# A Large-scale Testbed for Reproducible Ad hoc Protocol Evaluations *

Henrik Lundgren†, David Lundberg†, Johan Nielsen‡, Erik Nordström†, Christian Tschudin†

November, 2001

## Abstract

We have built an Ad hoc Protocol Evaluation testbed (APE) in order to perform large-scale, reproducible experiments. APE aims at assessing several different routing protocols in a *real world* environment instead of by simulation. We present the APE testbed architecture and report on initial experiments with up to 37 physical nodes that show the reproducibility and scalability of our approach. Several scenario scripts have been written that include strict choreographic instructions to the testers who walk around with ORiNOCO equipped laptops. We introduce a metric called *Virtual Mobility* that we use to compare different testruns. This metric is based on the *measured* signal quality instead of the geometric distance between nodes, hence it reflects how a routing protocol actually *perceives* the network's dynamics.

## 1 Introduction

Ad hoc networks consist of a number of wireless mobile nodes that dynamically form a network without any pre-existing communication infrastructure. Each node also serves as a router, forwarding its neighbors' traffic. As they may move arbitrarily and unpredictably, the network's topology will be subject to constant changes.

Conventional routing protocols used in the Internet today perform poorly under such circumstances. There exist several ad hoc routing protocol proposals both inside and outside the IETF WG MANET [1]. Some of these proposals are evaluated through simulations like in [2, 3] and a few of them through *small-scale* tests [4, 5, 6]. Still there is a lack of *large-scale* experiments. The APE testbed aims at filling this gap.

In simulations, conditions and scenarios can easily be altered on a per-variable basis in order to analyze specific behaviors of a protocol. What cannot be satisfactorily modeled in a simulation are the dynamic environments with complex signal propagation patterns in which ad hoc networking takes place. Evaluations of ad hoc routing protocols are incomplete without real-world experiments. The goal with the APE testbed is to assess routing protocols in a controlled manner in the real-world. Such results will provide valuable insights for simulation modeling and traces with real-world signal propagation information could be fed into the simulation environments to further validate existing models.

Several issues must be taken into consideration when building an ad hoc networking testbed. Most importantly, it should be possible to setup and reproduce testruns such that we obtain comparable results. To extend the characterization of repeated testruns a metric for capturing the nodes' movement is introduced. This metric calculates the changes in "virtual" distance between nodes in the network. Hence, we call this metric *Virtual Mobility (vM)*.

These changes are computed from signal quality measurements, which do not only reflect physical movement but also any influences due to the inherent radio properties like interference, reflection and multi-path fading. Ideally, we will be able reproduce testruns to assess statements like "under Virtual Mobility v, protocol A behaves better than protocol B". However, such a comparison is beyond the scope of this paper due to the low availability of working implementations of routing protocols. Nevertheless, we will show that this is possible with the APE testbed as more implementations become available.

When building an ad hoc networking testbed, it is also crucial that the installation and handling of the testbed software is as easy as possible when running experiments with a large number of people. In the APE testbed, this is facilitated by different scenario configuration files containing per-node movement instructions and automatic traffic generation.

* A shorter version of this paper was accepted for publication in the proceedings of IEEE WCNC'02.

† Uppsala University, Sweden

‡ Ericsson Research, Switchlab, Sweden

## Related Work

Toh et al describes in [5], outdoor experiments with four stationary nodes. Their focus is on how different network conditions like beaconing interval, route length, and packet size affect communication performance. Maltz et al describes in [4] their effort in performing outdoor experiments with 2 stationary and 6 car-mounted mobile nodes. Maltz et al set up a construction site like scenario and simulates a Push-To-Talk voice system. For positioning of the mobile nodes they use GPS. In [6] Ramanathan and Hain use specific hardware to set up experiments with 5 stationary nodes and report on throughput and delay measurements using Ping and report on testing with up to 10 nodes. They used a software emulation of the radio and a model of the channel when running larger settings.

All the above testbeds use only their own developed routing protocol. Two of them use specific hardware. Two of them run only static settings with stationary nodes, while the one with mobile nodes only run one specific setting.

In the APE testbed the kernel and the routing protocol can easily be replaced. This fact together with the strict choreography approach and the Virtual Mobility metric make it possible to perform *different mobile scenarios using different routing protocols and obtain comparable results*.

The rest of the paper is outlined as follows. The architecture and implementation issues of the testbed are presented in Section 2. Section 3 describes the computation of the virtual mobility metric. In Section 4 the physical environment is described together with three tested scenarios. Section 5 discusses the analysis of the experiments. Finally, we conclude and outline future work in Section 6.

# 2 APE Architecture and Implementation

The APE testbed comprises tools for choreography configuration, network interface data collection, and uploading of all measurement results to a central machine. We have written analysis tools for post-run analysis of the uploaded results. Any routing protocol implemented under Linux can be inserted into the testbed. We have successfully run the APE testbed with three different routing protocols; AODV [7, 8], OLSR [9, 10] and TORA [11, 12]. To run experiments we first design scenarios and make choreography scripts. Next, we run the experiments and gather data. The analysis of gathered data is performed after the experiments. Figure 1 shows an overview of the APE software that runs on each node during a testrun. The choreography script interpreter parses scenario files (see Figure 2) and controls the start up of traffic generators and a time synchronization application. A modification to Lucent Technologies' ORiNOCO driver [13] enables to log signal qualities from *all* packets it can detect.

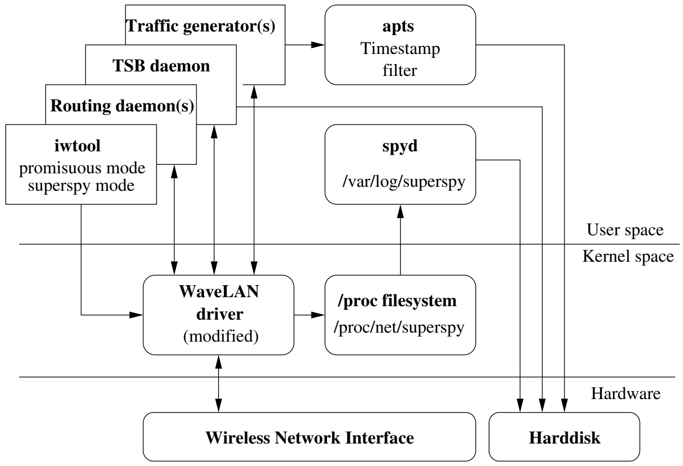


Figure 1: Overview of software components running on each node in the APE testbed.

