A Large-scale Testbed for Reproducible Ad hoc Protocol Evaluations

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

***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 mobility 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.**

***Index Terms*— Ad hoc networking, routing protocols, protocol evaluation, routing testbed, implementation benchmarking.**

I. Introduction

AD HOC networks consist of a number of wireless mobile nodes that dynamically form a network without any pre-existing communication infrastructure. As the nodes may move arbitrarily and unpredictably, the network's topology will be subject to constant changes. Special routing protocols are required in order to coordinate the forwarding of data packets sent by neighboring nodes. 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] but only a few of them through (small-scale) tests [4], [5], [6], [7]. We believe that after so many years of study it is time to expose these routing protocols to real environments. Our Ad hoc Protocol Evaluation testbed (APE) specifically aims at experiments at larger scale and enables a direct comparison of routing protocols by ways of the controlled reproducibility of testruns.

Previous testbed projects report on experiments with 10 nodes or less. In [5], [6], [7] they examine performance using only stationary settings (mobility is emulated in [6] by removing the wireless adapter). In [5], [6] they use off-the-shelf hardware while specific hardware is used in [7]. In [4] they use a mix of both stationary and mobile nodes and point out the problems of real-world, outdoor radio propagation. None of these testbeds report on the possibility to run different protocols or reproducing mobile testruns.

Scaling to larger node numbers is achieved in the APE testbed by making the installation as easy as possible, even for settings with 50 mobile nodes and persons; per-node choreography instructions and internal scripts enable to specify arbitrary mobility patterns and traffic conditions. With the APE's modular architecture we can alter kernel versions, routing protocols, and traffic generators.

Our approach with a strict choreography, combined with a special "virtual Mobility metric" which is derived from measured signal quality, allows us to create verifiably reproducible experiment settings. Our long term goal is assess statements like "under some network condition characterized by Virtual Mobility v, protocol A behaves better than protocol B". However, today such a comparison is beyond the scope of this paper due to the quite limited availability of working implementations of ad hoc routing protocols. Nevertheless, we report in this paper on our experiences and insights from first experiments with up to 37 nodes.

The rest of the paper is outlined as follows. The architecture and implementation of the testbed is presented in Section II. Section III describes the computation of the virtual mobility metric. In Section IV the physical environment is described together with three tested scenarios. Section V discusses the analysis of the experiments. Finally, we conclude and outline future work in Section VI.

II. APE Architecture and Implementation

The APE testbed comprises tools for choreography configuration, network interface data collection, uploading of all measurement results to a central machine as well as post-run analysis and chart generation. The testbed is based on Linux; 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 [8], [9], OLSR [10], [11] and TORA [12], [13].

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 startup of traffic generators and the time synchronization application (TSB). A modification to Lucent Technologies' ORiNOCO driver [14] enables to log signal qualities from *all* packets that a node can detect.

A. Choreography and Data Traffic Generation

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.

Different traffic generators can be inserted independently and would then simply be launched from within the choreography

H. Lundgren, D. Lundberg, E. Nordström and C. Tschudin are with the IT Department, Uppsala University, Sweden. E-mail: henrikl@docs.uu.se

J. Nielsen is with Ericsson Research/Switchlab, Stockholm, Sweden.

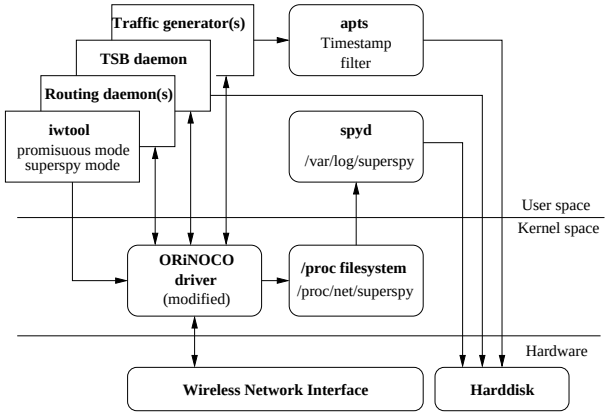

Fig. 1. Overview of software components running on each node in the APE testbed.

```
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
```

Fig. 2. Excerpt from the scenario file for a Double Lost'n'Found scenario. (see Section IV-B for scenario description)

script. Normal Ping, for example, generates useful traffic as we can record packet routes, end-to-end delay, and packet loss. Other types of traffic in the network are beaconing traffic from the time synchronization applications and from the routing daemons.

