Master of Science in Engineering
Examensarbete ITP
28th June 2002

APE - a Large Scale Ad Hoc Network Testbed for Reproducible Performance Tests

Erik Nordström

Information Technology
Department of Computer Systems
Uppsala University
Box 325
SE-751 05, Uppsala
Sweden

Supervisor: Christian Tschudin
Examiner: Christian Tschudin

Abstract

Wireless networks with autonomous mobile nodes, called ad hoc networks, have become increasingly interesting with the growth of wireless communication technologies. Several proposals for protocols to handle routing in such environments are under consideration by the Mobile Ad Hoc Networks (MANET) Working Group of the IETF. Simulations have been the driving force in development and evaluation of ad hoc routing protocols, but real world testing is currently lacking.

The goal of this master's thesis has been to develop a testbed such that large scale, real world tests of 50 or more nodes can be conducted. The testbed makes use of choreographed node movements through instructions to the user in order to reproduce testruns as accurately as possible. A new mobility metric called *virtual mobility* based on signal strength is introduced and enables post testrun analysis and characterization of network mobility. Initial results based among others on testruns with 40 nodes and analysis of this metric show that it has "fingerprint" like properties that makes it useful when verifying similarities between repeated testruns. The APE software has been put in the public domain and can be downloaded at `http://apetestbed.sourceforge.net`.

Contents

1 Introduction **3**
1.1 Problem Description . . . 3
1.2 Goals . . . 3
1.3 The APE Testbed . . . 4
1.4 Related Work . . . 4
1.4.1 Simulations . . . 4
1.4.2 Real World Testbeds . . . 5
1.4.3 Other Work . . . 5
1.5 Overview . . . 6

2 Wireless Ad hoc Networks **7**
2.1 Usage . . . 7
2.2 Challenges of Multi-hop Forwarding . . . 7
2.3 Proposed Ad hoc Routing Protocols . . . 8
2.4 AODV - Ad hoc On-demand Distance Vector . . . 9
2.4.1 Route Discovery . . . 10
2.4.2 Route Maintenance . . . 10

3 Performance Evaluation of Ad Hoc Routing Protocols **11**
3.1 Qualitative Metrics . . . 11
3.2 Quantitative Metrics . . . 11
3.3 Networking Context . . . 12
3.4 Simulations vs Real World Tests . . . 12

4 The Ad hoc Protocol Evaluation Testbed (APE) **14**
4.1 Test Goals . . . 14
4.2 Scaling of Tests . . . 14
4.3 Runtime Environment . . . 15
4.3.1 Routing Protocol . . . 15
4.4 Choreography . . . 15
4.4.1 “Monkey-Walk” . . . 16
4.5 Data Gathering . . . 17
4.5.1 Ethernet Level . . . 17
4.5.2 IP Level . . . 18
4.5.3 Time Synchronization . . . 19
4.6 Data Analysis . . . 19
4.6.1 APE Analysis Tools . . . 20
4.6.2 MAC Level Analysis . . . 20
4.6.3 IP Level Analysis . . . 20
4.6.4 Data Visualization . . . 22

5 A “Virtual Mobility” Metric **23**
5.1 Mobility Models . . . 23
5.2 Virtual Distance . . . 24
5.2.1 Path Loss Model . . . 24
5.3 Calculating “Virtual Mobility” (vM) . . . 24
5.4 vM Variations . . . 25
5.4.1 Multi-hop Virtual Mobility . . . 25

6 Performance Measurements **27**
6.1 Test Scenarios 27
6.1.1 Physical Environment 27
6.1.2 Scenario 1: “Global Warming” 28
6.1.3 Scenario 2: “Lost ’n’ Found” 28
6.1.4 Scenario 3: “Double Lost ’n’ Found” 28
6.1.5 Scenario 4: “Double Split” 28
6.2 vM Analysis 29
6.2.1 Lost’n’Found 29
6.2.2 Double Lost ’n’ Found 30
6.2.3 Double Split 31
6.3 Packet Loss Analysis 32

7 Conclusions and Outlook **34**
7.1 Future work 34

1 Introduction

At Uppsala University, selected students have been supplied with personal laptop computers in an attempt to encourage their use of computers in daily work. Some students have also had the opportunity to borrow wireless IEEE 802.11 based WaveLAN cards for wireless network access. This availability of people and hardware has given the department an opening for real world, hands-on research in the wireless network area, by having students participate in large scale tests.

During summer 2000, two students started development of a large scale multi-hop wireless ad hoc network testbed. This initial work lay the foundation for what would later become the APE testbed. APE stands for **A**d-hoc **P**rotocol **E**valuation and is a test environment consisting of a customized easy-to-install Linux distribution and a set of tools to perform post testrun analysis.

1.1 Problem Description

The performance and functionality of ad hoc routing protocols have so far mostly been evaluated through simulations. The picture is not complete without tests in a real world setting with inherently complex and dynamic environments. The APE testbed aims at filling this gap.

Three main issues with creating a real world ad hoc network testbed have been identified:

- Administration of large scale tests.
- Achieving reproducible results.
- Developing real world test metrics.

Large scale real world tests naturally have administrative issues. They include constructing robust easy to deploy software necessary to conduct the test, coordination and instructing of people participating.

Reproducible results are always of major concern in any serious research and methods have to be developed to make sure such are possible.

Test metrics are necessary to evaluate ad hoc routing protocols. It is preferred that metrics used in simulations are transferred to real world tests if possible, for comparisons and validation of simulations.

1.2 Goals

The work presented in this thesis aims to create a reliable platform for maintaining reproducible results and characterizations of real world ad hoc network environments. More specifically, the goals of this master's thesis has been to, in a hands-on way:

- implement a real world testbed for ad hoc routing protocols.
- conduct initial tests.
- provide results and analysis.
- show that reproducible results are achievable.

This report does not contain any discussions or test results of different routing protocols, but an analysis of in which manner these measurements can or should be executed in future research using the APE testbed.

1.3 The APE Testbed

The APE testbed is a self contained, stand-alone Linux based test environment. With APE, it is possible to deploy and administer large scale tests of wireless multi-hop ad hoc network protocols, utilizing students (which are suitably called monkeys) with no prior knowledge about tests of this kind. Furthermore, tests can be reproduced with great accuracy, so that results can be verified and different routing protocols be compared side by side. To achieve this, APE introduces the *Virtual Mobility* (vM) metric - a real world mobility metric based on the signal-to-noise ratio (SNR) of data transmissions. The SNR can be used to estimate distances between nodes in a connectivity sense. Virtual mobility has proved to be a useful tool for the analysis of movement during testruns and provide *"fingerprint"*-like properties which can, not only show similarities between reproduced testruns, but also provide detailed insights into the *heterogeneity* of movement within the network. Results with the vM metric will be presented using an implementation of the AODV routing protocol, presented in a later section.

The APE testbed also make use of scripted scenarios to coordinate movements during testruns. Participants carrying laptops are instructed via messages on the displays. The scripting system ensures that testruns can be reproduced in a movement and data traffic sense. It is also possible to perform completely uncoordinated random walk tests if desirable.

A graphical animation tool, called APE-view, has been developed to visualize logical node placements and movements that occurred during testruns. It is driven by logged signal strengths of packet transmissions and gives a surprisingly accurate view of node placement and movement during the course of a scenario.

For analyzing the gathered data, the APE testbed includes a set of analysis tools with a graphical frontend (GUI) for performing calculations on metrics such as vM, packet loss and path optimality.

In the future, the APE testbed will be available for public use and downloadable on the Internet at `http://apetestbed.sourceforge.net`. The software is distributed under the *GNU General Public License* (GPL)[3], which ensures the right for users to modify the code in any way desirable. The idea is for APE to evolve by developer contributions from other people interested in real world tests of wireless ad hoc networking protocols.

1.4 Related Work

This section gives an overview of the research efforts related to ad hoc networks and protocols made prior to the creation of the APE testbed.

1.4.1 Simulations

Wireless ad hoc routing protocols have been subject to intensive evaluations using simulators, like for example *ns-2*[1]. Johansson et al [22] (based on [23]) and Broch et al [18] - part of the MONARCH project [19], represents two such studies. Both deal with evaluations of ad hoc protocols such as AODV, DSDV, DSR and TORA, to name a few. In a simulated environment, Broch et al (CMU) chose to evaluate the performance of protocols using three metrics, namely *packet loss*, *routing overhead* (total number of routing packets) and *route optimality* (difference between number of hops taken and shortest path). They also defined a mobility model called *"random waypoint"* to model the movement patterns of the participating nodes and to examine how mobility affects the routing protocols ability to maintain a usable multi-hop network.

Johansson et al conduct similar studies, evaluating DSDV, AODV and DSR, but choosing a more complicated mobility model based on relative movements between nodes.

[1] Berkeley network simulator

Both papers report similar results. The simulations show that mobility and network scale have great impact on the performance of the routing protocols. DSDV show, as predicted (being a more traditional routing protocol), that high movement rates and large mobility hurts its performance while its functionality in less stressed environments is good. AODV and DSR was found to be the most competent protocols for use in an ad hoc network environment, with very similar performance, although representing different design choices. TORA was a worse performer, but held up quite well in most situations.

1.4.2 Real World Testbeds

The MONARCH project [19] has created a testbed described by Maltz et al in [29] and [30]. Maltz et al, also co-authors of the simulation study [18], present an outdoor ad hoc network testbed featuring two stationary nodes, five car-mounted nodes that drive around on a predefined testbed site, and one car-mounted roving node, entering and leaving the site. Routing in the network was handled by a testbed specific implementation of the *Dynamic Source Routing* protocol (DSR)[21]. Global Positioning System (GPS) was used to track nodes during their course. Positioning information was stored and could later be used to replay the scenario in a simulation.

The purpose of the testbed was to enable basic research on the behavior of real implementations of ad hoc networking protocols. The testbed aimed to push the protocols to the point where they broke.

The MONARCH testbed is a complex test environment with high costs and little or no possibility for others to reproduce tests or verify results. GPS provides accurate positioning, but adds to the cost, complexity and difficulty in deployment and is not suitable for indoor use. This eliminates the possibility to do comparisons between indoor and outdoor environments.

Toh et al describe in [24] tests in an outdoor environment with a four node setting. They evaluate performance of protocols using different network parameters like beaconing interval, route length and packet size.

Other approaches on testbeds exists, but they are often small and highly specialized for a specific task, such as the presented in [25]. This paper describe a small seven node testbed for a performance analysis of the ODMPR unicast routing protocol. ODMPR are interchangeably tested in similar simple scenario setups, where movement of at the most one node per test occur. Performance comparisons are made to a setup using static routing.

1.4.3 Other Work

Simulations and real tests are combined in [20] to validate results from simulation. Trace logs from simulations were used to drive emulations of wireless networks in a testbed consisting of a wired network. Time-sequence number plots of TCP throughput was produced to be able to assess the degree to which the results from the simulator match those from the real world.