## 2.1 Choreography and Data Traffic Generation

The scenarios are configured such that each participant receives individual instructions. The scenarios are strictly choreographed; after the initial "ready-set-go" testbed participants only need to follow the instructions that appear on the screen on when and where to move [1]. Hence, the participants do not have to be familiar with the testbed and the exact purpose of the experiment.

Currently, the traffic is generated by the well-known Ping command, i.e. 64 byte ICMP echo requests. The intensity, mode (unicast/broadcast) and payload size can be altered depending on the scenario. Other types of traffic in the network are beaconing traffic from a time synchronization application and from the routing daemon. Different traffic generators can be inserted independently and would then simply be launched from within the choreography script.

[1] We call the choreographed movement of a participant in the APE testbed a "monkey walk".

```
node.0.action.0.msg=Test is starting...
node.0.action.0.command=start_spyd
node.0.action.0.duration=1
node.0.action.1.command=ping_node 1 660
node.0.action.1.msg=Stay at this location.
          You are pinging node 1. (30sec).
node.0.action.1.duration=30
node.0.action.2.msg=Start moving! Go to the
          end of House 1 (DoCS). (75 sec)
node.0.action.2.duration=75
```

Figure 2: Excerpt from the scenario file for a Lost'n'Found scenario. (see Section 4.2 for scenario description)

### 2.2 Data Gathering Tools

Data is timestamped and logged both at the Ethernet layer and the IP layer. Using a user-space utility, the ORiNOCO driver is set in promiscuous mode to enable our spy routine. We get the MAC address from the Ethernet frame, and signal and noise levels from ORiNOCO driver. We timestamp this information with the time of packet arrival and log this in a buffer in the /proc filesystem. A user-space daemon regularly transfers this data to permanent disk storage.

At the IP level each node logs a timestamped result of its own Ping traffic. Information that can be extracted from Ping traffic includes the route of each packet and its reply, round-trip time, and packet losses. At the end of an experiment the local log files are uploaded to a central node.

### 2.3 Data Analysis Tools

In a first step, scripts are used to merge all log files and to adjust them for clock skews. This results in single log files for the ethernet level as well as the IP packet level where all entries are globally ordered in time. A separate "time stamp broadcaster" application (TSB) was implemented in order to enable log entries to be synchronized during the analysis phase. Each node is running the TSB application during the experiments and log the TSB packets from all its neighbors.

Analysis scripts enable to extract a testrun's salient features as for example the virtual mobility metric, connectivity in terms of number of neighbors, link changes per second, ping success ratio, hop count and path optimality.

A log-driven animation tool called APE-view was written that allows to replay scenarios by positioning the nodes on the screen based on the logged signal quality between node-pairs (see also Section 5.1.2).

### 2.4 Testbed Distribution and Set-up

The APE platform that runs on each mobile node during all the experiments is based on a customized version of the Red Hat Linux distribution. To be able to handle a large number of nodes (a number of tens or more) for a testrun the administration and configuration of each node must be very fast and simple. In APE everything is pre-configured except the node identity. The testbed participants can handle the installation themselves. For flexibility the distribution is designed to be installed on a host computer normally running either the Windows98 or Linux operating systems.

In Windows98 mode no harddisk re-partitioning or other OS re-configuration is needed. The testbed participants simply download a self-extracting file and execute one of the included program which make the computers to reboot to Linux, using the pre-made filesystem image included in the distribution.

To set up the testbed from within a Linux environment is equally simple. After download and extraction of the distribution file, the testbed participants only need to run a single script that adds a boot option in the Linux boot loader and makes the computers reboot to the testbed environment.

To build a completely new distribution the first steps are to choose a Linux kernel to base the testbed upon, customize it with given patches, and compile. Next step is to pick the routing protocol to use and follow that implementation's specific installation instructions. The last steps are to make sure that the scenario-scripts and traffic generators are present in given directories and to call the script that combines these pieces with other given testbed files and applications into a complete distribution.

The quick and easy installation process and the flexibility to switch kernel and routing protocol makes it possible to easily set up and repeat experiments using not only different user-space routing protocol implementations based on different kernels, but also different kernel-space routing protocol implementations.

## 3 Capturing Mobility in Ad hoc Networks

Several mobility metrics have been described in the literature. For example, Hu and Johnson [14] describe a couple of ways of calculating the mobility metric. The simplest model is called *minimal route change metric* and is calculated as the number of route changes, where a route only changes when it breaks. High node mobility results in higher number of link breaks during a scenario, which reflects the physical mobility within an ad-hoc network. On the other hand, if the quality of a link alters due to node movement, but the link does not break, this model would yield no degree of "mobility" at all.

Johansson et al describe in [3] a geometric mobility metric:

$$\frac{2}{N(N-1)T}\sum_{i=1}^{N}\sum_{j=i+1}^{N}\int_{t=0}^{T}\left|\frac{d\|P_j(t)-P_i(t)\|_2}{dt}\right|dt$$

where $P_k(t)$ is the position of node $k$ at time $t$, T is the length of the scenario duration and N is the number of nodes participating. The sum is calculated over all node pairs over the scenario duration.

Hu and Johnson [14] also used the geometric mobility model introduced by Johansson et al. However, Hu and Johnson calculated this mobility metric exactly whereas Johansson et al used an approximation in their simulations.

The geometric mobility metric described above is based on the nodes' physical positions, and how they change positions. In a simulation environment the nodes' positions are known at all times. To achieve this in the real world we would need to equip each node with GPS. However, GPS is limited to outdoor environments. Techniques like triangular positioning would work in an indoor environment but add extra complexity in terms of dedicated nodes that must deduce directions from transmissions. Still, none of these techniques would capture the most important aspect: the signal quality of links as *perceived* by the participating nodes. After all, it is the real world signal propagation dynamics that form the wireless landscape in which the routing daemon will operate.

