Improvement of IP-based MANET Emulation[1]

Henning Rogge, Alexander Wenzel,
Gabriel Klein, Marko Jahnke
Research Institute for Communication,
Information Processing and Ergonomics (FKIE)
D-53343 Wachtberg, Germany
E-mail: {rogge, lenze, g.klein, jahnke}@fgan.de

ABSTRACT

This contribution presents a MANET emulation environment that is able to combine real hardware nodes with an arbitrary number of virtual instances implemented using virtualization technologies. The environment is able to assign pre-recorded or calculated sequences of geocoordinates to the respective nodes and emulates the communication behavior of the MANET with respect to a selectable radio wave propagation model and other conditions. The environment provides flexible testing and evaluation capabilities of real-world software implementations running on MANET nodes. Our solution provides a remarkable level of realism and reproducibility while ensuring sufficient practical deployability.

KEYWORDS: Motion emulation, virtual testbed, semi-virtual testbed, reproducible node motion, node location awareness, real software deployment

1 INTRODUCTION

Testing and evaluating software for mobile ad hoc networks (MANETs) is a non-trivial task. It can be performed by using actual hardware, by simulation, or by emulation. Testing prototype software for mobile mesh nets on real hardware nodes is both expensive and inflexible. Also, most hardware testbeds are unable to emulate moving nodes and limit the reproducibility of the results because of interference by other devices and persons. Pure simulation or virtual machine testbeds may not model real hardware correctly which limits the usefulness of the results. Emulation tries to combine the advantages of both approaches mentioned before. In some wireless network emulators, only the lower OSI layers are simulated, allowing the use of real-world software and protocols on the upper layers.

Overall, very basic requirements for software testing can be formulated:

- *Realism*. To be able to test software components under realistic conditions, it is necessary to provide environments which are very close to a later deployment scenario. In MANETs, these scenarios do not only comprise the hardware and operating system platforms of the nodes and their 'typical' usage in terms of running applications and services. The motion of the nodes and the resulting dynamics (in terms of changing network topology, unstable links, and fragile connections) and other environmental influences also need to be considered.
- *Reproducibility*. For being able to compare measurement results, scenarios need to be reproducible. Obviously, the effort for moving actual MANET nodes in a representative deployment area is quite high, and guaranteeing the exactness of the motion is almost impossible. This is also the case for the external influences on the radio transmissions.
- *Representativeness*. For representative tests, the conditions for the aforementioned environmental factors need to be determined. This is relatively easy for hardware and software components. For motion sequences and the application and service usage behaviour, pre-recorded motion and traffic sequences might be deployed, but they are by their very nature restricted to a snapshot of a real-world scenario. Using statistical models, based on a sufficient number of observations, can be more appropriate for evaluation purposes.

Within the collaborative research project 'RITA' (Responsive Intrusion detection for Tactical Ad

[1] To appear in: Proc. of the Military Communication Conference (MCC2009), Prague, Czech Republic, Sep. 2009.

hoc networks, [9]), different interconnected software components are being developed and need to be evaluated with respect to the requirements specified above. In this context, it was desired to have both a larger number of MANET nodes for realistic scenarios and a smaller number of costly hardware nodes for demonstration purposes in one single setup that is able to be switched into different operation modes. As a result of our development efforts, we have created the Motion-Emu^{NG} emulation environment that is able to cover most of the requirements.

The rest of this contribution is structured as follows: Section 2 discusses different approaches to face the above challenges as published in literature. In section 3, we explain the architecture of Motion-Emu^{NG}. Section 4 presents three different application setups that we have used for evaluation and demonstration purposes. In section 5, we present performance details of our implementation. This is followed in section 6 by a discussion of the advantages and disadvantages of our solution. Finally, section 7 concludes with results that have been achieved so far, and points out different important aspects of our future work.

2 RELATED WORK

We looked at a number of free software tools which have been used for mesh net research, but each of them had some issues with our existing project work.

NS-2 [17] is one of the main simulation tools for mesh network simulations and can also be used for network emulation. It was designed for simulating and emulating multiple types of wired networks, but has also been adapted for wireless networks such as IEEE 802.11. The University of Magdeburg, Germany, has published an extension [13] for the emulation mode to allow NS-2 to be used as a network backbone for a number of virtual machines. Unfortunately, NS-2 was not designed for real-time emulation, therefore the performance is low. Together with the complexity of the combined C++/OTcl source code which prevents easy modification this prohibits its use for large-scale MANET emulations.