B. Data Gathering Tools

Data is time stamped and logged both at the Ethernet 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. Each node logs the result of its own IP traffic. After an experiment the log files are directly uploaded by the participants to a mobile "collect node".

C. Data Analysis Tools

Sorting and synchronization scripts are used to merge all log files and to adjust them for clock skews. This results in single log files for the Ethernet as well as for the IP packet level where all entries are globally ordered in time. 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, packet loss and hop count.

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 V-A.2 and the figure at the end of this paper).

D. Testbed Distribution

Due to the large number of nodes and persons involved, the configuration of each mobile node must be very simple. We assume that participants download on an experiment basis a pre-made software package. The package to install is identical for everybody and can be used on a host computer running either the Windows98 or Linux operating systems. No harddisk re-partitioning is required: under Windows it suffices to execute a self-extracting file which makes the computers reboot to Linux, reading and writing to a filesystem image included in the distribution. Under Linux, 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. A set of special "software package fabrication programs" allows us to easily generate packages for different routing protocols, Linux kernels and choreography scripts.

III. Capturing Mobility in Ad hoc Networks

A mobility metric is useful for characterizing the state of the network and its dynamics. Geometric mobility metrics like those described in [3], [15] require that the nodes' physical positions are known at all times. Real-world settings require GPS or triangulation techniques to achieve this in outdoor or indoor environments, respectively. Looking at route changes due to link breakages is the basis of another mobility metric proposed in [15]. More (and less binary) information can be obtained by monitoring the link quality, as we will show below.

A. A "Virtual Mobility" Metric

Background noise, obstacles and even small movement can influence the signal quality. Instead of using the real distance between two nodes, we propose to compute a "virtual distance" from the perceived signal quality. This metric captures the real world dynamics as perceived by the nodes. In this section we define this new mobility metric; in Section V we show that this metric gives a "fingerprint" of the link quality changes that characterizes a testrun.

1) Path Loss Model: The definition of our mobility metric relies on "virtual" distances which we base on the *measured* signal quality. The path loss model mentioned in [16] is used to relate signal quality and distance: for the far field case (indoor) it proposes a path loss coefficient of 3.3:

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

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

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

where Q is the signal quality $(0 \ldots 75)$ 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.

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

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

where P_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

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

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

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

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

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

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

where N is the number of nodes in the network.

This basic definition yields the *mean* network vM-value at every time interval. It represents how an average node moves during a test. The upper and lower quantiles of this mean value reflect the movement heterogeneity and can reveal different movement patterns within the network. Depending on the focus of interest, the basic vM definition can be altered in order to take multiple hops into account, to use active links only, or to weight links differently etc.

IV. Experimental Set-up

A. 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 lecture halls. The white paths through the buildings are the corridors and bridges connecting 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 buildings stretches over totally 174 meters. The walls within the buildings are very thick and radio signals are effectively damped when loosing line-of-sight. The nodes move at normal walking speed, in this case meaning slightly above 1 m/s.

B. Scenario Descriptions

Three scenarios will be described briefly below for which we will show measurement results in Section V. Figure 4 in conjunction with Figure 3 give an overview of the exact movements and times within these three scenarios.

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. The scenario is choreographed in the following way. Assume a group of nodes that resides at location D. After a while the group splits into two clusters and one of the clusters moves away towards location A. At some point when the moving cluster is between location D and A the two clusters lose radio contact with each other. The remote cluster stays at location A for a while before returning to D. The data traffic in this scenario consists of all nodes sending broadcast Ping.

2) Double Lost'n'Found: This scenario focuses on how different movement patterns are visualized by the vM metric. The differences from the Lost'n'Found scenario are that here there are three clusters residing at E and two of them are "lost" simultaneously when moving towards A and H, respectively, but "found" at different times when moving back towards E. The traffic in this scenario consists of unicast Pings from each cluster to the other two clusters.