## 3.1 A "Virtual" Mobility Metric

Background noise and obstacles can influence the signal quality. Even very small changes in location can lead to big changes in signal quality. Instead of using the real distance between two nodes, we propose to compute a distance from the perceived signal quality. In this way we get a metric that captures the real world dynamics as perceived by the nodes. We will use this mobility metric to extend the characterization of different testruns.

### 3.1.1 Path Loss Model

The definition of our virtual Mobility metric is based on "virtual" distances which we based on the *measured* signal quality. We used the path loss model mentioned in [15] to relate signal quality and distance: for the far field case (indoor) it proposes a path loss coefficient of 3.3:

$$Q \text{ in dB} = \alpha - 33 * \log(\text{dist}/\beta) \qquad (1)$$

which, after calibration with the signal quality range of the WaveLAN card and our measurements, results in the following inverse path loss formula:

$$D_j(node_i) = 4 * 10^{\frac{40-0.9*Q_j(node_i)}{33}} \qquad (2)$$

where $Q$ is the WaveLAN signal quality (0. . . 75) for a packet received from node $j$ at node $i$. $D$ is in the range of 0.5 to 65 m. We only considered the far field model as our experiments focus on large movements and long distances with nodes going out of reception range.

### 3.1.2 Calculating vM

Virtual Mobility between $node_i$ and $node_j$ is calculated for $node_i$ as follows. For a given time interval $t_k$ we average the virtual distances obtained from all packets heard from a specific $node_j$. We define $D_j^k$, the mean virtual distance to $node_j$ for time slot $t_k$, as

$$D_j^k(node_i) = \frac{1}{N_j^k}\sum_{a=1}^{N_j^k} D_j^a \qquad (3)$$

where $N_j^k$ is the number of packets received from $node_j$ during $t_k$ and $D_j^a$ is the virtual distance obtained from the signal quality of packet $a$ that was received from $node_j$ during interval $t_k$.

The virtual mobility $vM$ for $node_i$ with respect to $node_j$ for time interval $t_{k+1}$ is simply the change in mean virtual distance, namely

$$vM_j^{k+1}(node_i) = |D_j^{k+1}(node_i) - D_j^k(node_i)| \qquad (4)$$

and the average virtual mobility perceived by $node_i$ at time $t_k$ is

$$vM_{avg}^k(node_i) = \frac{1}{S}\sum_{l=1}^{S} vM_l^k(node_i) \qquad (5)$$

where S is the number of nodes vM is calculated over.

Finally, let *network* virtual mobility for time $t_k$ be

$$vM^k = \frac{1}{N}\sum_{i=1}^{N} vM^k_{avg}(node_i) \qquad (6)$$

where N is the number of nodes in the network.

This definition yields the *mean* network vM-value at every time interval. It represents how an average node moves during a test.

In order to examine the heterogeneity between how different nodes move it is useful to display not only the mean vM-value, but also the upper and lower quantile. If the link qualities change in a homogeneous way the upper and lower quantile will be close to the mean value. On the other hand, if some link qualities change very slowly and other link qualities change rapidly, the upper and lower quantile will be more separated from the mean value, implying a more heterogeneous movement within the network. By looking at the upper and lower quantile of virtual mobility values we get a way of capturing different movement patterns in the network.

#### 3.1.3 vM variations

We have presented a specific way of how to calculate vM considering the perceived link quality from one-hop neighbors only, i.e., there is no aggregation over multiple hops for distant nodes. Also, time is divided into small intervals and all link qualities captured from a specific node are averaged over such an interval. Finally, we use *all* link qualities perceived by each node in the calculation. Several alternative variations could be considered, like only use links that are used by the routing daemon, give different links different weight and take network partitions into consideration when calculating network vM.

## 4 Experimental Set-up

### 4.1 Physical environment

We have created scenarios for both outdoor and indoor environments. To stress the vM metric with as complex signal propagation patterns as possible we here choose to focus on the experiments performed indoors. The indoor experiments took place in three interconnected buildings on campus shown in Figure 3. The three buildings have a combination of offices and lectures halls. The white paths through the buildings are the corridors and bridges connect the buildings. The letters [A…H] marked in the corridors refer to specific spots where nodes will be placed during different scenarios described below. The corridor lengths are 56 meters in Building 1 and Building 2. The Middle Building has a corridor length of 28 meters and the bridges are 17 meters each. The width of corridors and bridges varies between 2 to 4 meters. The walls within the buildings are very thick and radio signals are effectively damped when losing line-of-sight. A radio signal sent from a node at one end of Building 1 will not be detected at the other end of the same building. The nodes move at normal walking speed, in this case meaning slightly above 1 m/s.

### 4.2 Scenario descriptions

Three scenarios will be described in detail below for which we will show measurement results in Section 5. Figure 4 in conjunction with Figure 3 gives an overview of the movements within these three scenarios.

#### 4.2.1 Lost'n'Found

The purpose of this scenario is to examine how different movements and link breakages can be captured by the vM metric. In this scenario a group of nodes splits into two clusters, lose radio contact for some time and then join again. It is choreographed in the following way. Assume a group of nodes standing at point D at the start of the experiment. After 30 seconds half of the group starts moving away towards point A, while the other half remains at point D. At some point before the first cluster reaches point A the two clusters will lose radio contact. At time 75 the first cluster should have reached point A. They will remain at point A until time 105 and then start moving back. The two cluster re-join at point D at time 150 and will remain there for 30 seconds before the experiment ends. The traffic in this scenario consists of all nodes sending broadcast Ping during the duration of the experiment.