The Netem framework [21] has been a part of the Linux kernel since version 2.6.8 and can be used to simulate network delay, packet drop and reordering directly on a network interface. As a kernel module it has a good performance, but it lacks important features such as node movement, the support of geographic coordinates, and the ability to integrate external hardware nodes.

NCTUns [22] is a complex network emulation and simulation tool with a graphical user interface developed by a team at the National Chiao Tung University in Taiwan. It is based on a modified Fedora Linux core and supports multiple protocols and standards on different OSI layers. It can be used to connect real software instances with simulated networks in real-time. However, its complexity and the need for a specific operating system version make NCTUns difficult to integrate into existing testbeds.

A comparable approach is pursued by Ahrenholz et al. [1]. They propose a framework for emulating MANET nodes on multiple machines throughout a wired network. However, their approach requires the use of a custom virtual network stack, which inhibits easy portability.

Similar proposals [11, 15, 19] offer solutions for emulating MANET nodes, but lack features such as GPS location awareness or the ability to incorporate entire virtual operating system instances into the emulation. Virtual MANET emulations are often limited to the routing protocol itself or a few chosen applications but ignore the effects of the operation system on each node.

In our earlier work [10], we have already proposed a motion emulation framework based on a client/server design. Our previous approach was realized using client-side packet filtering by means of the NetFilter-IPTables subsystem. No connectivity to the emulation server was required after the initiation of the emulation, but all client nodes required full connectivity with each other. However, due to the Perl implementation, portability to small handheld devices was severely limited. In addition to this, the old approach was incompatible with the use of *libpcap* in client programs, which prevented us from using the same programs for emulation and real networks. This adds unnecessary complexity to prototype development and limited the comparability of emulation results with live tests.

The solution presented in this paper builds on our previous work. We follow a centralistic approach and enable easily plugging in custom radio propagation models. [12] and [14] present related solutions.

3 ARCHITECTURE

The Motion-Emu^{NG} environment is based on a packet distribution server with clients on each of the hosted nodes (see Figure 1), spanning up virtual network interfaces for applications on the node. The clients also provide the current geocoordinates to local applications using the GPS daemon interface. For each IP packet transmitted by a node, the server asks a component called "packet oracle" whether the packet is to be delivered under the current conditions (e. g. wave propagation model, position, distance, obstacles …). The only prerequisite is connectivity between the server and all clients. This facilitates the distribution of nodes participating in the emulation throughout multiple networks or using multiple hosts running virtual machines.

Motion-Emu^{NG} operates entirely on the IP layer. It is thus platform independent and does not necessitate specific layer 1 or layer 2 technologies. A very advantageous consequence of this is the ability to dynamically exchange the underlying transmission technology. For example, a wired evaluation and testing setup in which MANET nodes are connected via Ethernet cables could easily be transformed into a demonstration setup with IEEE 802.11 connectivity between the nodes.

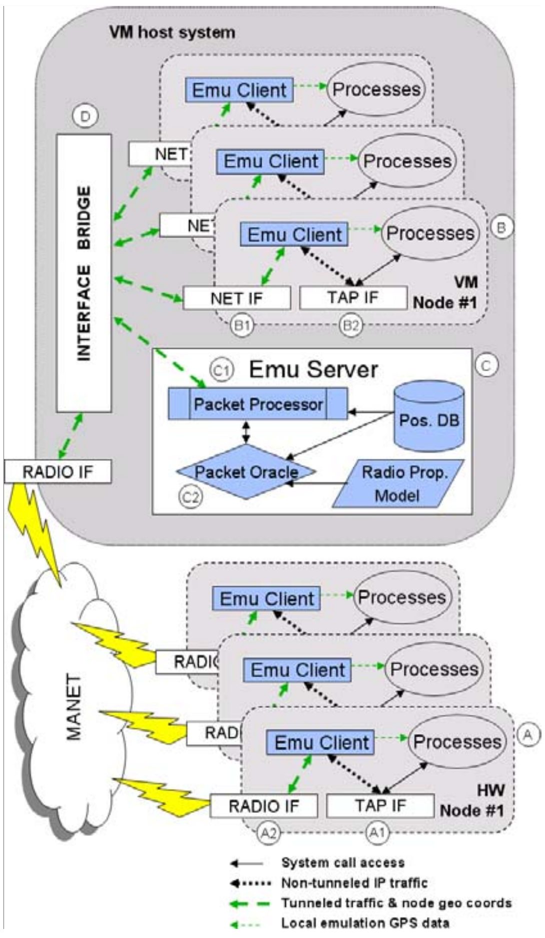

Figure 1: Motion emulator components

3.1 Emulation Server

The emulation server is the central instance which is responsible for packet relay between communicating MANET nodes. It has permanent TCP connections to all clients in the emulation, receives the packets transmitted by the corresponding nodes in the network and then decides on a per-packet basis whether the packet is dropped or whether it reaches any/all of the other nodes in range.