The ARRCANE project [26, 27] at the department of computer systems (DoCS) at Uppsala University, conduct research in the area of active routing protocols. A product of this work is an implementation of two active routing protocols for ad hoc networks. The ARRCANE project will take advantage of work done in the APE project by integrating its active routing protocols into the testbed.

Also at DoCS, the AODV ad hoc network routing protocol is implemented for operation in a Linux/UNIX environment. It will be integrated into the APE testbed, replacing the currently used user space implementation of AODV, [31].

1.5 Overview

The rest of this thesis is outlined as follows. Section 2 gives an introduction to wireless ad hoc networks along with an overview of the AODV routing protocol. Issues related to performance evaluations of ad hoc routing protocols are discussed in section 3. A detailed look at the APE testbed and its components can be found in section 4. In section 5 a new mobility metric based on signal strength called "virtual mobility" (vM) is introduced. It can be used to characterize the movements during real world tests and assess the similarities between repeated testruns. Preliminary performance measurements of an AODV implementation are presented in section 6, along with a presentation of APE scenarios and vM analysis. Section 7 concludes this master's thesis with an outlook.

2 Wireless Ad hoc Networks

Ad hoc networks are dynamic networks without any permanent or pre-established infrastructure. Wireless communication devices are commonly used in these environments with technologies such as IEEE 802.11 (WaveLAN) and BlueTooth being the most recognized ones. This thesis will only deal with IEEE 802.11.

In ad hoc networks, nodes may move arbitrarily and adjust their transmission and reception parameters at any time and can establish both unidirectional and bidirectional links with other nodes.

An ad hoc network is typically formed by terminals that meet and wants to easily exchange information. An example could be a conference, where the participants wants to exchange documents between their computers, without the necessity of a local fixed network infrastructure.

Ad hoc networks do not rely on any centralized administration and often no distinction is made between nodes, since all of them can act as routers. Therefore, ad hoc networks are robust - nodes can join or leave at any time and the network does not rely on a few dedicated routers. A system may operate in isolation or in connection with other networks, typically an inter-network (such as the Internet) through a gateway.

2.1 Usage

Ad hoc networks are a new phenomenon due to the recent broad availability of cheap wireless communication hardware. Ad hoc networks with sophisticated routing are practically non-existent and there is no standardized or established protocol for these purposes. However, future applications commonly mentioned may include deployment of networks in disaster areas, where other infrastructure have been destroyed, conference scenario where people easily wants to exchange information and various military applications.

There may also exist other reasons for using ad hoc networks. They could be a natural and cheap extension to a fixed network, where a node in the ad hoc network is attached to the fixed network, possible acting as an Internet gateway for other ad hoc network nodes. Another possible use could be to limit the range and accessibility of traditional base-station based WaveLAN's to minimize eavesdropping of the traffic in the network. By removing the base-stations and using routing capable nodes with limited short radio transmission ranges, it is possible to limit transmission to the geographical locations of nodes. One would have to get very close to the network's outer limit to intercept any transmissions.

2.2 Challenges of Multi-hop Forwarding

There are several reasons why routing traffic between intermediate nodes is desirable. A typical ad hoc network node is mobile - a laptop or other smaller appliance - and therefore holds limited battery capacity. Sending radio signals over large distances consumes much power. The *inverse square law* [28] defines the power of a received signal to be proportional to $1/d^2$, where d is the distance to the source. Increasing the distance to twice the original distance has the effect that the power of the signal at the source must be increased four times for the received signal to retain its original power. This makes it desirable for two communicating nodes to route packets over intermediate nodes. The power needed to transmit packets are lowered and batteries are saved. At the same time it can be argued that acting as a router also increases power consumption, although to what amount is unclear. Power consumption in wireless ad hoc networks is a research topic of its own and will not be further discussed in this thesis.

Naturally, routing packets also increase the theoretical range of communication above that of the wireless interface. This alone is a good reason for using intermediate nodes to forward traffic. But routing can also be useful when trying to reach nodes that are occluded or in radio shadow.

Because there are no stable network topologies or pre-defined infrastructures in ad hoc networks, traditional routing methods are not applicable. In ad hoc networks, often all nodes act as routers, forwarding each others packets. Traditional routing algorithms in wired networks work too slowly or inefficiently and routing information can, due to the dynamic environment, be rendered obsolete in a matter of seconds. The routing protocol can not rely on the availability of dedicated routers.

Asymmetric links are common in ad hoc networks. Nodes can not make any assumptions about the quality of a connection in the reverse direction just because it is good in the forward direction. There can also be a lot of redundant routes, something traditional routing protocols are not constructed to handle.

To conclude the problematics of routing in wireless ad hoc networks, it can be stated that there is a necessity for dedicated ad hoc network routing protocols to handle the unique environments and their inherent properties.

2.3 Proposed Ad hoc Routing Protocols

At the time of writing of this thesis there exists no standard ad hoc network routing protocol, but several proposals are currently under consideration at the IETF MANET working group[1]. The general purpose of this working group, as stated in rfc2501[2]:

> *The intent of the newly formed IETF manet working group is to develop a peer-to-peer mobile routing capability in a purely mobile, wireless domain. This capability will exist beyond the fixed network (as supported by traditional IP networking) and beyond the one-hop fringe of the fixed network.*

Ad hoc routing protocols can be divided into two categories - namely *reactive* and *proactive* protocols. Proactive protocols share many similarities with traditional routing algorithms in the sense that they continuously evaluate routes within the network. When a data packet is sent onto the network a route is already available. Reactive protocols works on demand, such that a global route search routine is initiated whenever packets need to be forwarded and a route to the desired destination is not known by the source. Listed below are the IETF MANET ad hoc routing protocols along with a brief description of each of them.

- **AODV** Ad hoc On Demand Distance Vector.[4]
  Described separately in section 2.4.
- **TORA** Temporally Ordered Routing Algorithm.[5]
  Distributed execution with loop-free, multipath routing. Both reactive and proactive route establishment and maintenance.
- **DSR** Dynamic Source Routing.[6]
  Data packets contain complete ordered lists of nodes to travel, providing loop-free operation. Route discovery is reactive, complete on-demand with low overhead.
- **OLSR** Optimized Link State Routing Protocol.[7]
  Table driven, proactive protocol which exchanges topology information at regular intervals through optimized flooding. Well suited for large and dense networks.
- **TBRPF** Topology Broadcast based on Reverse-Path Forwarding.[8]
  Proactive, link-state routing protocol. It maintains optimal paths to all destinations at all times, unlike on-demand protocols. No periodic broadcasting of topology information is needed.
- **LANMAR** Landmark Routing Protocol for Large Scale Ad Hoc Networks.[9]
  Uses the notion of “landmarks” to keep track of logical subnets. It relies on the

movement of nodes in logical groups. Each group has an elected landmark, to which a route is propagated throughout the network using a distance vector algorithm. Runs on top of a proactive routing protocol, like Fisheye. LANMAR is best suited for large scale mobile ad hoc wireless networks.

- A Simple Protocol for Multicast and Broadcast in Mobile Ad Hoc Networks.[10]
  Utilizes the route discovery mechanism of DSR. Make use of controlled flooding to distribute data in the network. Provides less overhead in small networks with the same level of functionality than a stateful protocol which require explicit establishment of multicast state for data distribution.

- **FSR** Fisheye State Routing Protocol.[11]
  Fisheye State Routing is a proactive, table driven routing protocol which is adapted to the wireless ad hoc environment. It introduces the notion of multi-level "scope" to reduce routing update overhead in large networks. The scope relative to a destination decide the frequency of broadcasting link state updates. Because of this, Fisheye is best suited for large scale ad hoc networks.

- **IERP** The Interzone Routing Protocol.[12]
  The reactive component of the Zone Routing Protocol (ZRP). Use the notation of “routing zones” provided by the IARP protocol to adapt reactive routing algorithms to make use of surrounding topology. This way it can suppress route queries for local destinations.

- **IARP** The Intrazone Routing Protocol.[13]
  Limited scope proactive routing component of ZRP used to improve the performance of existing globally reactive routing protocols. Uses the notation of “routing zones” and “bordercasting” for global route searches.

- **BRP** The Bordercast Resolution Protocol.[14]
  Provides the bordercasting packet delivery service used to support network querying applications. Bordercasting is proposed as a more efficient and tunable alternative to broadcasting of route request messages for reactive (on-demand) routing protocols. BRP is not a fully featured routing protocol and work with the IARP and IERP protocols within the ZRP context.

- Flow State in the Dynamic Source Routing Protocol.[15]
  Extension to the DSR protocol, known as "flow state", that allows the routing of most packets without an explicit source route header in the packet, further reducing the overhead of the DSR protocol.

Other ad hoc routing protocols that can be mentioned are the Cluster Based Routing Protocol (CBRP), the Core Extraction Distributed Ad Hoc Routing (CEDAR), Associativity Base Routing (ABR - a former MANET protocol)[16] and the Ad hoc Multicast Routing Protocol (AMRoute).

2.4 AODV - Ad hoc On-demand Distance Vector

This thesis focus on evaluating an implementation of the Ad Hoc on-Demand Distance Vector (AODV) routing protocol[4]. It is a dynamic, self-starting multi-hop routing algorithm which can be used by nodes that wish to form an ad hoc network. Because it is a routing protocol it deals with route table management, enabling intermediate nodes to forward packets for other nodes. It is a *reactive* protocol which, as opposed to *proactive* protocols, find and maintain routes only when needed. Despite the reactive properties, AODV allows mobile nodes to quickly obtain routes to new destinations due to its fast response to link breaks and network topology changes.

Features include loop-free operation and quick convergence at network topology changes by avoiding the Bellman-Ford “count-to-infinity” problem[17], commonly associated with traditional distance vector routing.

A distinguishing feature of AODV is the use of destination sequence numbers for each route entry, which ensures the loop-free operation. Destination sequence numbers represent the“freshness” of routes.

2.4.1 Route Discovery

As long as the endpoints of a connection where communication occurs have valid routes to each other, AODV plays no role. When a route to a destination is needed, the source node issues a Route Request (RREQ) to find a route to the destination. RREQs are, like all AODV control messages, UDP based. RREQs are broadcast messages which may be forwarded throughout the network until the range of dissemination is reached. This can be indicated by the TTL (Time-To-Live) entry in the IP header. A route can be determined when the RREQ either reaches the destination itself, or an intermediate node with a “fresh enough” route to the destination, indicated by the destination sequence number. A route is made available by sending a unicast RREP (Route REPly) message back to the source of the RREQ.

A node using AODV may also offer connectivity information by broadcasting hello messages at regular time intervals, typically every second. Hello messages are broadcast RREPs with TTL=1. Hello messages make the AODV protocol independent of link-layer support, but can significantly add to the overhead of the protocol.

2.4.2 Route Maintenance

Each forwarding node keeps track of its active next hops. An active next hop is one where packets have been forwarded within the last ACTIVE_ROUTE_TIMEOUT milliseconds. It is also possible, for increased efficiency, to among other methods use suitable link layer notifications to determine connectivity.