#### 4.2.2 Double Lost'n'Found

Besides examine movement and link breakages as in the Lost'n'Found mentioned above, this scenario aims at examine how different movement patterns are visualized by the vM metric. In this scenario there are two different moving clusters that are "lost" simultaneously but "found" at different times. Assume a group located at point E. After 30 seconds the group splits into three equally sized clusters. Two clusters start moving away in opposite directions, the first cluster towards point A and the second cluster towards point H. The third cluster

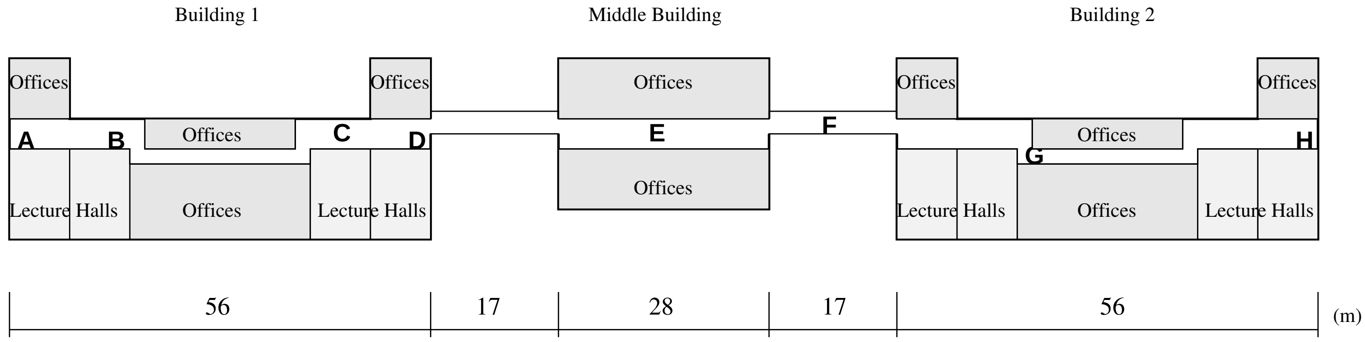


Figure 3: The physical test environment. The white paths are corridors and the letters [A...H] refer to positions used in the scenarios.

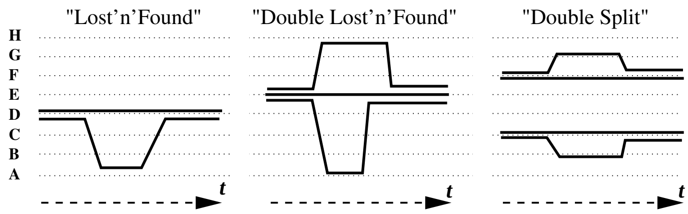


Figure 4: Time-space diagrams for the three experimental scenarios discussed in Section 4.2.

remains at point E. On their way towards their respective destinations they will lose radio contact with the other clusters. At time 105 the two moving clusters should have arrived at their respective remote places. At time 135 the first cluster starts returning to E and should arrive there at time 210 according to the instructions. The second cluster starts returning at time 225 and should have arrived at E at time 300. The experiment ends at time 330. The traffic in this scenario consists of unicast Pings from each cluster to the other two clusters.

#### 4.2.3 Double Split

The goal with this scenario is to create a multi-hop setting and to stress the routing daemon with weak and fluctuating link qualities. Assume two groups of nodes, one located at point C and the other at point F. There should be radio contact between point C and point F. After 50 seconds the two groups split into equally sized clusters. One of the clusters located at C starts moving towards point B, and one of the clusters located at point F starts moving towards point G. At time 70 the two moving clusters should have arrived at their respective destinations and will remain there for 50 seconds. During the stay at the remote destinations these clusters will only have radio contact with the other neighbor cluster at their respective place of origin, i.e., each group has radio contact with only their adjacent group(s). At time 120 both remote clusters start moving back towards their respective place of origin and the other cluster. The moving clusters will be back at time 140 and remain there another 50 seconds before the test ends. The traffic in this scenario consists of unicast Pings from each cluster to the other three clusters.

## 5 Results

We will present results from repeated experiments with the number of participating nodes ranging from 19 to 37 nodes using the Madhoc-AODV implementation. The focus will be on using the *vM* metric as a tool for capturing movement within the network and comparing repeated testruns. The size of some of the log files used to extract these results reached 70 MB and contained timestamped entries for more than 3 million packets

### 5.1 vM Analysis

The figures in this section show the network virtual mobility (m/s) as a function of time (s). Each figure contains three graphs. The solid lines represent the mean value. The virtual mobility of the 25% of the nodes that move the most (upper quantile) and the 25% that move the least (lower quantile) are represented by the dotted lines and dashed lines respectively. These are here called vM-high and vM-low.

Although all nodes at some point are standing still according to the choreography, the vM value will indicate small movements. This is simply the normal fluctuation of a radio signal between nodes closely standing together. Very small movements by participants, like turning around, will also cause signal quality changes.

### 5.1.1 Characterizing a Scenario with vM

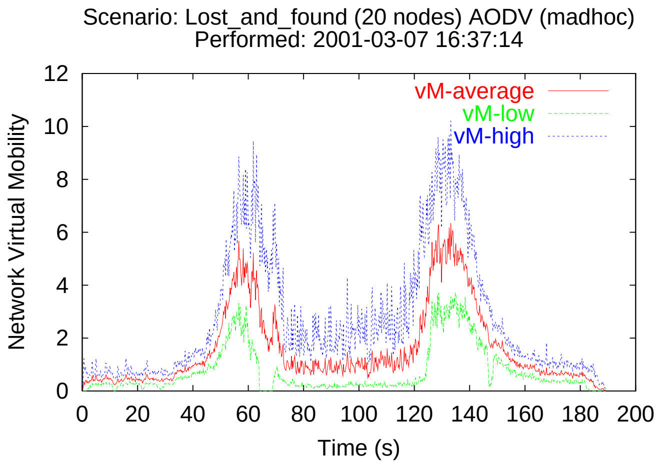


Figure 5: The virtual mobility for scenario *lost'n'found* with 20 participating nodes.

Figure 5 shows vM for a Lost'n'Found scenario with 20 nodes. As can be seen in the graph there are two significant peaks at time 60 and at time 125. These correspond to when the two groups start losing and regaining radio contact with the other group. During the time when the two groups are out of radio range of each other the mean curve is back to approximately the same level as when the two groups were standing right next to each other.