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 a while 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. 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). After some time the remote clusters start moving back towards their respective place of origin and the other cluster. The traffic in this scenario consists of unicast Pings from each cluster to the other three clusters.

V. First Results

We will present results from repeated experiments using the AODV Mad-hoc implementation [9] and Inria's OLSR implementation [11] with the number of participating nodes ranging from 9 to 37 nodes. The focus in section V-A will be on using the *vM* metric as a tool for capturing movements within the network and comparing repeated testruns. In section V-B we focus on the ping success ratio and IP packet hop count calculations. The size of some of the log files used to extract these results reached 70 MB and contained time stamped entries for more than 3 million packets.

A. vM Analysis

Figures 5, 6 and 7 in this section show the network virtual mobility (m/s) as a function of time (s). The solid line represents the mean value. The upper quantile and lower quantile are represented by dotted and dashed lines, respectively. These are here called vM-high and vM-low.

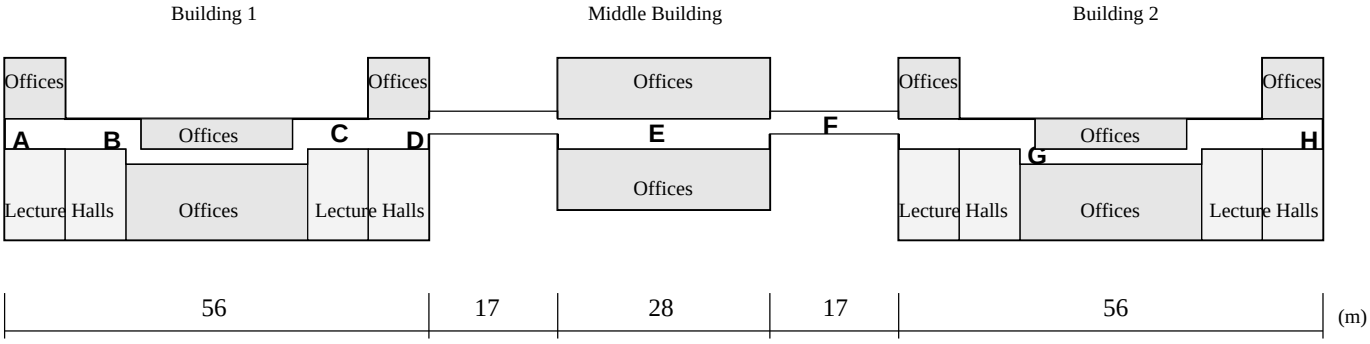

Fig. 3. The physical test environment.

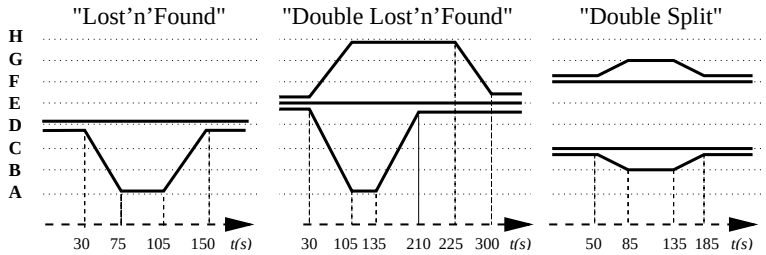

Fig. 4. Time-space diagrams for the three experimental scenarios discussed in Section IV-B.

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 the radio signal quality between nodes standing closely together. Inherent radio propagation properties like reflection as well as very small movements by participants could cause signal quality changes.

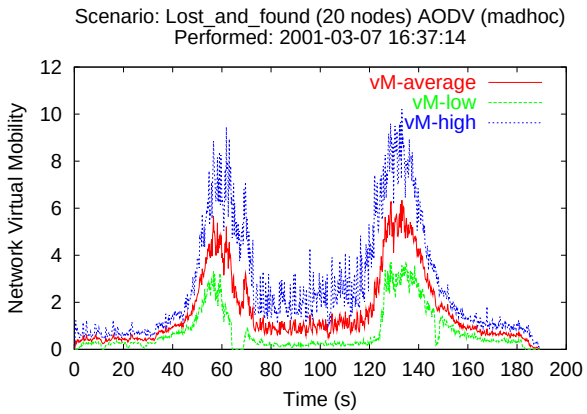

Fig. 5. The virtual mobility for scenario *lost'n'found* with 20 participating nodes.