If a link to the next hop can not be detected by the methods described above, the link is assumed to be broken and a RERR (Route ERRor message) is broadcasted to neighboring nodes. A precursors list of nodes that may be using an active route is maintained by all forwarding nodes for all active routes. If this list is non-empty for a route and a RERR message is received about the destination of this route, a new RERR is broadcasted by the forwarding node. This way all nodes using a route which is broken are informed of this. They can then chose to either stop sending data on this route or to request a new route by issuing a new RREQ.

3 Performance Evaluation of Ad Hoc Routing Protocols

When evaluating ad hoc routing protocols it is necessary to identify both qualitative and quantitative metrics tht makes it possible to judge the merits of the protocol. These metrics should be protocol independent, so that side by side comparisons of different protocols are possible. The rfc2501[2] from the IETF MANET Working Group[1] lists many metrics which are important for performance evaluations of ad hoc routing protocols.

3.1 Qualitative Metrics

Desirable qualitative metrics for the evaluation of ad hoc routing protocols can be (taken from [2]):

- Distributed operation. A basic and essential property for an ad hoc network.
- Loop-freedom, to avoid worst-case phenomena, e.g. a small fraction of packets spinning around in the network for arbitrary time periods.
- Demand based operation. A reactive algorithm may adapt to the traffic pattern on a demand or need basis. This can save network bandwidth and energy, at the cost of increased route discovery delay.
- Proactive operation. If the latency introduced by demand based operation is undesirable, the routing algorithm can work proactively at the cost of network bandwidth and energy conservation.
- Security. Ad hoc protocols that operate in wireless environments are prone to attacks. Sufficient security protection to prohibit disruption or modification of protocol operation is desirable.
- Energy efficiency. Due to the limited battery capacity of mobile terminals, it is desirable for ad hoc protocols to include energy saving operations, such as sleep periods when no transmission occurs.
- Unidirectional link support. In design, ad hoc protocols often assume bidirectional links for communication. But since unidirectional links commonly occur, it can be valuable to make use of those when no duplex bidirectional links exists.

3.2 Quantitative Metrics

The following is list of quantitative metrics that can be considered when conducting performance analysis of ad hoc routing protocols (taken from [2]).

- End-to-end data throughput. Statistical measurements of data routing performance provide the basic efficiency and performance measures.
- Route acquisition time. The time to establish a route can be an important measure, especially when considering reactive algorithms operating in a highly dynamic and mobile environment.
- Percentage out-of-order delivery. An external metric of connectionless routing performance. It is of particular interest to transport layer protocols which prefer in-order delivery, such as TCP.
- Efficiency. This would portray the internal effectiveness of a algorithm. Two policies can expend different amounts of overhead, depending on their internal efficiency. The efficiency can be estimated by tracking ratios like:

  - Average number of data bits transmitted/data bit delivered. This can be thought of as the bit efficiency of delivering data within the network. Indirectly it also gives the average hop count taken by data packets.
  - Average number of control bits transmitted/data bit delivered. The idea is to evaluate the efficiency of expending control overhead to deliver data. This should include not only the bits in the control packets, but also header bits of data packets.
  - Average number of control and data packets transmitted/data packet delivered. This ratio illuminates the protocols channel efficiency, as the cost of channel access is high in contention-based link layers.

3.3 Networking Context

The networking context in which protocol evaluations take place is important to consider. The context can be seen as defined by a set of parameters. Examples of parameters of this kind are (taken from [2]):

- Network size/scale. Measured in number of nodes.
- Network connectivity or "Density". Measured as the average number of neighbors per node.
- Topological rate of change. The speed of which a network's topology is changing.
- Link capacity. Effective link speed measured in bits/second, after accounting for losses due to multiple access, coding, framing, etc.
- Fraction of unidirectional links. How efficiently does a protocol perform as a function of number of unidirectional links present.
- Traffic patterns. How does the protocol perform and adapt to different traffic patterns i.e., uniform or bursty traffic.
- Mobility.
- Fraction and frequency of sleeping nodes. How does a protocol perform in the presence of sleeping and awakening nodes?

3.4 Simulations vs Real World Tests

For network protocol evaluations, both simulation and real world test are important. Both have their merits and together they provide a good picture of the functionality and performance of protocols. To make the best use of these methods it is important to assess the differences between them.

Simulations have several advantages. They are easy to perform and can be conducted by a single person, something which is often not true for real tests. Furthermore, there is a long tradition of conducting network simulations, which implies easy access to powerful tools, such as the network simulator *ns-2*. Quantitative performance metrics are generally easily evaluated in a simulated environment, since it is simple to access and collect data. The effects of networking context is also more easily evaluated in simulation than in the real world.

For real world tests, the single most important issue for the confidence of the analysis is the difficulty to replay a given scenario or repeat a test in a precise and correct manner. Due to the difficulties to reproduce results, comparisons of different routing protocols may be problematic. In a simulated environment, scenarios can be re-run and reproduced infinitely,

giving the ability to expose different software algorithms, in this case routing protocols, to the exact same networking context.

A combination of simulations and real world tests may bring together the very best of both worlds, possibly improving the results and correctness of simulation studies, such as described in section 1.4.3. Simulations have a limited ability to model the dynamic environments in which ad hoc networking takes place, and may therefore not reveal the whole picture of the routing protocols behaviors. This is especially true when the network is stressed due to heavy loads, interference or other unpredictable factors. This is an area where real world tests can contribute with new data to show the validity of simulation results.

4 The Ad hoc Protocol Evaluation Testbed (APE)

The key to any productive *real world* testbed is easy management and smooth, effective usage. Large scale tests with many participants, not necessarily familiar with the testbed, puts high demands on the testbed environment. The idea of the APE testbed is to excel in these areas, with a high degree of automation and ease of use. The tools and applications to enable this, used during tests and post testrun analysis, will be presented and explained in this section along with an general introduction to the testbed.

4.1 Test Goals

The purpose of the APE tests presented here is to examine how scaling affects an ad hoc network in terms of usability, and if ad hoc routing protocols are capable of handling loads in such networks. What happens in large networks with extensive interference, lots of packet collisions and administrative routing traffic? Will the network reach a "thrashing state" where most of the communication taking place is due to the routing protocols trying to set up peer-to-peer communication rather than sending actual data?

The central component of an ad hoc network node which is tied to these issues are the routing protocol. The purpose of APE is to highlight the differences in routing protocols and algorithms, finding issues related to ad hoc networking and routing, thus helping development and possible the selection of a standardized routing protocol for ad hoc networks.

To easily and accurately compare different protocol implementations, the test environments and movement patterns of nodes should be re-created between consecutive testruns.

Test results should also be reproduced, so that there is a high confidence in the findings.

Using students as test participants puts high demand on the interface towards the users. The testbed must be easy to handle for people not familiar with APE and the software environment must install seamlessly on users computers without any pre-test preparations.

These are the basic features of the APE testbed:

- Extensible.
- Multiple protocol architecture with runtime selection of protocol to evaluate.
- Simple and seamless installation without repartitioning.
- Scripting system for coordinated and choreographed movement.
- Powerful analysis tools.

4.2 Scaling of Tests

To examine the scalability of ad hoc networks, the APE testbed must be able to handle large scale tests. The goal has been to perform tests with up to 50 or more participant nodes - which may not seem large when referring to traditional networks. Ad hoc networks are temporary networks consisting of mobile nodes seldom in numbers exceeding the participating users. In the APE case, 50 nodes are considered to be a large ad hoc network. This imply that 50 people meet and enjoy network access to each other at the same time, something that most likely require a bigger event to take place. Furthermore, connectivity is not guaranteed between all nodes, therefore ad hoc networks easily form clusters which effectively shrinks the size of networks.

The administrative and technical issues associated with conducting tests involving 50 or more participants, in APE they are most likely students, also justifies the use of "large scale" when talking about real world tests of this magnitude.

4.3 Runtime Environment

The central part of the APE testbed consists of a slimmed down customized and pre-configured Linux distribution. This is the runtime environment for all APE scenarios. The software can be installed and launched by the click of a mouse button from a DOS supporting Windows environment or a Linux system. It contains all the software necessary to conduct a test and there is no need for re-partitioning or any other preparations.

The runtime environment is distributed as a self-extracting zip-file containing the following components:

- Linux kernel
- Boot-time initial RAM-disk image
- Root filesystem image file
- Swap filesystem image file
- LoadLin executable (DOS based Linux loader)

A filesystem image is a file containing a complete filesystem. It can be mounted and read as any regular filesystem. In Linux this is done by first associating the image file with a *loop device*[1]. The associated loop device can then be mounted as any other block device.

The APE Linux environment is booted using the DOS utility LoadLin. It has the ability to boot a linux kernel and also has initial RAM-disk support. The necessary use of loop devices severely complicates the boot process. There is no default loop device associated with the image file holding the root filesystem and it can therefore not be mounted upon booting the kernel, as it normally would. This is solved by using an initial RAM-disk holding a temporary root filesystem in RAM. Upon mounting this RAM-disk the kernel automatically executes a script in the RAM-disk image. The script mounts the DOS filesystem holding the root filesystem image file and associates it with a loop device. After that, the temporary root filesystem provided by the initial RAM-disk is replaced by the larger filesystem held in the root filesystem image on the harddisk. The boot process then continues as it would normally in a any other Linux distribution.

4.3.1 Routing Protocol

The routing protocols evaluated in the APE testbed so far is a user-space implementation of AODV [31] and an implementation of OLSR [32] as well as an implementation of TORA [33]. The AODV implementation is the one used for the tests presented in this thesis. It is a fairly immature implementation based on AODV draft version 5. Many issues with it have been discovered, especially with route discovery and setup. Serious routing protocol evaluations have not started yet, but the intentions of the APE project is to test new protocol implementations as they emerge.

4.4 Choreography

There is no easy way to reproduce results in real world tests of ad hoc routing protocols. The APE testbed use scenario choreography - the entire test scenario from data traffic to node movement are choreographed through a scripting system. Node movement is initiated through instructions to the participants. An APE testrun, also called “monkey-walk”, can be replayed accurately several times - granted the participants follow the on-screen instructions.

[1] Loop devices can be used to access filesystems within files. See the *losetup* Linux man page for more information.

A scenario is defined in a scenario file which holds movement instructions for all participants. The scenario file defines data traffic and commands to be executed during the test. The general look and semantics of the file is shown in figure 1.

```
choreography.scenario.title=outside
choreography.startup.command.0=startup
choreography.shutdown.command.0=copy_files
choreography.shutdown.command.1=pack_files

# node 0   ------------------------------------------------------

node.0.ip=193.10.133.40 node.0.ipmask=255.255.255.128
node.0.action.0.msg=Starting Spyd!
node.0.action.0.command=start_spyd
node.0.action.1.msg=You are stationary!
node.0.action.1.command=ping_node 1 2020
node.0.action.1.duration=192
node.0.action.2.msg=The test will end within 10s
node.0.action.2.duration=10
node.0.action.3.msg=The test is over. The other node is coming back
node.0.action.3.command=exit

# node 1   ------------------------------------------------------

node.1.ip=193.10.133.41
node.1.ipmask=255.255.255.128
node.1.action.0.msg=The test starts!
node.1.action.0.command=start_spyd
node.1.action.1.msg=Stand still for 10 seconds.
```

Figure 1: Excerpt from an APE scenario file.