### 5.1.2 Deriving Topological Positions from Signal Quality information

The result that vM visualizes in an aggregated form can also be applied to single nodes: based on signal quality data we can recreate a topological correct configuration of an experiment. For this, we position nodes such that the mismatch between geometric distance and computed virtual distance is minimized. This allows us to visualize logical connectivity for each node and provide an intuitive background for understanding metrics like virtual mobility, packet loss, and optimal route set-up. In Figure 12 we have taken a snapshot of APE-view every 25 seconds during a replay of the Lost'n'Found scenario. We see that at time 25 the nodes are still standing together, while at time 50 half of the nodes have started to move away. At time 75 and 100 the nodes are out of radio contact of each other (no connectivity lines between nodes in the different groups) and at time 125 we see that the two groups start to regain contact. The snapshot at time 125 corresponds to the second peak in Figure 5. The next snapshot is at time 150, the time when the moving group should have returned according to the choreography. The last snapshot shows the view 5 seconds before the testrun ends. As expected this is similar to the first snapshot.

### 5.1.3 Discerning Different Movement Patterns Among Nodes

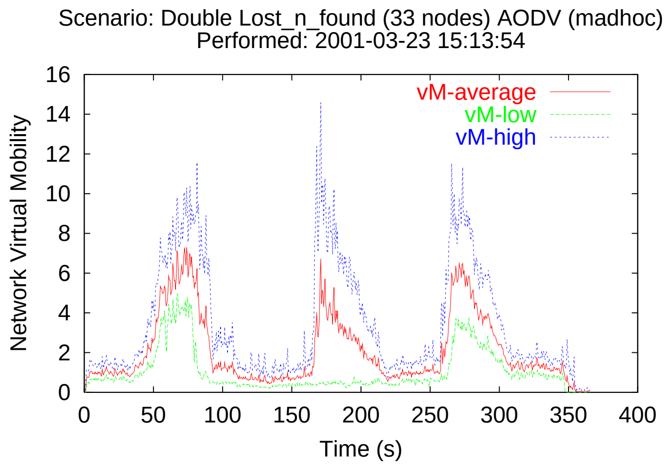


Figure 6: The virtual mobility for scenario *double lost'n'found* with 33 participating nodes.

Figure 6 shows the vM values for a Double-Lost'n'Found scenario run with 33 nodes. Remember the differences from the Lost'n'Found scenario: there are now three groups where *two* of them are *lost simultaneously* but *found at different times*. These movements are clearly shown in the graph where the three peaks representing the split (at time 80) and the two re-connections (at time 170 and 260) are very significant. The split get the highest mobility factor because the number of nodes that lost their contact with neighboring nodes during the split is higher than the number of nodes that did regain contact with other nodes during the re-connections.

Now turn the focus towards the vM-low curve in Figure 6. Observe that during the first regain, around time 170, the vM-low curve has no peak. This indicates very low mobility among some of the nodes. This is perfectly reasonable because the third group, still standing still at the remote place at this time, did not have contact with any of the other two groups and should not yield any significant mobility. Hence, we can use the vM-low and vM-high in order to examine how homogeneous or heterogeneous the mobility in a network is from the networks' point of view.

### 5.1.4 Reproducing Testruns

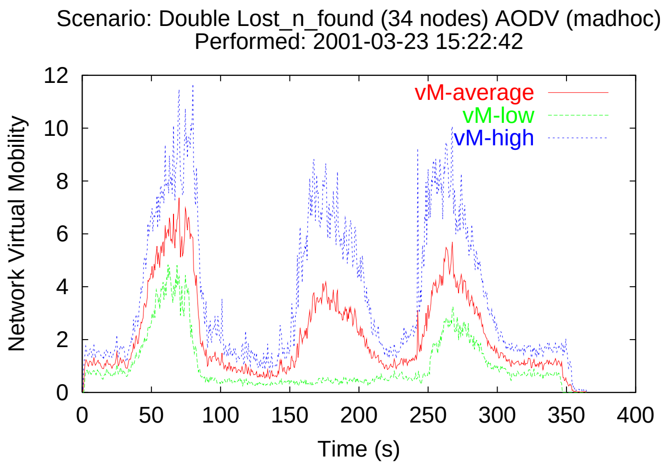


Figure 7: The virtual mobility for scenario *double lost'n'found* with 34 participating nodes.

So far the choreography approach seems to have worked well and the vM metric can satisfactorily represent different movements during a testrun.

In order to examine if we can *reproduce* results we repeated testruns several times. We show here the second run of the Double Lost'n'Found scenario from Figure 6. The only difference to the first testrun is that the second testrun comprises 34 nodes instead of 33 (which was due to a computer crash). The participants get exactly the same instructions in both testruns and thus should move in the same way. We use vM to extend the characterization of the testruns. Comparing our first testrun (Figure 6) with the second testrun (Figure 7) we can see that the overall shape of the curves match well and to some extent also the value of the vM. We conclude that our choreographed approach ensure that the participants move in the same fashion and that vM can reveal if the signal quality fluctuations in two testruns are similar enough to make results from further performance evaluations comparable.

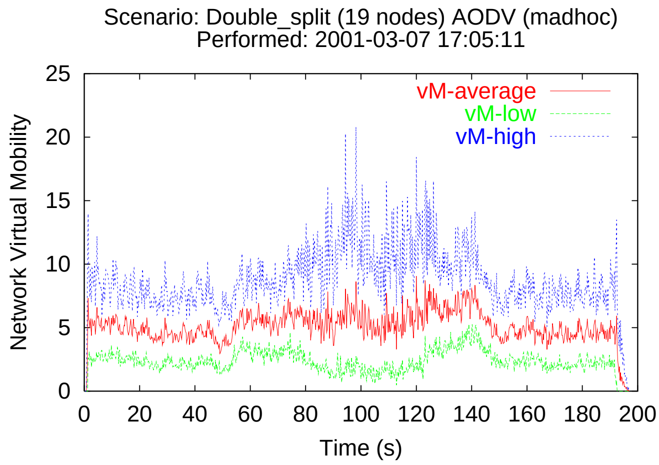


Figure 8: The virtual mobility for scenario *double split* with 19 participating nodes.