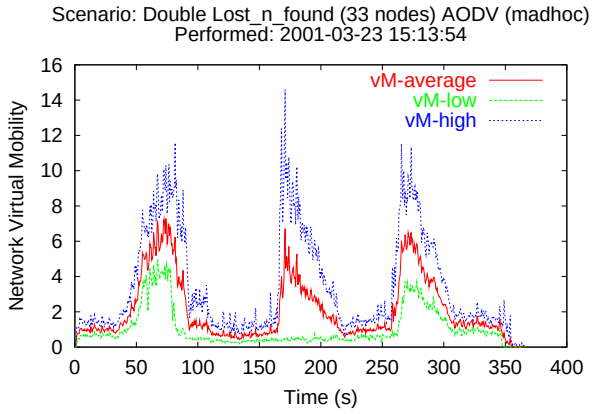

Fig. 6. The virtual mobility for scenario *double lost'n'found* with 33 participating nodes.

1) Characterizing a Testrun with vM: Figure 5 shows vM for a Lost'n'Found scenario with 20 nodes. The two peaks at time 60 and 125 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-value curve is back to approximately the same level as when the two groups were standing right next to each other. This vM graph together with the known choreography represents the character of this testrun.

2) Deriving Topological Positions from Signal Quality Information: It is possible to recreate the topological correct configuration of an experiment using the collected signal quality data. This allows us to visualize logical connectivity for each node and to 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.

3) Discerning Different Movement Patterns Among Nodes: Figure 6 shows the vM values for a Double-Lost'n'Found scenario run with 33 nodes. The three peaks represents the split (at time 80) and the two re-connections (at time 170 and 260). The split event gets the highest average 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 each of the two re-connections.

The vM-low curve in Figure 6 has no peak during the first re-connection around time 170, indicating low mobility among

some of the nodes. This is perfectly reasonable because the third group, still standing still at the remote location 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 values to test whether different groups have different mobility patterns during a testrun.

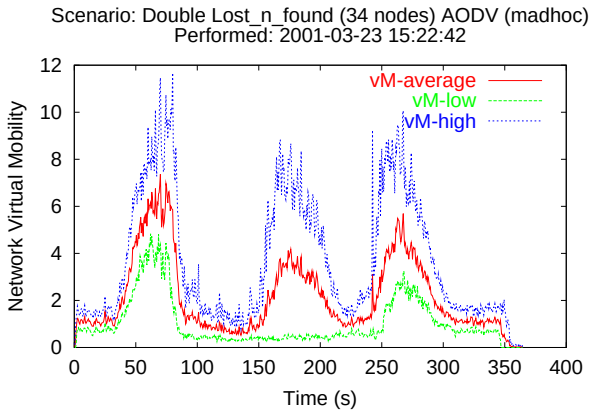

Fig. 7. The virtual mobility for scenario *double lost'n'found* with 34 paticipating nodes.

4) Reproducing Testruns: We repeated testruns several times in order to examine how well we can *reproduce* results. Figure 7 show the second run of the Double Lost'n'Found scenario (compare with 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. 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 ensures that the participants move in the same fashion and that the vM can reveal whether the signal quality fluctuations in two testruns are similar enough to make results from further performance analysis comparable.

B. Packet Loss and Hop Count Analysis

Figure 8 shows the Ping success ratio for the *Double Lost'n'Found* scenario with 34 nodes, using the Mad-hoc AODV [9] implementation that is based on AODV draft version 5. 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. During time 90 and 160 no Ping packets get through between the groups. This is the time when the three groups have no radio contact. During the re-connection phase of the first groups the ping success ratio increases and stabilizes around 0.5 until time 250 when the second group enters into radio range and all nodes gradually regain full connectivity. Figure 9 shows the connectivity over time and one can clearly see the different phases of the Double Lost'n'Found scenario.