The scenario execution engine parses the scenario file and outputs different instructions on the screen depending on the node number. Instructions have time durations before the next instruction is executed and an audio cue from computers system bell is sounded whenever a new instruction pops up on the screen. This minimizes the chance of a participant being distracted when movement should be initiated. A counter shows the time until the next instruction.

The same scenario file is used for all nodes and holds instructions for each of them. Therefore it is possible for any machine to become any node at any time. It is easy to replace or add nodes at will.

4.4.1 "Monkey-Walk"

To get a feel for how a typical testrun can be executed using the APE testbed, this section describe this process as seen by the participants.

Upon booting the testbed's customized Linux distribution, the participant is greeted with a standard Linux login prompt. The participant logs in using the "root" account, which is configured for login without entering a password. Security is not of major concern during APE tests, so this is a reasonable setup. After login, a standard "message of the day" instructs the user of the necessary steps to start the test, but only after the instructors OK.

The first step is to execute `"./start_test"` which is the only command the participant has to run. Next, a few on screen instructions performs a simple setup ensuring each node gets a unique network address. The participant has to type in its node number and is then asked to select the test to be run from a numbered list of scenarios. The computer enters an initialization phase and then halts waiting for a last "Enter" key strike before the test is started. Log files are time synchronized during post test analysis, but for the movement patterns to coincide between nodes, a reasonable synchronized start is required, typically consisting of a "ready-set-go" from the test instructor.

After the "Go" the participants follow the on-screen instructions, carrying out the testrun as described in the scenario file (figure 1). To minimize the confusion that could arise if the participant is not familiar with the movement destination there should be a pre-test scenario briefing held, familiarizing the participants with the scenario and the movement patterns.

When the scenario ends the participants are requested to upload their log files to a central *"collect node"*, by executing an upload script. The central node verifies that all nodes have uploaded their log files before proceeding to the next scenario, or ending the test session.

4.5 Data Gathering

The APE testbed has several data gathering and logging facilities which permanently saves information from testruns. The motto is "save now, analyze later" so that as much data as possible is saved for post-testrun analysis. It is possible to extend the analysis at a later time, without conducting the same tests again.

Logging and data gathering are performed at several different levels of networking layers and are intentionally routing protocol independent, so that the same source data can be collected independently of the ad hoc routing protocol used.

4.5.1 Ethernet Level

Data gathering at ethernet level consists of a combination of user space tools and kernel space routines. Figure 2 provides a logical overview of this environment.

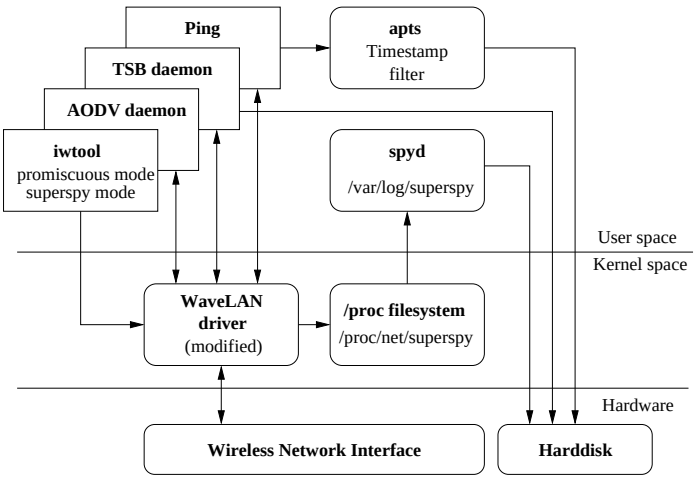

Figure 2: Logical overview of software components and data flow.

Ethernet level data is collected using a modification of the Orinoco IEEE 802.11 WaveLAN driver. A custom spy mode makes it possible to capture data from every packet the interface picks up from the ether. The new spy mode is based on the standard spy routine provided by the wireless extensions support[34] within the driver.

With the existing wireless extensions spy mode it is possible to spy on up to eight unique hardware addresses (MAC), providing signal quality of the most recently received packets from the hosts in question. Each time a packet is picked up by the network interface, it compares the source address of the ethernet frame against the list of hosts to spy on. If there is a match, the signal quality of the host is updated. To avoid driver performance issues the list of hosts to spy on is limited to eight.

The modified APE spy routine, referred to as *superspy*, goes one step further and provides a way to get information from all packets discernible by the network interface. The network interface is set to promiscuous mode and the superspy routine is enabled by a user space utility called *iwtool*. Whenever a packet is captured, signal strength, signal noise, source address and time of reception is stored in a temporary buffer. To access the buffer from a user space application, an entry in the proc filesystem is used, `/proc/net/superspy`. The output is shown in figure 3.

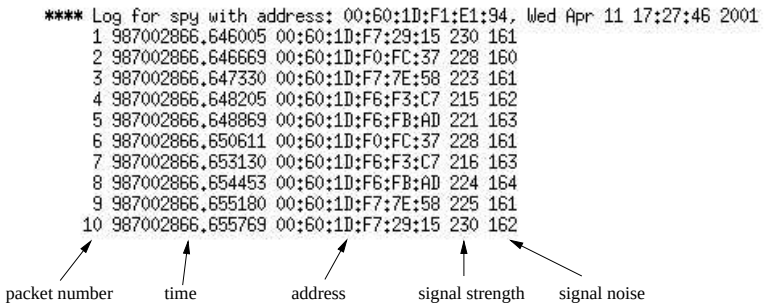

Figure 3: Output from /proc/net/superspy.

The APE spy routine is not plagued by the same performance constraints as the standard wireless extensions spy mode. Unlike the latter, each received packet is not checked against a list of hosts to spy on before it is logged. Since the routine is called for each packet anyway, the superspy routine just adds a few instructions to store the wanted information in the buffer.

To avoid that the buffer is filled and packet information is lost, a user space daemon called *spyd* continuously polls the `/proc/net/superspy` entry for new data to transfer to permanent disk storage.

4.5.2 IP Level

At the IP level, the APE testbed make extensive use of the well known PING-tool to generate traffic. Per default, the ping tool sends 64-byte ICMP ECHO_REQUEST packets to a specified host or using broadcast. With the "-R" option, ping use the IP options field of the IP header to record the route of packets. The options field is only 40 bytes large, therefore at most 10 32-bit IP addresses can be recorded. The rate at which packets are sent is easily modified. The ping tool has proven a useful tool for generating connectionless traffic while at the same time providing diversity in the data.

During testruns, nodes save their ping output to a ping log file. A header containing a timestamp is added to the top of the file when the ping command is invoked. Since ping packets are sent at regular time intervals, defaulting to 1 second, the time of each sent packet can be derived from the timestamp in the header.

Reply times of ping packets can be calculated by adding the send time and the round trip time of each packet. However, to be able to more accurately derive ping reply times a small filter program called *apts* was developed. The ping command output is piped through apts (see figure 2), which whenever a line indicating a sent packet is discovered, inserts a microsecond resolution timestamp. The final ping output is shown in figure 4.

Future development of the APE testbed will most likely add new ways to generate traffic and perform IP-level data gathering.

```
Wed Apr 11 17:28:40.342689 2001
PING 193.10.133.44 (193.10.133.44) from 193.10.133.42 : 56(124) bytes of data.
# time=987002920.540518
64 bytes from 193.10.133.44: icmp_seq=0 ttl=255 time=2.3 ms
RR:     193.10.133.42
        193.10.133.44
        193.10.133.44
        193.10.133.42

# time=987002920.987077
64 bytes from 193.10.133.44: icmp_seq=1 ttl=255 time=2.3 ms     (same route)
# time=987002921.486961
64 bytes from 193.10.133.44: icmp_seq=2 ttl=255 time=2.3 ms     (same route)
# time=987002921.987222
64 bytes from 193.10.133.44: icmp_seq=3 ttl=255 time=2.5 ms     (same route)
# time=987002922.486970
64 bytes from 193.10.133.44: icmp_seq=4 ttl=255 time=2.3 ms     (same route)
```

Figure 4: Example of ping output using *apts*.

4.5.3 Time Synchronization

Post run analysis requires that log files are synchronized. APE has the *TimeStamp Broadcaster* (TSB) for this purpose. During testruns the TSB-daemon is running on each node, transmitting broadcast TSB-packets at regular intervals. TSB packets contain a timestamp of the time of transmission and are received by any host able to hear the broadcast. When receiving a TSB-packet, a local timestamp is stored in a TSB log file (figure 5) along with the received timestamp. Using the TSB log files, it is possible to derive the time differences between the nodes' internal clocks.

```
# tsb - TimeStamp Broadcaster and listener
#
#   ip 193.10.133.40
#   mask 255.255.255.128
#   UDP broadcasting at 193.10.133.127/9332
#   delay 10 sec
#
# received-at received-from sent-at
987002568.318204 193.10.133.43 987002593.865523
987002568.373249 193.10.133.42 987002620.623909
987002573.682764 193.10.133.45 987002653.345940
987002578.318356 193.10.133.43 987002603.865517
987002578.373426 193.10.133.42 987002630.623912
987002583.680970 193.10.133.45 987002663.345939
987002588.318467 193.10.133.43 987002613.865520
987002588.373556 193.10.133.42 987002640.623910
```

Figure 5: Example of TSB output.

4.6 Data Analysis

The APE testbed provides tools to perform post-testrun analysis of gathered data. The purpose of these tools are twofold. Firstly, they aim to examine *ethernet level* characteristics of the network and testrun. Secondly, they extract *IP level* performance metrics. The formers are often protocol independent and are affected by network parameters which can be altered and used to expose routing protocols to different environments, such as mobility and network scale. In the APE testbed, ethernet level metrics are also used to "fingerprint" the test scenarios used. The latters are protocol dependent metrics that describe network usability and protocol performance, such as packet loss and throughput.

4.6.1 APE Analysis Tools

The APE analysis tools are a collection of text parsing scripts. They are mainly written in Perl[35] - a well known text processing language which provide the power of "real" programming languages such as C, combined with the ease of use and low overhead of standard shell scripting. These properties makes it a powerful tool for analysis of the text based log files like those from APE testruns. Currently scripts exists for:

- log file merging,
- virtual mobility (vM) and variations,
- packet loss ratio,
- path optimality,
- plot generation.

Handling of the data sources (log files) from nodes is simplified by creation of combined log files which analysis scripts can work on. Plot generating scripts use *gnuplot* to simplify the creation of nice graphs.

There is also a graphical frontend (GUI) available for the script tools which greatly simplifies the analysis process, especially for those unfamiliar with the tools. The command line scripts are still more suitable for doing batch type processing when dealing with massive amounts of data.