### 5.1.5 Influence of Network Density

Figure 8 shows the virtual mobility for a Double Split scenario with 19 nodes. This scenario starts and ends with two groups separated such that the signal qualities of the radio links between the groups are low (time interval 0-50 and 140-190). During time interval 70-120 the network consists of four separated groups that have poor connectivity and only to its adjacent group(s). We observe in Figure 8 that the vM is relatively high during the phases when the nodes do not move compared to such phases in Figure 5, Figure 6 and Figure 7. This is due to the fact that the signal qualities between the groups are low and fluctuate more. The nodes move during time interval 50-70 and 120-140, but that is not shown as clear as in previous scenarios. The high vM values during the "separated" interval (70-120) smooths out the peaks. The vM values during time interval 70-120 are higher than during time intervals 0-50 and 140-190, although the nodes in time interval 70-120 do not move either. The reason for this is that there are more radio links in the network that are low and fluctuating. This indicates that the vM increases with decreasing network "density".

One of the goals when designing this scenario was to stress the routing daemon with multi-hop, and weak and fluctuating link qualities. However, we experienced problems with the Madhoc-AODV implementation. For example, the time for detecting a link breakage and establish a new route varies a lot and can be up to one minute. This is not an acceptable behavior for reasoning about routing performance wherefore we will have to first invest time in debugging the routing daemon.

### 5.1.6 Scaling up testruns

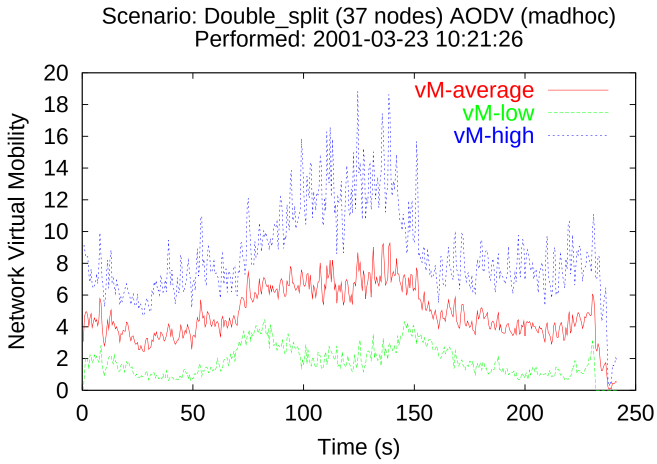


Figure 9: The virtual mobility for scenario *double split* with 37 participating nodes.

The Double split scenario with 19 nodes (see Section 5.1.5) has been scaled up to double its size, resulting in a 37 node setting. Minor differences regarding the duration of movement periods between the 19 node testrun and the 37 node testrun exist but do not affect the overall result [2].

The traffic in this scenario consists of unicast Pings from each cluster to a few of the nodes in all other clusters. In the 19 node setting the Ping intensity was 5 Pings/sec and in the 37 node setting it was 2 Pings/second. Figure 9 shows the vM graphs for a 37 node testrun. Comparing Figure 8 and Figure 9 we see that the overall shape of the curves match well. Because we chose the time interval T over which we average the virtual distances to be equal to to the inter-ping gap, we obtain a much smoother graph for the 37 node experiment (T=0.5) than for the 19 node case (T=0.2). We conclude that it is possible to both reproduce and *scale up* experiments.

## 5.2 Ping success and Hop count analysis

Figure 10 shows the Ping success ratio for the *Double Lost'n'Found* scenario with 34 nodes. As we experienced problems with the Madhoc implementation, and because Ping traffic relies on whether routes are established or not, we could for the moment not assess realistic properties of it. Nevertheless, we can see that the Ping success ratio starts to decrease right after the nodes start to move (compare with Figure 7). At time 65 the slope of the Ping curve becomes steeper which is right in the middle of the first peak in Figure 7. At time 90 no Ping packets get through between the groups. Again referring to Figure 7 we see that this is the time where the groups lost radio contact. At time 160 Ping packets start getting through again reaching a stable level that is maintained until time 250. At this point the second group enters into radio range and all nodes gradually regain full connectivity. Figure 11 shows the connectivity over time and one can clearly see the different phases of the Double Lost'n'Found scenario.

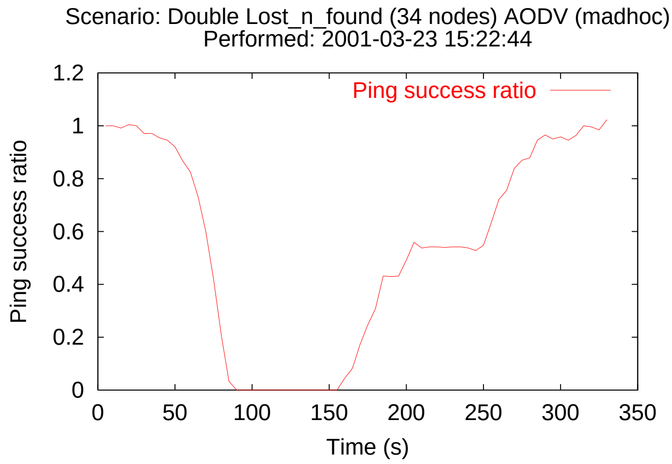


Figure 10: Ping success ratio for the *Double Lost'n'Found* testrun with 34 nodes.

One would expect that – as the two clusters at the edges move apart from each other – they would re-route their traffic via the middle cluster. However, analysis of the hop count reveals that this is not the case (see table below). Even in the Double Split scenario where the clusters retain contact with their adjacent groups, we can see that Madhoc fails to set up multihop paths (see second table):

Number of successful pings classified by the number of hop counts, for Madhoc in the *Double Lost'n'Found* scenario with 34 nodes

| Travel distance: | 1 hop | 2 hops | 3 hops |
|---|---|---|---|
| Request pings | 1127 | 1 | – |
| Reply pings | 1127 | 1 | – |
| Complete pings | n/a | 1126 | 2 |

[2] The times for moving to remote spots were very tight in the 19 nodes setting. Twice as many people walking in the corridor will affect when the last node arrives at the remote spot. Therefore, the walking times were increased by 15 seconds each. Referring to the scenario description in Section 4.2.3, the arrival and departure times should be increased with 15 seconds to 85 and 135, respectively. The arrival time of the return to the original spot should be increased by 30 seconds to 170.