This decision is made by a software component called a *packet oracle*. The oracle takes an input/output pair of node IDs as its input and determines whether the packet in question will be received or not. Within the oracle, the decision is based on two factors. First, the nodes current geographic locations are considered. These discrete *waypoints* are specified in a motion scenario file and stored in a location database before the start of the emulation. Since the file contains only waypoint information, interpolation needs to be performed between two waypoints. The second factor is the radio propagation model implemented within the oracle. The emulation server currently supports three exchangeable oracles, which implement different packet drop conditions:

- *Full connectivity*. All packets reach all nodes, irrespective of the nodes' locations.
- *Limited range connectivity*. Nodes have full connectivity with no packet drops until a certain distance between them is exceeded. Beyond that, all packets are dropped.
- *Connectivity depending on radio propagation model*. The packet drop probability is based on accommodates distance-dependent effects, e. g. by log-normal shadowing, and on small-scale effects such as atmospheric disturbances, e. g. Ricean fading.

In our implementation of the third oracle type, we have realized the log-normal large-scale fading effects by using a predetermined probability distribution. Since IEEE 802.11 employs MAC-layer retransmission, the oracle simulates multiple transmissions to deliver the packet and calculate the drop rate. We have also tested an oracle with small-scale fading effects, but the results do not differ from the probabilistic packet oracle without implementing MAC-layer collisions.

Customized oracles can be easily developed and integrated into the server, because the server requires the packet oracle to implement only a single method.

In support of the evaluation requirements of the applications illustrated in section 4, we have also integrated the following features:

- Server-triggered GPS updates for each waypoint with client-side interpolation
- Scripted timer-based execution of client applications. Thus, automated network protocol and application evaluation is facilitated.
- Restarting of the server component without reconfiguration of the clients. Network components on the nodes will just detect a short period of disconnection and a jump of the GPS position.
- Logging the amount of layer 2 traffic at the emulation server.

3.2 Emulation Client

The emulation client is the emulation architecture component running on every MANET node. It creates a virtual network device which is used to communicate with other nodes within the emulation. The emulation client tunnels traffic sent to this device to the server through a permanently maintained TCP connection. There, according to the emulation parameters, the decision is made which subset of nodes is within radio range (cf. sect 3.1). The reception of packets works in a similar manner. The packets are read from the TCP connection on the real network interface and forwarded to the TAP device.

The emulation clients exhibit a great measure of resilience where loss of server connectivity is concerned. Should the connection to the server be deactivated, this results in local processes detecting a disconnection from the emulated MANET. Once the TCP connection is re-established, processes can also reconnect to the emulated MANET. There are a variety of reasons for such disconnections, for example, network fragmentation when IEEE 802.11 is used as a backbone, or when the backbone network is being reconfigured from the server side.

Similarly, client processes themselves are not affected if the server should become unavailable. This only entails consequences for communicating local applications, as the will lose their network connectivity.

During the initialization phase of the client, a local GPS daemon application can be launched which local processes can query. They can thus be made location aware by the location updates sent to the client by the server.

The client has the ability to initiate local processes. This is controlled by the emulation server and enables automatic and unattended execution of applications on the client during the scenario.

3.3 Packet Transfer Procedure

Consider the following scenario (cf. figure 1): Hardware node #1 (A) wishes to send a packet with virtual node #1 (B) as the destination. The packet is first written to the TAP interface (A1) created by the emulation client on the hardware node. Via the connected backbone network to which the node is connected through an actual radio interface (A2), this packet is tunneled to the emulation server (C). The physical location of the emulation server is irrelevant to the emulation and is dictated only by backbone network connectivity. In the scenario depicted in Figure 1, the emulation server is running on the system also hosting the virtual MANET nodes.

At the server, the packet is received by the *packet processor* (C1) and the *packet oracle* (C2) is queried whether the packet should be relayed. The oracle makes this decision based on the current geographic coordinates of hardware node #1 and virtual node #1 along with the pre-loaded radio propagation model. Furthermore, the oracle differentiates between unicast and broadcast traffic, thereby approximating the effects of IEEE 80211 retransmissions.