One would expect that – as the two clusters at the edges move apart from each other – they would re-route their traffic via the

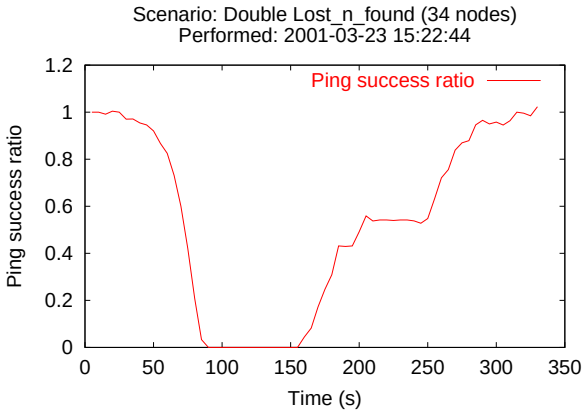

Fig. 8. Ping success ratio for the *Double Lost'n'Found* testrun with 34 nodes.

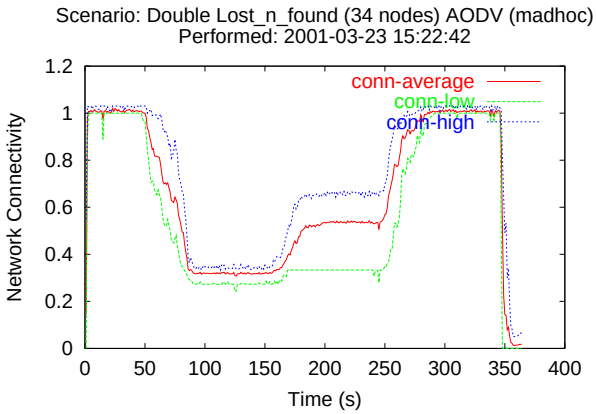

Fig. 9. Connectivity for the *Double Lost'n'Found* testrun with 34 nodes.

middle cluster. However, analysis of the hop count reveals that this is not the case for the Mad-hoc implementation (see Table I). In a Double Split scenario where the clusters retain contact with their adjacent groups, we even observed that Mad-hoc did not find ping paths longer than 2 hops at all.

	1 hop	*2 hops*	*3 hops*
Request paths	1127	1	–
Reply paths	1127	1	–
Complete pings	n/a	1126	2

TABLE I

NUMBER OF SUCCESSFUL PINGS CLASSIFIED BY THE NUMBER OF HOPS, FOR MAD-HOC IN *Double Lost'n'Found* WITH 34 NODES

Hop Count Analysis – Mad-hoc-AODV vs. OLSR-Inria: We repeated the Double Lost'n'Found and Double Split testruns, with 9 and 8 nodes, respectively, with the Mad-hoc-AODV software and the OLSR-Inria implementation [11] which is based on OLSR draft version 3. In the tables II to V we compare the number of hops that pings were taking during the testruns.

These figures confirm our intuitive finding: The OLSR-Inria implementation runs smoothly and has no problem to set up multi-hop paths. For example, it managed to set up 4-hop paths (7 hops when counting forth and back path) in the Double Split

	1 hop	*2 hops*	*3 hops*
Request paths	2940	1	–
Reply paths	2940	1	–
Complete pings	n/a	2939	2

TABLE II

NUMBER OF SUCCESSFUL PINGS CLASSIFIED BY THE NUMBER OF HOPS, FOR MAD-HOC IN *Double Lost'n'Found* WITH 9 NODES

	1 hop	*2 hops*	*3 hops*	*4 hops*
Request paths	3009	40	–	–
Reply paths	2974	71	4	–
Complete pings	n/a	2963	53	33

TABLE III

NUMBER OF SUCCESSFUL PINGS CLASSIFIED BY THE NUMBER OF HOPS, FOR OLSR-INRIA IN *Double Lost'n'Found* WITH 9 NODES

scenario. Mad-hoc-AODV, on the other hand, seems to fail in detecting link breakages and setting up new multi-hop routes in mobile scenarios.