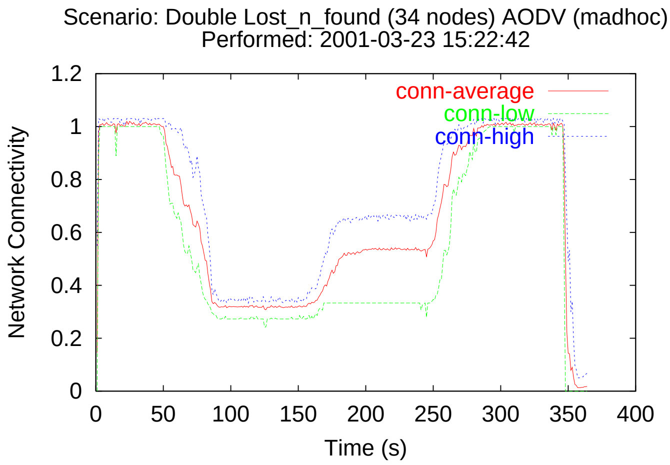


Figure 11: Connectivity for *Double Lost'n'Found* testrun with 34 nodes.

Number of successful pings classified by the number of hop counts, for Madhoc in the *Double Split* scenario with 34 nodes

| Travel distance: | 1 hop | 2 hops | 3 hops |
|---|---|---|---|
| Request pings | 405 | – | – |
| Reply pings | 405 | – | – |
| Complete pings | n/a | 405 | – |

#### 5.2.1 Hop count in OLSR-Inria and Madhoc-AODV

OLSR-Inria [10] is one of the other few ad hoc routing protocols for which an implementation is available. We repeated the Double Lost'n'Found and Double Split testruns, with 9 and 8 nodes, respectively, with both the Madhoc-AODV [8] and OLSR-Inria implementations. In the tables below we list the number of hops that pings were taking during the scenarios. These figures reflect what we experienced intuitively; The OLSR-inria implementation runs smoothly and has no problem to set up multihop connections. Madhoc-AODV only sets up very low number of multihop links, while OLSR-Inria managed to set up 4-hop paths (7 hops when counting forth and back path).

Number of successful pings classified by the number of hop counts, for Madhoc in the *Double Lost'n'Found* scenario with 9 nodes

| Travel distance: | 1 hop | 2 hops | 3 hops |
|---|---|---|---|
| Request pings | 2940 | 1 | – |
| Reply pings | 2940 | 1 | – |
| Complete pings | n/a | 2939 | 2 |

Number of successful pings classified by the number of hop counts, for OLSR-Inria in the *Double Lost'n'Found* scenario with 9 nodes

| Travel distance: | 1 hop | 2 hops | 3 hops | 4 hops |
|---|---|---|---|---|
| Request pings | 3009 | 40 | – | – |
| Reply pings | 2974 | 71 | 4 | – |
| Complete pings | n/a | 2963 | 53 | 33 |

Number of successful pings classified by the number of hop counts, for Madhoc in the *Double Split* scenario with 8 nodes:

| Travel distance: | 1 hop | 2 hops | 3 hops |
|---|---|---|---|
| Request pings | 2962 | 2 | – |
| Reply pings | 2959 | 5 | – |
| Complete pings | n/a | 2957 | 7 |

Number of successful pings classified by the number of hop counts, for OLSR-Inria in the *Double Split* scenario with 8 nodes:

| Travel distance: | 1 hop | 2 hops | 3 hops | 4 hops |
|---|---|---|---|---|
| Request pings | 2892 | 371 | 100 | 15 |
| Reply pings | 2894 | 380 | 98 | 6 |
| Complete pings | n/a | 2870 | 46 | 336 |

| Travel distance: | 5 hops | 6 hops | 7 hops | 8 hops |
|---|---|---|---|---|
| Complete pings | 26 | 86 | 14 | – |

The OLSR-Inria implementation seems stable and capable of setting up multi-hop routes and will be considered for further evaluations in the future, while Madhoc needs more debugging first.

## 6 Conclusions and Future Work

We believe that after many years of simulations it is important to test ad hoc routing protocols in real world settings with a large number of nodes. The testbed that we have built is based on strictly controlled and choreographed testruns which enables us to reproduce and compare protocol performance. In this paper we introduce the concept of virtual mobility (vM) based on changes in the perceived signal quality. This metric allows us to characterize a network's configuration and to assess the reproducibility and scalability of our test environment. As more implementations of different ad hoc routing protocols become available we will plug them into the APE testbed and push to run 50 node experiments. We also plan to use the testbed to run active routing protocols that were developed within the ARRCANE project [16].

Our experiences are that the quality and maturity of available protocol implementation differs heavily and that – before more stable implementations are available – we are merely testing implementation quality rather than protocol functionality. Therefore we believe that it is most important to speed up implementation and debugging of ad hoc routing protocol software. The APE testbed will be very useful as a tool to analyze implementation behaviour and also as a benchmarking tool when implementations have become more mature. Ideally, we will be able to assess statements like "under Virtual Mobility v, protocol A behaves better than protocol B".

Considerable effort has been put into making the installation of the APE software as lightweight as possible. We plan to make the APE testbed distribution available to other research institutions [17]. A long term goal is to feed connectivity information from our real world experiments into different simulation environments. The objective is to improve and validate existing simulation models against real experiments.

## Acknowledgement

We would like to thank Ericsson for donating the wireless WaveLan PCMCIA cards and Per Gunningberg for setting up this project's framework. Many thanks to all members in the Communications Research (CoRe) group and to the students in the Advanced computer networks course for participating in experiments and carrying around laptops.

## References

[1] *The Official IETF working group Manet webpage*, `http://www.ietf.org/html.charters/manet-charter.html`

[2] 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 Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), October 1998

[3] P. Johanson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, *Scenario-based Performance Analysis of Routing Protocols for Mobile Adhoc Networks,* Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), August 1999, pp. 195-206.