4.6.2 MAC Level Analysis

In the APE testbed *Virtual Mobility* (vM) is a characterization metric for describing the networking context (see section 3.3). Virtual Mobility is calculated from signal qualities of packet transmissions during testruns, using the ethernet level data gathered in the superspy log file (see section 4.5.1). Signal quality is mapped to a distance measure and vM is calculated as the average change in distance during a given time interval - similar to what is done for simulation in [23]. Thus vM can be calculated as a discrete function of time. The exact formula used, properties and uses of virtual mobility are thoroughly explored in section 5.

Characterizations such as vM are useful when studying properties of routing protocols in different networking contexts. For mobility, the context is the degree of movement in the network. As an example one could see how mobility affects packet delivery ratio using different routing protocols as done in [18].

Other possible ethernet level characterizations are connectivity - the average number of neighbors per node, or network density - a simple measure based on the average distance between nodes, offered network load, as described in [23] and many others. These metrics are likely to be used in APE in the future.

4.6.3 IP Level Analysis

Network performance, as perceived by applications using the network, are protocol dependent since they rely on the services provided by the routing algorithms used. Traffic generated by the **ping** application consists of ICMP ECHO_REQUEST packets to network hosts, described in section 4.5.2. Several Perl scripts have been created to analyze the information contained in the ping log file (see figure 4). Metrics currently examined are *packet loss rate* and *route optimality* (unnecessary hops).

Packet Loss Ratio

The packet loss ratio script examines the ping log file, searching for missing `icmp_seq` (packet sequence) numbers. The transmission time of a missing packet is calculated from the packet receive time recorded by *apts* (section 4.5.2) and the round trip time reported by the ping. Detailed information about which packets where lost, where they originated from and at what time they were sent (and thus lost) can be extracted.

The correlation between vM and packet loss ratio can easily be examined. Provided there exist two consecutive testruns with comparable vM-fingerprints, using different routing protocols, it may be possible to discern protocol specific differences in packet loss ratio. However, the exact configuration of protocol evaluation and comparisons are not yet thoroughly explored in the APE testbed, which is still in an early development stage. Ideally one would like to assess statements such as *"under virtual mobility V, protocol A behaves better than protocol B"*.

Route Optimality

Route optimality intends to assess the routing protocols abilities to choose the "optimal" routes. The proposed ad hoc routing protocols under consideration have varying ways of choosing routes depending on their ability to update routing information. Some protocols tend to use already known routes or the first discovered, avoiding the overhead of finding a better one, while others strive to always use "optimal" routes. Route optimality can be used to examine the effects of these implementation choices. It may or may not be worth the overhead to find an optimal route.

A metric of this kind has been used in the simulation study [18], where the shortest path available was compared to the path actually taken by data packets. It is assumed that the shortest or optimal path is the one with least number of hops. The result was a plot where the amount of packets for each extra hop away from the optimal path was shown. The APE testbed provides tools to examine this metric in real ad hoc networks.

Optimal routes are easy to calculate in a simulation, but this can prove to be a very difficult task when dealing with real world dynamic ad hoc networks. In APE optimal routes are based on the connectivity information in the *superspy* log files (figure 3). Since superspy logs are gathered from all participating test nodes it is possible to re-create the direct neighbor relationships between nodes at a certain time of the test. By choosing a sufficiently small time interval, a node's direct neighbors during an interval are the nodes from which packets have been intercepted. The interval size should be wisely chosen - it should be small enough to capture node movements and thus changes in connectivity, but not so small that there is no confidence in that a node would have received packets from all nodes within its radio range. The interval could be adjusted according to traffic load or interval between hello beacons of routing protocols.

With the availability of connectivity and direct neighbor relations between nodes, theoretical "optimal routes" between nodes can be calculated. Signal quality is mapped to a "virtual distance", so that the shortest virtual path between nodes can be derived from the direct neighbor relationships. The routes are optimal in a quality sense, as perceived by individual nodes. Therefore, a route of three hops may not traverse the same intermediate nodes in both directions. Neither is it necessarily true that the route has the same number of hops in both directions.

The path optimality in [18] is based on excess hops compared to the shortest path when routing packets. Ad hoc networks are dynamic and unpredictable environments, so the physically shortest route in distance or the one with least number of hops is not necessarily "optimal" in a radio sense. Therefore, the optimal route metric in APE may fit the ad hoc environment better.

However, it is possible to choose the definition of "optimal". Either an optimal route is the minimum hops necessary to reach the destination or it is the path with shortest "virtual

distance", i.e. the route that provides the best quality when sending data packets. However, one would expect these two definitions of optimal to refer to the same route in most situations.

The routes of packets can be extreacted from the ping log file (figure 4), provided ping is used as the traffic generation tool of choice. Ping use the *record route* option of the IP-header to record the route of each packet. Route optimality may be evaluated by combining and comparing the theoretical optimal routes based on "virtual distances" and the recorded route of the IP-protocol.

4.6.4 Data Visualization

A graphical Java-based visualization tool has been created to display network connectivity during APE testruns, called APE-view. APE-view shows node movements by using data from the ethernet level logs. Logical node placement is calculated by minimizing the difference between the geometric distance and signal strength of nodes. Thereby, virtual positioning and logical connectivity is shown in an intuitive way. Nodes are shown as small squares with their unique node number in it. Connectivity between nodes are visualized with lines drawn between nodes where packet traffic occur. A screenshot of APE-view replaying one of the scenarios currently used in the testbed is shown in figure 6. This scenario is described in section 6.1.3 and is more thoroughly analyzed in section 6.2.

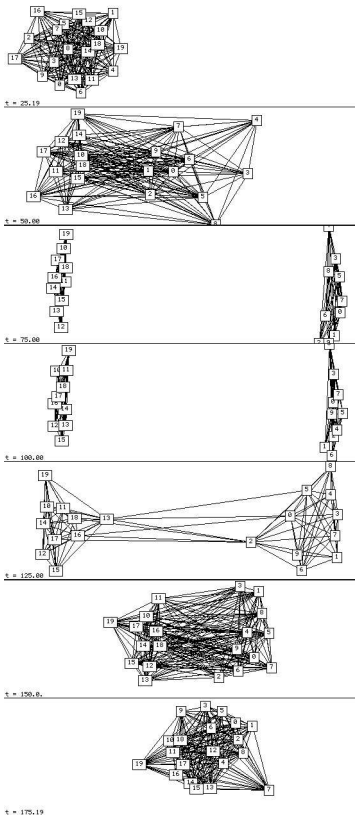

Figure 6: APE-view showing different snapshots from the *Lost 'n' found* scenario.

APE-view also provide an intuitive way of evaluating the characteristic movements of a given scenario. As seen in figure 6 APE-view quite efficiently show clustering of nodes and their individual movement relative other nodes.

5 A "Virtual Mobility" Metric

The APE testbed introduce a way of describing mobility based on the fluctuations of signal strength in radio transmissions. Physical change in distance between nodes is not necessarily portrayed since SNR might change due to reflections, background traffic, or objects suddenly blocking the line of sight between communicating nodes. Hence the emphasis on *"virtual"*. However, this may be a good thing, since vM portrays the network as seen by individual nodes and reflect nodes relative movement.

For the network as a whole, vM gives an estimation of how "stressed" the network is. It can be argued that since stress may not solely depend on the networks real mobility, another name, without references to mobility, would be more appropriate. However, since many parallels can be drawn to similar metrics used in simulations, it seemed appropriate to keep the mobility reference in the name. It can also be seen by the results of vM measurements for some of the testruns, that lots of movement (high mobility) in fact makes a large impact on vM.

A graph representation of "virtaul mobility" gives a *"fingerprint"*, showing the uniqueness of the test. Comparing fingerprints is an intuitive and hands-on way of evaluating the success of reproducing a testrun in a connectivity and movement sense.

The following sections will start with a short introduction to mobility models used in other research, mostly simulations of ad hoc networks and protocols. It will continue with an introduction to the formula and mechanisms of the virtual mobility metric.

5.1 Mobility Models

Many simulation studies of ad hoc networks like [18, 22, 23, 36] have in common that they define a network mobility metric to characterize network movement. Properties and behavior of routing algorithms are studied as mobility increase or decrease. In simulations, the purpose of the mobility model is to generate node movement to drive the simulation. This is the reverse situation when comparing to real world tests, where movement is available but a model is not.

Mobility models in simulations are often stochastic with random selections of movement directions and speeds. The most commonly used is the *random mobility* model. According to this model, speed and direction of a node is randomly selected every simulation interval and has nothing to do with the speed and direction of the previous interval. This model can generate unrealistic movement, with sharp turns and instant speed changes.

Broch et al, in the CMU simulation study [18] use a movement model called *random waypoint* which is an extension of the random walk model. Nodes randomly choose a waypoint where they stay for as long as the predefined *pause time*. Nodes move at a speed uniformly distributed between [0, MaxSpeed] every time they select a new destination. By repeating simulations using varying pause times, different mobilities are generated. A drawback of this metric is that it does not see uniform movement of a network cluster as having zero mobility, since it is only dependent on the nodes pause times. Thus, a clustered network scenario with uniform movements can have a mobility similar to that of a non-clustered network with chaotic movements, provided they have the same waypoint pause times. Since we are not interested in actual movement or mobility, but how it affects the propagation of data traffic in the ad hoc environment, this model is not well suited for our purposes.

Johansson et al [22, 23], use a more complicated algorithm to compute mobility, averaging distance changes between nodes at discrete time intervals. The mobility is dependent on distances between nodes at fine granularity time steps of 0.1 seconds, which is impossible to achieve in real world tests. The factor used is calculated for a complete testrun yielding an average estimate of the mobility. Johansson et al also shown that the mobility factor is directly proportional to the number of link changes during the testrun. This effectively renders the mobility calculation redundant since you can count link changes instead.

An intuitive way of looking at mobility in ad hoc networks would not necessarily relate to link changes since mobility could be experienced anyway, still affecting the network with varying transmission quality over established links.

There exists many more models which researchers have used to describe mobility in ad hoc networks. Often they are modifications of the *random mobility* or *random waypoint* models, with a varying degree of complexity and exactness.

5.2 Virtual Distance

The vM metric in the APE testbed is, like the one used by Johansson et al in the simulation study [22] (and [23]), based on distances between nodes. But unlike simulations, distances are not easily available in real world tests. The CMU testbed described in [30] and [29] use GPS positioning providing exact locations of each participating node throughout the testrun. Hence, distances can easily be calculated. As discussed in section 1.4.2, GPS positioning has several drawbacks. In particular, cost and the inability to perform indoor tests is the main reason for GPS to not be used in the APE testbed.

Instead, *"virtual distances"* based on signal quality from the ethernet level logs (section 4.5.1) are used. The purpose is to provide a virtual distance as perceived nodes participating in the tests.

5.2.1 Path Loss Model

The *path loss model* [37] provide a mapping between signal quality and distance. The far field case (indoor) yields a path loss coefficient of 3.3:

$$Q\ in\ dB = \alpha - 3.3 \times 10 \times log\left(\frac{dist}{\beta}\right) \tag{1}$$

Calibration using the signal quality range of the IEEE 802.11 WaveLAN cards combined with our measurements results in the following inverse path loss formula:

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

Q is the WaveLAN signal quality in the range of $0 \ldots 75$, for a packet received from $node_j$ by $node_i$. D is the distance which has a range of 0.5 to 65 m. Tests presented in this thesis often include large movements with nodes going out of reception range, therefore the path loss model is well suited for our purposes.