	1 hop	*2 hops*	*3 hops*
Request paths	2962	2	–
Reply paths	2959	5	–
Complete pings	n/a	2957	7

TABLE IV

NUMBER OF SUCCESSFUL PINGS CLASSIFIED BY THE NUMBER OF HOPS, FOR MAD-HOC IN *Double Split* WITH 8 NODES

	1 hop	*2 hops*	*3 hops*	*4 hops*
Request paths	2892	371	100	15
Reply paths	2894	380	98	6
Complete pings	n/a	2870	46	336

	5 hops	*6 hops*	*7 hops*	*8 hops*
Complete pings	26	86	14	–

TABLE V

NUMBER OF SUCCESSFUL PINGS CLASSIFIED BY THE NUMBER OF HOPS, FOR OLSR-INRIA IN *Double Split* WITH 8 NODES

The Figures 10 and 11 show the link changes per second during the Double Split scenario for Mad-hoc and OLSR-Inria, respectively. The graphs indicate that the implementations have been exposed to similar degrees of fluctuating links.

Although the OLSR-Inria implementation seems stable and capable of setting up multi-hop routes, we have to conclude for the moment that a direct comparison of the protocol performances is not yet possible.

VI. SUMMARY AND FUTURE WORK

After many years of simulations and in order to become accepted Internet protocols it is important to test ad hoc routing

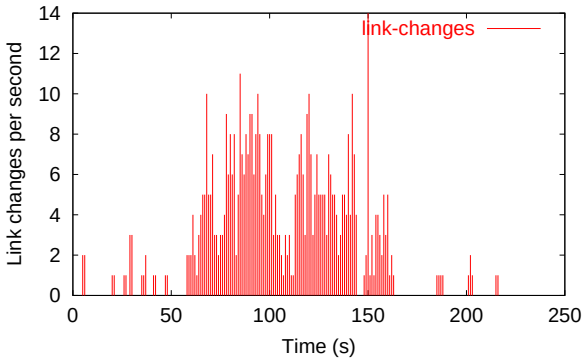

Fig. 10. Link changes per second for Mad-hoc in the *Double Split* testrun with 8 nodes.

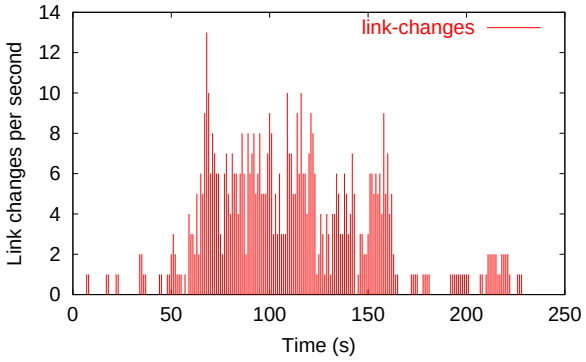

Fig. 11. Link changes per second for OLSR-Inria in the *Double Split* testrun with 8 nodes.

protocols with a large number of nodes in real world settings. In this paper we report on the Ad hoc Protocol Evaluation testbed (APE): It is based on choreographed testruns which enables to accurately repeat experiments. APE has a modular architecture allowing us to plug in different routing protocols and traffic generators. So far APE was used for testruns with 3 different protocols and up to 37 nodes. Our experiences are that quality and maturity of protocol implementations differs heavily and although we have AODV, TORA and OLSR working inside APE, we need more (and more stable) implementations to make full use of APE. However, our experiments show that APE can also be used for examining implementation behavior, as in the side-by-side comparison between Mad-hoc AODV and OLSR-Inria. We plan to make the APE testbed distribution available to other research institutions [17].

As a part of our APE efforts we have developed a new mobility metric called "virtual mobility" (vM); It is based on changes in the measured signal quality and characterizes a network's mobility condition. This metric provides the basis for assessing the reproducibility of testruns in comparison experiments. Our choreography approach turned out to be successful under this metric; Virtual distances can also be used to reconstruct topological network configurations without additional positioning information. We expect the gathered signal strength data to be further useful for validating existing simulation models and for running trace-driven simulations.