[4] D. Maltz, J. Broch and D.B. Johnson, *Quantitative Lessons from a Full-Scale Multi-Hop Ad Hoc Network Testbed.* In Proc. of WCNC 2000, September 2000.

[5] C.-K. Toh, R. Chen, M. Delwar, D. Allen, *Experimenting with an Ad Hoc Wireless Network on Campus: Insights and Experiences* ACM SIGMETRICS, December 2000, pp. 21-29.

[6] R. Ramanathan, R. Hain, *An ad hoc wireless testbed for scalable, adaptive QoS support.* In Proc. of WCNC 2000, September 2000.

[7] C. Perkins, E.Royer and S. Das: *Ad hoc On-demand Distance Vector (AODV) Routing*, Internet Draft, draft-ietf-manet-aodv-09.txt, *work in progress*, November 2001.

[8] *Mad-hoc AODV technical documentation*, `http://mad-hoc.flyinglinux.net/techdoc.ps`

[9] P. Jacquet, P. Muhlethaler, A. Qayyum, A. Laouiti, L. Viennot, T. Clausen: *Optimized Link State Routing Protocol*, Internet Draft, draft-ietf-manet-olsr-05.txt, *work in progress*, October 2001.

[10] *HIPERCOM OLSR implementation*, `http://menetou.inria.fr/olsr/`

[11] V. Park, S. Corson: *Temporally-Ordered Routing Algorithm (TORA)*, Internet Draft, draft-ietf-manet-tora-spec-04.txt, *work in progress*, July 2001.

[12] *UMD TORA/IMEP implementation*, `http://www.cshcn.umd.edu/tora.shtml`

[13] *Linux Driver for ORiNOCO v6.06 webpage*, `http://www.wavelan.com/`

[14] Y-C. Hu and D. Johnson: *Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks*, In Proc. of The Sixth Annual International Conference on Mobile Computing and Networking (MobiCom 2000), U.S. August 2000.

[15] A. Kamerman, *Coexistence between Bluetooth and IEEE 802.11 CCK Solutions to Avoid Mutual Interference*, Lucent Technologies Bell Laboratories,

Jan. 1999, also available as IEEE 802.11-00/162, July 2000.

[16] C. Tschudin, H. Lundgren and H. Gulbrandsen, *Active Routing for Ad-hoc Networks*. IEEE Communications Magazine, pages 122-127, April 2000.

[17] *APE testbed project page*, `http://apetestbed.sourceforge.net`

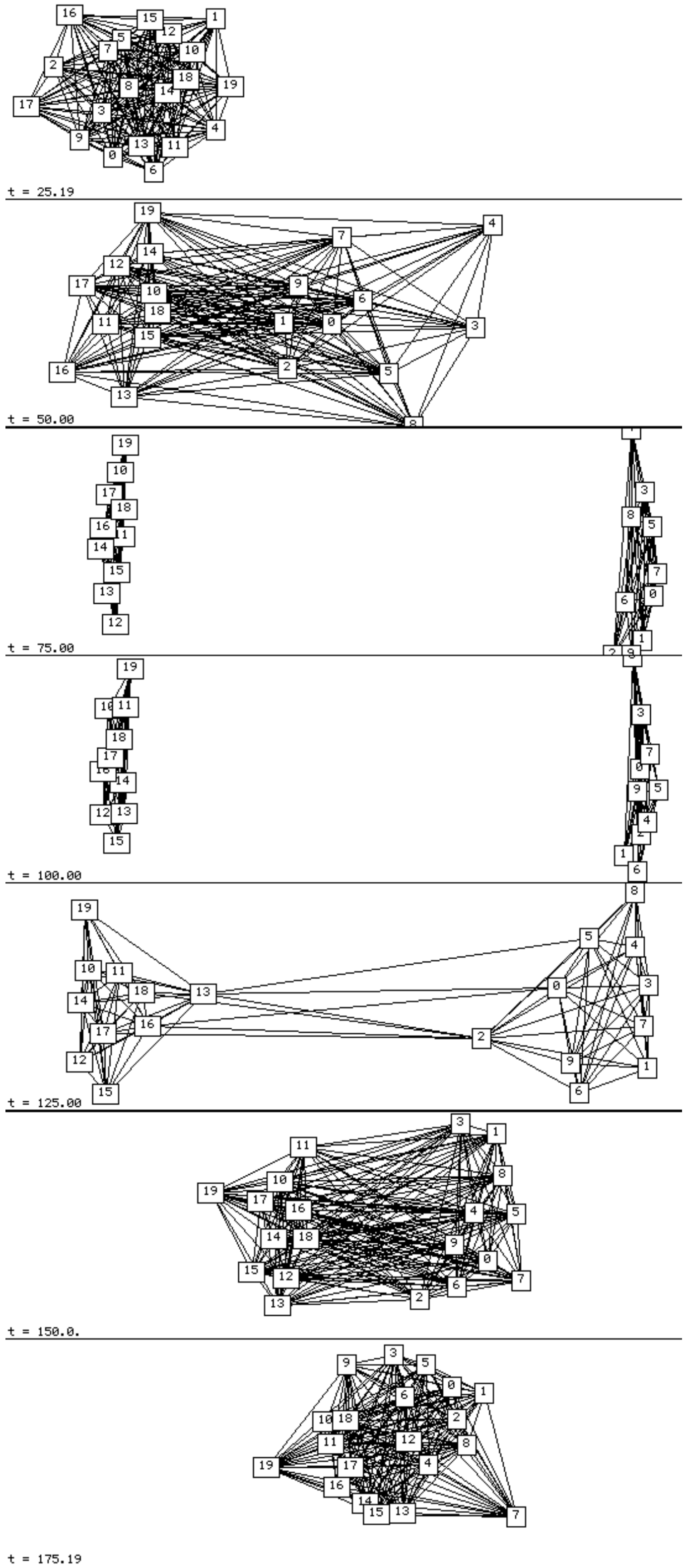


Figure 12: APE-view showing snapshots of virtual positioning from a replay of the "Lost'n'Found" scenario with 20 nodes (see 4.2.1).