5.3 Calculating "Virtual Mobility" (vM)

At time interval t_k the average of all virtual distances obtained from packets received from a specific node ($node_j$) is calculated. The mean virtual distance D_j^k to $node_j$ at time slot t_k is defined as

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

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 between $node_j$ and $node_i$ as seen by the latter at time interval t_{k+1} is the change in mean virtual distance

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

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

$$vM^k_{avg}(node_i) = \frac{1}{S}\sum_{l=1}^{S} vM^k_l(node_i) \tag{5}$$

where S is the number of nodes contributing to the vM of $node_i$.

The *network virtual mobility* is calculated for time t_k as

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

where N is the number nodes in the network.

5.4 vM Variations

The virtual mobility analysis tools consists of a combination of Perl scripts. Logged data are streamed through a "pipeline" where each script has a specific calculation task before sending its output to the next script in the pipeline. This way modifications in the vM calculation can easily be implemented by inserting a modification script somewhere in the pipeline. Different ways of calculating vM have been evaluated using this functionality.

5.4.1 Multi-hop Virtual Mobility

Critical links can be given greater importance in the vM calculation by extending the metric to include node-to-node distance calculations over multiple hops. By summarizing virtual distances over theoretical communication routes it is possible to calculate distances between nodes without direct radio contact. They become "multi-hop neighbors". The result is that all nodes theoretically reachable by a node contribute to the individual node's perceived mobility. The formula presented in section 5.3 can be used without further modification using the distances to these new "neighbors". If there exists several different paths between two nodes, it is possible to decide one by either picking the one with shortest "virtual distance" or the one with least number of hops, depending on preference. A third option would be to use the path selected by a routing protocol.

Multi-hop virtual mobility will yield high mobility when nodes part of a critical route move. A link or route is critical if many nodes depend on it for their communication. If an intermediate node move, it will affect the mobility of all nodes that could theoretically use the route for communication.

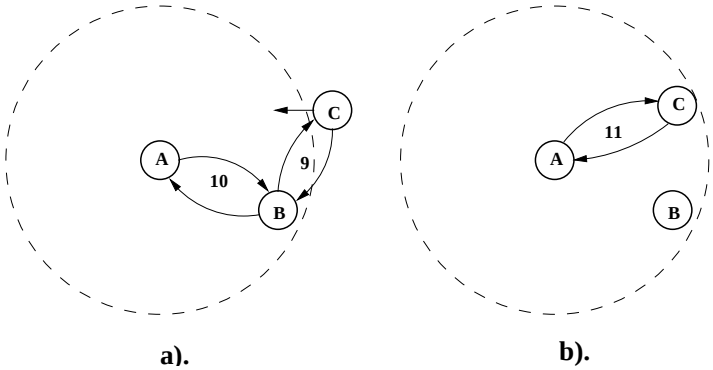

Figure 7: Example of multi-hop vM mechanism.

There are also some issues when computing multi-hop virtual mobility. Consider three nodes A, B and C with node B forwarding traffic between node A and C as shown in figure 7 a). Node C is moving in the direction shown by it's movement vector.

When node C move into reception range of node A's transmission, shown as dotted circles, as in 7 b) they become direct neighbors. Their multi-hop virtual distance goes from 19 to 11, which if experienced in two consecutive time intervals could indicate relative large virtual mobility compared to the actual physical movement. It could however be a valid vM change as seen from the nodes' perspectives, since the large vM could be the way they perceive the reality in a connectivity sense.

6 Performance Measurements

Initial results of testruns ranging from 19 to 37 nodes will be presented here. The tests have been performed during march 2001. Analysis is focused on virtual mobility which forms the basis for characterization and verification of repeated testruns. All tests have been conducted at least two consecutive times to verify vM-fingerprints and the repeatability of choreography driven movements.

6.1 Test Scenarios

Figure 8 depicts a conceptual overview of some of the scenarios developed for the APE testbed.

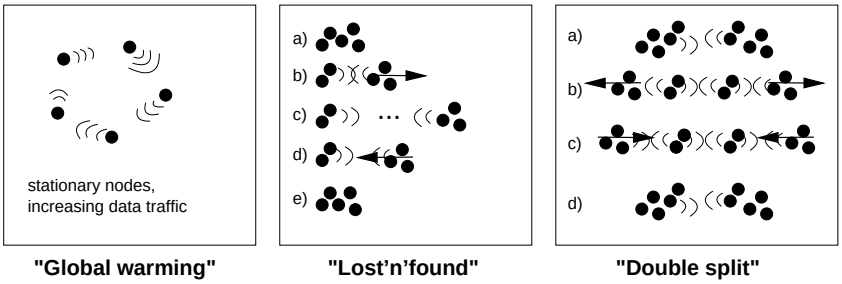

Figure 8: Conceptual image of scenarios showing node movements.

Scenarios scaling from 5 up to 50 nodes has been created, but actual tests has been performed with at most 37 nodes due to technical and administrative issues. The scenarios have been constructed to examine scaling of network clusters with varying number of nodes in ad hoc environments. Nodes move within radio range and to the point where communication cannot be withheld anymore.

Although the scenarios described here are simple, it should be stated that there is no limit to the complexity of a scenario. Each node can have a unique movement pattern and can generate traffic independently. Initial tests aimed for simplicity in the scenarios but using a large number of participants for scale.

Scenarios typically span up to about five minutes. They can be of any length, but five minutes have been found to be close to the limit where students willingly follow the predefined choreography in a satisfactory way. Nodes move at normal walking speeds, slightly above 1 m/s.

6.1.1 Physical Environment

The test scenarios have so far been conducted indoor at the Polacksbacken campus of Uppsala University, shown in figure 9. Three buildings connected with walk-bridges have been chosen as the location where the tests are conducted.

The indoor environment has sufficient space to move in and out of communication range, with both line-of-sight communication and effective dampening of radio signals behind thick walls and corners. Corridors have a length of 56 meters in building 1 and 2. The middle building has a length of 28 meters while the bridges are each 17 meters long. Widths of corridors and bridges varies between 2 and 4 meters. Any descriptions of locations in the following sections, refer to those found in figure 9.

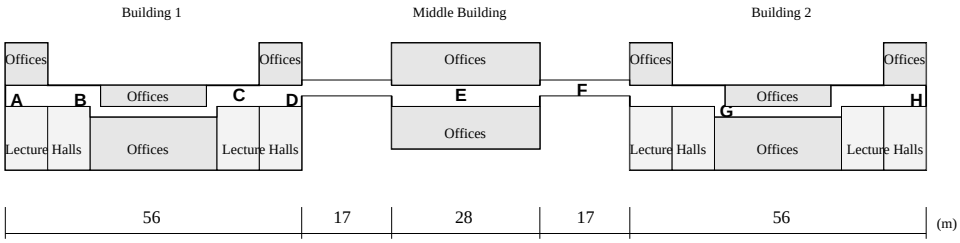

Figure 9: Test environment at Polacksbacken campus.

6.1.2 Scenario 1: "Global Warming"

This is a very simple scenario which aim to examine how increasing traffic affect virtual mobility. Stationary nodes are gathered in a room (see figure 8). One node starts to send broadcast "pings" with the others consecutively joining in over time, effectively increasing the network traffic over time. No user interaction or movement is really needed. A variant is also to have users walking around at the spot, to see how moving persons or objects affect virtual mobility.

6.1.3 Scenario 2: "Lost 'n' Found"

In this scenario a group of nodes split into two clusters, as illustrated in figure 8. They move out of radio contact range for some time, before moving back to join at the start position. The purpose is to show the effect of homogeneous movement on virtual mobility. At the start of the test, the initial node grouping is located at point D. After 30 seconds the group split into two clusters, with one moving away towards point A. The other cluster remains at location D. Upon reaching point A at 75 seconds, both clusters should have lost radio contact with each other. The remote cluster will remain at point A for 105 seconds before moving back. The two clusters re-join at point D at time 150 and will remain there for 30 seconds before the end of the test. During the test, nodes send broadcast pings.

6.1.4 Scenario 3: "Double Lost 'n' Found"

This scenario is a modification of the "Lost 'n' found" scenario. Two clusters move away from a third. Radio contact is lost at the same time, while radio contact is re-established at different times. All nodes start out at point E, similar to the "Lost 'n' found" scenario. After 30 seconds the group splits into three equally sized clusters, with two of them walking away towards point A and point H respectively. The third cluster remains at point E. At time 105 the two moving clusters will have reached their destinations and all radio contact between clusters will have been lost. At time 135 one of the distant clusters starts moving back towards point E and should arrive at time 210. The other distant cluster move back at time 225 and should have returned to point E at time 300. The test ends at time 330. Unlike the "Lost 'n' found" scenario, the nodes now use unicast pings between clusters.

6.1.5 Scenario 4: "Double Split"

Two clusters of nodes are located at point C and F. They should have radio contact with each other. 50 seconds after the start of the test the two clusters split into a total of four clusters. One of the two clusters at C starts moving towards point B while one of the two clusters at point F starts moving towards point G. At time 70 the two moving clusters should have reached their distant locations and they will remain there for 50 seconds. At these locations all clusters will only have radio contact with its adjacent cluster(s). Traffic between point G and C, G and B and so on must be routed over intermediate clusters. At

time 120 the remote clusters start moving back to join with their former cluster members at their starting positions. They will have reached point C and F at time 140 and will remain there for 50 seconds until the end of the test.

6.2 vM Analysis

In this section virtual mobility is shown in a series of graphs for some of the scenarios described in section 6.1. A mean vM line shows the average network virtual mobility calculated over all participating nodes during the test. There are also plotted lines showing the lower and upper quantile of the virtual mobility. That is, the combined vM of 25% of the nodes with lowest vM and in the same manner the vM of the 25% of the nodes with highest vM. They are referred to as **vM-low** and **vM-high** respectively. The purpose is to show the *heterogeneity* of the different scenarios and the diversity of the vM metric.

At stand-still, the virtual mobility will seldom, or never, reach a zero-level, due to the normal fluctuations of radio signals even at close range. The discrete time interval used for vM-calculations described in section 5.3 is 0.2 seconds. The choice of this small time interval ensures any movements are captured, but it will at the same time allow enough packets to be captured by each node so that signal qualities from all neighbors are available. Choosing a to small interval will introduce false vM, since there may be two consecutive time intervals where one contain no packets while the next one does. Choosing a to large interval (> 1 second) could result in that movements visible in the virtual mobility are smoothened out and averaged over the interval, resulting in loss of information.

6.2.1 Lost'n'Found

Figure 10 shows vM for the *Lost 'n' found* scenario with 20 nodes. Two significant peaks are present at time 60 and at time 125. This corresponds well to the scenario defined movements. At the first peak nodes of the two clusters start losing radio contact and then coming to a halt again, resulting in a drop of vM between the two peaks, almost back to the initial levels. At this time the network is separated into two clusters. The low-vM would then display the intra-cluster communication while the high-vM would be a result of the inter-cluster communication which is higher and considerably more unstable due to the unreliable signal levels at those distances.

