODMRP, and STARA in a thirty-three node outdoor testbed. They use GPS to collect movement traces from experiments in an open field where 40 people walk around randomly with laptops. The traces are later fed into a “tabletop” emulation and a simulator. The simple radio models yields acceptable results for open fields but this is not true for an indoor corridor environment. They use direct execution to allow protocols developed in simulation to run in the real world, similarly to the work of Saha et al. [18] as well as the nsclck [14] project. In contrast to our code, packets are forwarded in user space and separate event-loops and scheduling increase the overhead. While our approach is scenario based, they instead use a larger scale network with random mobility and random traffic using only UDP. Each routing protocol is run separately and subjected to different mobility and traffic. Therefore, it is not feasible to compare them side-by-side. Their focus is instead on validating different propagation models in simulation, which is the topic of a follow-up paper by Liu et al. [11].

The authors conclude that it is possible to achieve fairly accurate results using simple radio models. However, the open field scenario they use in their validation is not likely to reflect realistic settings in comparison to more complex environments, e.g., indoors.

Haq and Kunz [8] have evaluated OLSR using two different simulators as well as an emulated testbed. They study the total number of successfully transmitted packets using CBR traffic (UDP) at two different rates and two different packet sizes. The authors use a single scenario with five nodes and report that at low traffic rates, testbed results match closely with those from simulation. However, at higher rates they see very significant differences. Apart from providing a much more extensive study in terms of ad hoc routing protocols, scenarios and traffic types we compare simulation, emulation and real world testing. Haq and Kunz have further only studied the total number of packets received whereas we look at protocol behaviors during the whole scenario and also report on latencies and protocol overheads.

Johnson [9] recorded traffic traces from laptops, running DSR, mounted in cars whose positions were constantly logged using GPS. Several different traffic types were used and the collected data drove simulations
as well as emulations. The author believes that simply comparing the average number of received packets from simulations and real experiments does not provide enough information to answer the question of how closely emulations come to reproducing simulation results. It can even produce an incorrect conclusion. He therefore suggests studying time-sequence number plots as well as other performance metrics over time. In our work we use different performance metrics over time and compare simulations to emulations but also to the real world.

V. CONCLUSIONS

Our contributions in this paper can be summarized in three categories.

First, we provide a step-by-step description on how to perform structured and repeatable real world experiments on protocols for wireless multi-hop networks. An important part of this experimentation is the feature of running complementary simulations and emulations, using the same protocol implementation and topological changes as in the real world tests. That way, packet delivery and time-sequence number graphs can be compared to quickly identify discrepancies between, e.g., simulation and the real world. The simulation runs thus constitute an idealized baseline case. This comparison requires careful coupling of the environments and we have presented how it can be achieved.

Second, as a proof-of-concept we report on the origins of two important problems found in real ad hoc networks. One is the performance problem for TCP which we could trace to the use of HELLO messages for ad hoc neighbor discovery. The other is the large latencies in real networks running DSR; something which is caused by the extensive buffering in combination with route flapping. The optimistic buffering strategy gives a better delivery ratio in artificial environments. For real applications, however, data often has a best-before date and it is not always acceptable to trade low latency for higher total delivery ratios.

Third, we summarize general quantitative observations that can be made from our measurement data, resulting from experiments constituting 270 individually performed real world experiments. This body of data (2 Gbytes) will be made available to other researchers. The public availability of all software then enables further data analysis studies to be conducted.

REFERENCES