ACKNOWLEDGEMENTS

We would like to thank Ericsson for donating the wireless ORiNOCO PCMCIA cards and Per Gunningberg for setting up this project's framework. Many thanks to our colleagues and the students who participated in the experiments.

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.* Proceedings of ACM/IEEE MobiCom'98, 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 ACM/IEEE MobiCom'99, August 1999.

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

[5] C-K. Toh, Richard Chen, Minar Delwar, and Donald Allen. *Experimenting with an Ad Hoc Wireless Network on Campus: Insights and Experiences* ACM SIGMETRICS Performance Evaluation Review, Vol. 28, No. 3, pages 21-29, 2001.

[6] C-K. Toh, Minar Delwar, and Donald Allen. *Evaluating the Communication Performance of an Ad Hoc Wireless Network* To Appear in IEEE Journal on Selected Areas in Communications (JSAC Wireless Series), 2001.

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

[8] 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.

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

[10] 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.

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

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

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

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

[15] Y-C. HU AND D. JOHNSON: *Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks*, Proceedings of ACM/IEEE MobiCom'2000, August 2000.

[16] 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.

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

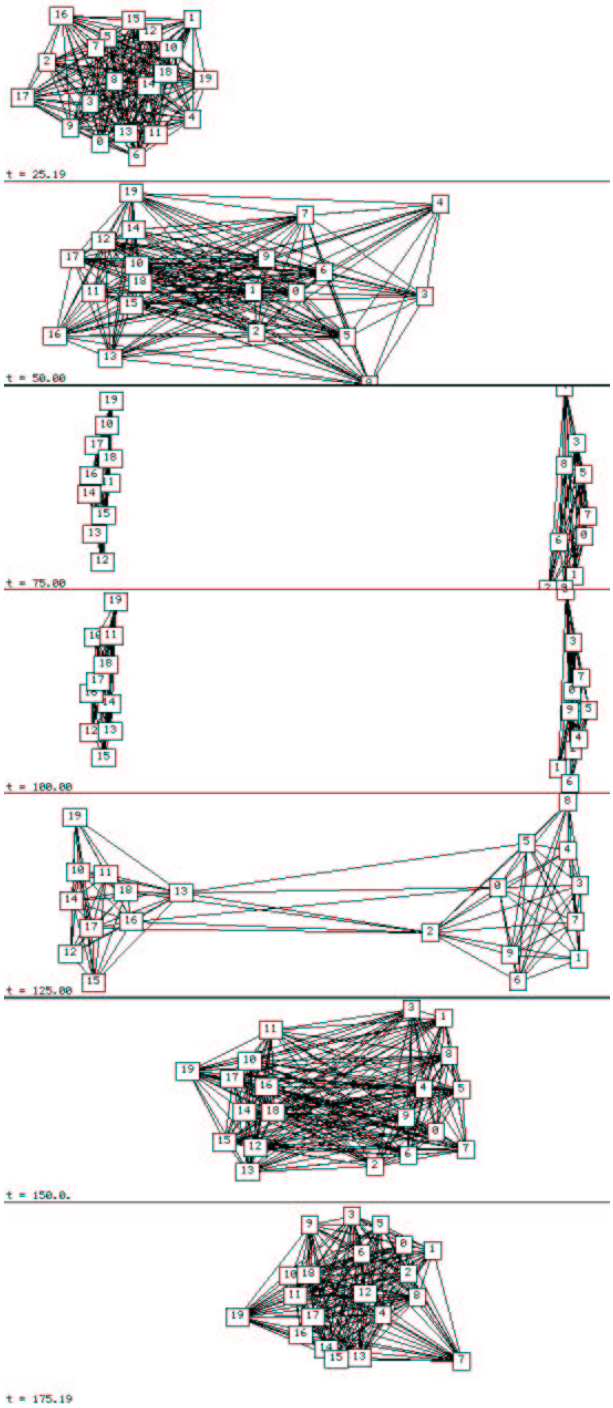

Fig. 12. Snapshots of virtual positioning from a replay of the "Lost'n'Found" scenario with 20 nodes (see Section V-A.2).