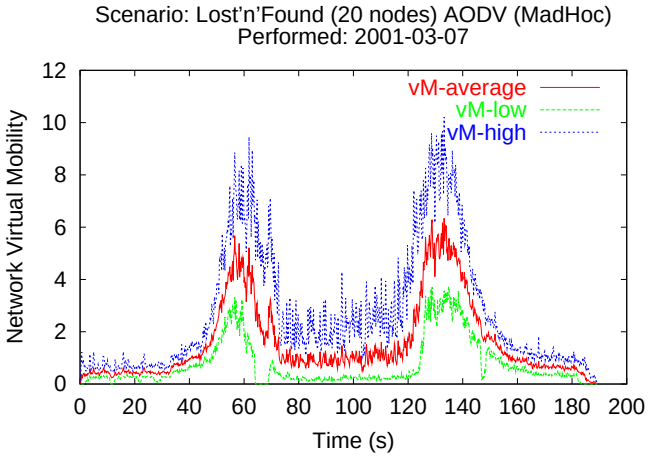

Figure 10: 20 node Lost 'n' Found vM.

In the same manner the low-vM and high-vM shows the intra- and inter-cluster movements during the peaks, where inter-cluster vM is considerably larger due to the increase in

distance between nodes, while intra-cluster vM only show random movements within the cluster during the separation. The second peak indicates the clusters moving back towards each other again coming to a halt around time 160.

Snapshots from a replay of this “Lost ’n’ found” scenario in APE-view is shown in figure 6. They are taken every 25 seconds so that the different movement states are effectively shown. At 25 seconds all nodes are standing still together, while at time 50 half of the nodes are moving away from the others corresponding to the first peak in figure 10. Nodes 0 to 9 are the ones moving, which can be seen by the movement induced in-cluster scattering, compared to the other cluster. At snapshot time 75 and 100 the two clusters that have formed are completely separated without any radio contact or packets coming through. This can be seen by the absence of connectivity lines between the clusters. At time 125 the distant cluster has started to move back again and some of the nodes has regained contact with the other cluster. This corresponds to the second peak in figure 10. The next snapshot at time 150 shows that the nodes have almost returned to their starting position. The last snapshots show the nodes 5 seconds before the end of the test and shows similarities to the first as expected.

6.2.2 Double Lost ’n’ Found

Increasing the number of nodes from 20 to 33 and using the slightly modified Lost ’n’ found scenario called *Double lost ’n’ found* (see section 6.1.4) results in the vM-fingerprint showed in figure 11. There are now three clusters where two of them are lost simultaneously but found again at different times.

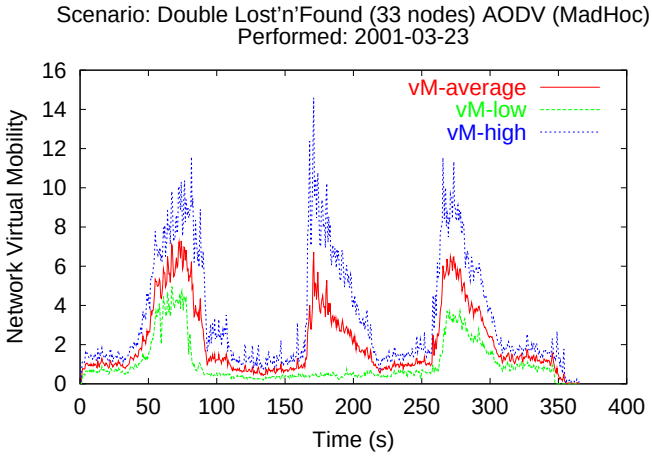

Figure 11: 33 node Double Lost ’n’ Found vM.

Similar to figure 10 there are peaks when the nodes split and move to their distant positions at time 80 and when each of the two remote clusters move back at time 170 and 260 respectively. The first peak representing the split clearly has the highest mobility, which seem reasonable considering the movement of two clusters (2/3 of the nodes) at the same time. The second peak represents the return of one of the clusters followed by the remaining distant cluster at the third peak. Again the vM-low curve holds some interesting information. When the first remote cluster return at around time 170 the vM-low curve has no peak, as during the other two major movements. Since the two remote clusters have no radio contact with the middle cluster at their distant location, only the one moving back and the middle cluster will contribute to the vM when they regain their radio contact. The remaining cluster will therefore not experience any mobility. It has been verified that it in

fact are the remaining distant nodes that account for the vM-low curve at the middle peak by manually looking in the log files.

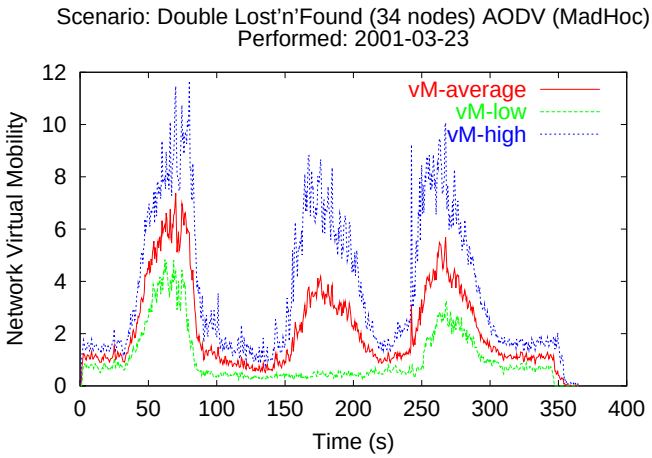

Figure 12: 34 node Double Lost 'n' Found vM.

The analysis show that virtual mobility indeed clearly show distinct movement patterns and that it is also possible to discern the homogeneity or heterogeneity of the network mobility. To examine the reproducibility using choreographed movement all test have been run twice. Figure 12 shows the network virtual mobility from the second testrun of "Double lost 'n' found". Unfortunately, a bad computer made the first testrun only comprise 33 nodes compared to the 34 of the second testrun. None the less, figure 12 and 11 show strong similarities. Differences are certainly visible, but the "fingerprints" match well and reproducible testruns seem to be achievable.

6.2.3 Double Split

Tests with the *Double split* scenario will show how scaling affects virtual mobility. The plots here show testruns with 19 and 37 nodes (figure 13 and 14).

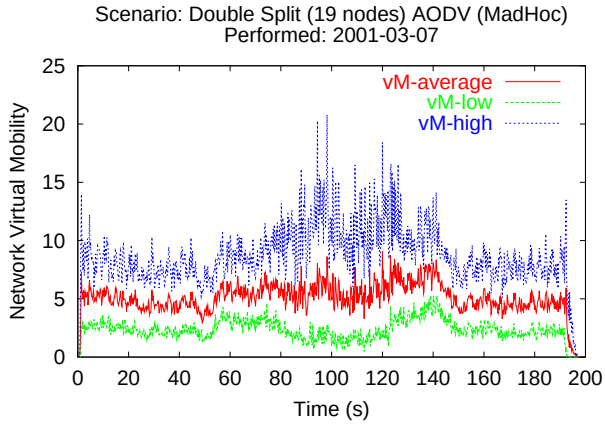

Figure 13: 19 node Double Split vM.

The scenario choreography of the larger test was somewhat different from the smaller,

to cope with the increased number of nodes. Walking times were increased with 15 seconds for each of the different movements. Departure and arrival times therefore increase to 85 and 135 from those described in section 6.1.5. The return with arrival at the original location is increased with 30 seconds to 170. For the sake of comparison the double split vM-graph is truncated at time 200, although it ends at time 220. The traffic intensity was also changed from 5 pings/second to 2 pings/second for the 37 node setting and consists of unicast communication from each of the clusters to nodes in the other clusters.

The 19 nodes double split vM-graph is shown in figure 13. The scenario starts and ends with two groups of nodes separated so that they have radio contact but with very low quality, which is visible at times 0-50 and 140-190. During time interval 70-120 the network consists of four separated clusters with poor connectivity with adjacent clusters. It can be seen in figure 13 that vM is much higher in the stand-still phases than compared to the previous graphs, which is the result of the fluctuating signal qualities at that distance. It is possible to make out node movement at times 50-70 and 120-140, although not as clearly visible as in previous scenarios. The high stand-still vM effectively smoothens the movement peaks. At time interval 70-120 the nodes has gone from two clusters to four smaller clusters all separated so that signal qualities are low. This increases the number of unstable links and thus contributing to an overall higher vM. One might say that vM has the property of increasing with decreased network density.

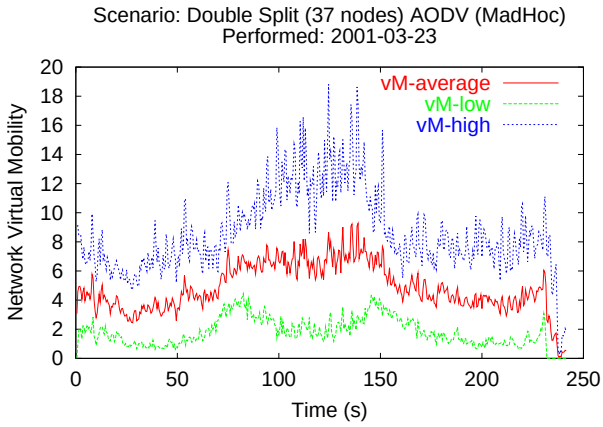

Figure 14: 37 node Double Split vM.

Scaling Double split to 37 nodes results in the graph shown in figure 14. Minor differences in test length exists as in the scaling of the double Lost 'n' found scenario, but as can be seen from the graph it match well with the 19 node testrun. The characteristic look of the scenario is still maintained, which is a good indicator that vM can be useful to distinguish between different movement patterns, despite the scale of the network.

6.3 Packet Loss Analysis

Figure 15 show an example of what can be done with *IP level* data analysis, in this case using logged information from Ping. It shows the total ping success ratio for the 34 node *Double lost 'n' found* scenario. A success ratio of one means that all packets in the network reach their destination and a success ratio of zero that none did.

It can be seen in figure 15 that there is an optimal success ratio at the beginning of the test when the nodes are tightly centered around the same spot. 30 seconds after the test has started the network splits into three clusters, with two moving away. At time 105 they should have reached their remote destination. Since nodes in a cluster only ping nodes

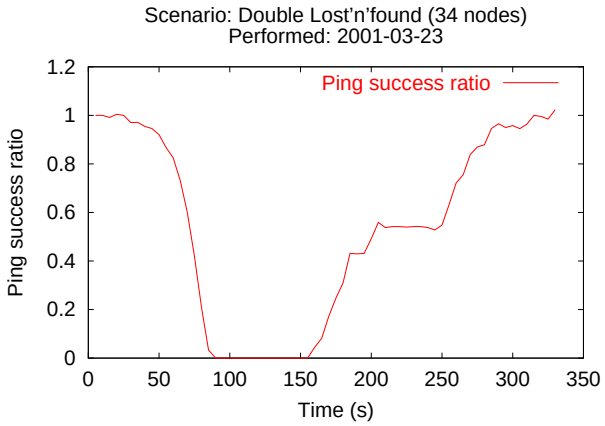

Figure 15: Ping success ratio for 34 node Double Lost 'n' Found.

within other clusters, it can be seen, looking at figure 15, that after about 90 seconds into the test no ping traffic any longer reach its destination, indicating the clusters have lost their radio contact at their new locations.

At time 135 the first remote cluster starts to move back. Radio contact is re-established with the middle cluster at time 150 and the ping success ratio settles at about 0.5-0.6 when back at the initial location. The reason for the ping success ratio to not be exactly 0.5 is due to nodes not being evenly distributed over the different clusters. At time 225 the cluster still at its remote location also starts moving back and starts to re-establish radio contact with the two other clusters at time 250 as shown in figure 15. Upon return of the last cluster, the ping success ratio is back at one again.

When comparing the ping success ratio in figure 15 to the virtual mobility in figure 12 it can be seen that the peaks indicating movement in the vM-graph coincides with the changes in ping success ratio.

7 Conclusions and Outlook

The APE project aims at evaluating the performance of ad hoc routing protocols in a real world environment. The goal of this thesis was to build the testbed foundation that would satisfy this ambition. When constructing a testbed, two major issues can be identified: The first is the construction of the runtime environment and related software. The second consists of finding ways to collect data and create analysis tools. The work presented in this thesis has addressed the first issue, and begun exploring the second.

When creating the runtime environment one important experience is that easy deployment, smooth installation and simple user interaction is important, especially when scaling tests to include many participants. Reproducibility is also important for the confidence of results both for verifying the performance of protocols and for doing side-by-side comparisons. The APE scripting system makes it possible to reproduce tests using coordinated and choreographed movements.

On the analysis side, the focus has been on constructing tools to evaluate the networking context (see section 3.3). Special emphasis is put on the *virtual mobility* metric, which is a multi-facet tool that can tell much about the networking environment, as shown in the results analysis. Metrics like virtual mobility, makes it possible to verify the success of reproducing tests, but furthermore, it makes it possible to evaluate how the quantitative metrics developed are affected by different networking contexts. Results presented in this report show how ping success ratio is related to virtual mobility in a network consisting of 34 nodes.

It can be argued that the scenarios used are too simple for any conclusions to be drawn about the performance of ad hoc routing protocols. The decision for simplicity was of course deliberate. The main goal was to evaluate how the testbed would function in practical use and during testing of routing protocols. A lot has been learned about management issues when dealing with in-between 30 and 40 participating “monkeys” and troublesome laptops of different types and brands. The APE scripting system has been put to the test and scenario files describing the movements and traffic generated by over 50 nodes.

The results presented show that with APE it is possible to satisfactorily reproduce tests in a real world environment. The initial analysis does not provide much in the way of evaluating ad hoc routing protocols, but that was not the intention. Such evaluations will be the result of continuing efforts in the APE project, and possibly the result of research world wide if people find the APE testbed useful.

7.1 Future work

As the APE testbed matures, more routing protocol tests are planned. The currently used implementation of AODV had many problems and performance results should not be applied to the AODV protocol in general. Therefore, an implementation project was started at DoCS that aims at producing a more stable and conformant AODV software. It will be used for AODV evaluations in the APE testbed when ready. There is also research on active routing protocols at DoCS within the ARRCANE project. Work is being done to integrate protocols from that research into the testbed.

There are plans for a graphical scenario generation tool, similar to *Ad-hockey*[23] for the simulator ns-2, which would greatly simplify the process of creating complex large scale scenarios for APE. An idea would be to make this tool capable of generating both ns-2 and APE scenarios, so that the exact same scenario can be evaluated both in simulation and in real life. Other connections to ns-2, like the ability to use APE logs to drive simulations are also being investigated.

New traffic generation tools are a natural next step for the testbed, to move beyond the limitations of the Ping tool. MP3 streams and FTP transfers are traffic types likely to be included.

Acknowledgments

The APE project members are (excluding the author of this thesis): David Lundberg, Henrik Lundgren, Johan Nielsen and Christian Tschudin. They deserve recognition for the work done in making the APE testbed a reality and for the support of my work on this thesis. Special recognition also goes to former member Josh Fennessey and summer intern Mattis Fjällström who helped lay the initial foundation of the testbed. Gratitude also goes to professor Per Gunningberg for supporting APE and for fundings, and the Mad-hoc team for providing us with the AODV implementation.

References

[1] Mobile Ad hoc Networks (MANET).
URL: `http://www.ietf.org/html.charters/manet-charter.html`

[2] S. Corson, J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations" *Request For Comments document, rfc2501*, January 1999.

[3] GNU General Public License, Free Software Foundation. June 1991.
URL: `http://www.gnu.org`

[4] C. Perkins, E Royer and S. Das, "Ad hoc On-demand Distance Vector (AODV) Routing". *Internet Draft, draft-ietf-manet-aodv-08.txt, work in progress*, March 2001.

[5] V. Park, S. Corson, "Temporally-Ordered Routing Algorithm (TORA)". *Internet Draft, draft-ietf-manet-tora-spec-03.txt, work in progress*, November 2000.

[6] David B. Johnson, David A. Maltz, Yih-Chun Hu, Jorjeta G. Jetcheva, "Dynamic Source Routing (DSR)". *Internet Draft, draft-ietf-manet-dsr-05.txt, work in progress*, March 2001.

[7] Philippe Jacquet, Paul Muhlethaler, Amir Qayyum, Anis Laouiti, Laurent Viennot, Thomas Clausen, "Optimized Link State Routing Protocol". *Internet Draft, draft-ietf-manet-olsr-04.txt, work in progress*, March 2001.

[8] Bhargav Bellur, Richard G. Ogier, Fred L. Templin, "Topology Broadcast Based on Reverse-Path Forwarding (TBRPF)". *Internet Draft, draft-ietf-manet-tbrpf-01.txt, work in progress*, March 2001.

[9] Mario Gerla, Xiaoyan Hong, Li Ma, Guangyu Pei, "Landmark Routing Protocol (LANMAR) for Large Scale Ad Hoc Networks". *Internet Draft, draft-ietf-manet-lanmar-02.txt, work in progress*, May 2001.

[10] Jorjeta G. Jetcheva, Yih-Chun Hu, David A. Maltz, David B. Johnson, "A Simple Protocol for Multicast and Broadcast in Mobile Ad Hoc Networks". *Internet Draft, draft-ietf-manet-simple-mbcast-00.txt, work in progress*, November 2000.

[11] Mario Gerla, Guangyu Pei, Xiaoyan Hong, Tsu-Wei Chen, "Fisheye State Routing Protocol (FSR) for Ad Hoc Networks" *Internet Draft, draft-ietf-manet-fsr-01.txt, work in progress*, November 2000.

[12] Zygmunt J. Haas, Marc R. Pearlman, Prince Samar, "The Interzone Routing Protocol (IERP) for Ad Hoc Networks". *Internet Draft, draft-ietf-manet-zone-ierp-01.txt, work in progress*, June 2001.

[13] Zygmunt J. Haas, Marc R. Pearlman, Prince Samar, "The Intrazone Routing Protocol (IARP) for Ad Hoc Networks". *Internet Draft, draft-ietf-manet-zone-iarp-01.txt, work in progress*, June 2001.

[14] Zygmunt J. Haas, Marc R. Pearlman, Prince Samar, "The Bordercast Resolution Protocol (BRP) for Ad Hoc Networks". *Internet Draft, draft-ietf-manet-zone-brp-01.txt, work in progress*, June 2001.

[15] Yih-Chun Hu, David B. Johnson, David A. Maltz, "Flow State in the Dynamic Source Routing Protocol for Mobile Ad Hoc Networks". *Internet Draft, draft-ietf-manet-dsrflow-00.txt, work in progress*, February 2001.

[16] Chai-Keong Toh, “Associativity-Based Routing For Ad-Hoc Mobile Networks”. *University of Cambridge Computer Laboratory Cambridge CB2 3QG*,
URL: `http://citeseer.nj.nec.com/495494.html`.

[17] Andrew S. Tanenbaum, Computer Networks (Third Edition), Prentice-Hall 1996.

[18] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, Jorjeta Jetcheva, “A Performance Comparison of Multi-hop Wireless Ad Hoc Network Routing Protocols”. *Mobicom’98, Dallas Texas, 25-30 October, 1998.*

[19] The MONARCH project.
URL: `http://www.monarch.cs.cmu.edu`

[20] David B. Johnson, “Validation of Wireless and Mobile Network Models and Simulation”.

[21] David B. Johnson, David A. Maltz, Yih-Chun Hu, Jorjeta G. Jetcheva, “The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks”. *Internet Draft, draft-ietf-manet-dsr-05.txt*, March 2001.

[22] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, M. Degermark, “Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks”. *Proceedings of the fifth annual conference on mobile computing and networking, U.S. August 1999.*

[23] Niklas Hedman, Tony Larsson, “Routing Protocols in Wireless ad-hoc Networks - A Silumation Study”. *Stockholm, 1998.*

[24] 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.*

[25] Sang Ho Bae, Sung-Ju Lee, Mario Gerla, “Unicast Performance Analysis of the ODMRP in a Mobile Ad Hoc Network Testbed”. *Computer Science Department, Universty of California.*

[26] The ARRCANE Project
URL: `http://www.docs.uu.se/arrcane`

[27] C. Tschudin, H. Lundgren, H. Gulbrandsen, “Active routing for Ad hoc Networks”. *IEEE Communications Magazine, April 2000.*

[28] Jochen Schiller, “Mobile Communications”. *Addison-Wesley, 2000*

[29] David A. Maltz, Josh Broch, David B. Johnson, “Quantative Lessons From a Full-Scale Multi-Hop Wireless Ad Hoc Network Testbed. *To appear in Proceedings of the IEEE Wireless Communications and Networking Conference, IEEE, Chicago, September 2000.*

[30] David A. Maltz, Josh Broch, David B. Johnson, “Experiences Designing and Building a Multi-Hop Wireless Ad Hoc Network Testbed. *CMU School of Computer Science Technical Report CMU-CS-99-116. March 1999.*

[31] The Mad-hoc Project,
URL: `http://mad-hoc.flyinglinux.net`

[32] HIPERCOM implementation of OLSR.
URL: `http://hipercom.inria.fr/olsr`

[33] University of Maryland, College Park. TORA/IMEP implementation
URL: `http://www.cshcn.umd.edu/tora.shtml`

[34] Wireless Extensions for Linux by Jean Tourrilhes, 23 January 1997.
URL: `http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Linux.Wireless.Extensions.html`

[35] Perl homepage.
URL: `http://www.perl.com`

[36] Xiaoyan Hong, Mario Gerla, Guangyu Pei and Ching-Chuan Chiang, "A Group Mobility Model for Ad hoc Wireless Networks". *Computer Science Department, Universty of California.*

[37] 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.