If the oracle decides that the two nodes are within radio communication range, the packet processor sends the packet to the interface bridge (D) located on the virtual machine host. The bridge then forwards the packet to the network interface of virtual node #1 (B1) where it is received by the emulation client and forwarded to the TAP interface (B2) used by local applications.

4 APPLICATIONS

The architecture described above offers the flexibility to easily adapt to different scenarios. The server and clients of our emulation solution may be spread throughout an entire network or hosted as virtual machines on a single server machine. In this section we illustrate two testbed instances we envision.

4.1 Virtual Testbed

This section illustrates our virtual testbed approach and describes the hardware and software setup along with measurement scenarios for which it has been successfully used.

Our purely virtual testbed consists of 15 virtual machines hosted on a server with a quad-core 2.80 GHz processor with 4 GB of RAM running Debian Lenny Linux. Our emulation framework allows us to use a standard distribution without any modifications or patches.

We employ the OpenVZ [18] kernel-based virtualization software, as it provides an easy way to handle large numbers of virtual nodes with a real software base. All virtual machines contain a regular Debian Lenny installation, although applications requiring a GUI were left out. Additionally, the RITA MANET IDS [9] along with the network management protocol SNMPv3 [4, 5] and an OLSR [6] routing daemon are installed. The containers are connected via a Linux Ethernet bridge.

The described testbed was used successfully for performing long-term measurements and evaluations of the RITA software components. Two of these are described in more detail below.

For the different measurements we used different motion models, which are loaded into the Motion-Emu^{NG} server: the very generic Random Waypoint motion model (RWP [2,3]) and a motion sequence which was generated for a very specific scenario, in our case a military hostage rescue situation (HR scenario, [9]) which is characterized by phases of different node mobility.

One of our test cases was the evaluation of a Message Engine component that is deployed on all MANET nodes and is tasked with transmitting alarm messages from locally installed detectors to a specific server node. We evaluated the time required to process alert messages in the Message Engine and measured the reliability of the alarm message transport using both motion models mentioned above. In addition to this, we also evaluated a robust flooding mechanism to deliver the messages to the server node in case of denial of service attacks.

Another evaluation we performed in the virtual testbed was that of a watchdog tool that monitors neighbor nodes in the MANET and detects whether they modify or do not correctly forward protocol packets along multi-hop routes. The aim of our tests was to improve the detection rate and minimize the occurrence of false positives by determining a notification threshold.

During both of the tests described above, we used a routing-blackhole application which broadcasts fake routing messages throughout the network to disrupt as much unicast traffic as possible. The benefit of our emulation environment is the ability to use programs implemented for attacking real-world systems.

4.2 Semi-virtual Testbed

In this section, we describe a modification of the purely virtual testbed introduced above. This setup was developed mainly for demonstration purposes but can also be used for software and/or protocol evaluation and testing.

In this setup, we combine five virtual OpenVZ nodes with Internet Tablets running on a 400 MHz ARM CPU and 128 MB memory. The virtual nodes are hosted on a notebook computer equipped with a dual-core 2 GHz processor and 2 GB of RAM.

The installed software base was identical to the one described in sect. 4.1. Additionally, the handheld devices were equipped with two applications with graphical user interface: a VoIP client and a command & control information system.

We have successfully demonstrated our hardware setup at two international events, corresponding to the two hardware configurations mentioned above.

Both at the 2008 NATO Coalition Warrior Interoperability Demonstration [7] and during a demo session at the MobiCom 2008 [16], we presented various malicious activities and attacks against tactical MANETs and the corresponding RITA intrusion detection mechanisms:

- Anomalous network traffic patterns produced by a port scan on a MANET node are recognized by CBAD [20], a detection method based on clustered traffic graphs.
- A blackhole attack on the OLSR routing protocol is detected by TOGBAD [8], which performs a centralized plausibility check of routing messages that is based on the creation and analysis of network topology graphs.
- An attack on the relaying of packets required in a multi-hop network is demonstrated by a malicious node dropping packets of a VoIP stream. The RITA watchdog monitors the relay behavior of all nodes and detects when packets are dropped.

All three attacks can be easily carried out with as-is software. The packet drop attack can be demonstrated particularly well in our emulation environment, because due to the use of real VoIP software, the loss of single or multiple packets is audibly recognized through reduced audio quality.

With the aid of our MANET emulation framework, we were able to display the effects of larger numbers of MANET nodes without having to invest in a corresponding number of hand-held devices. The ability to integrate actual hardware devices with virtual nodes facilitates more successful demonstrations.

5 PERFORMANCE

The emulation server is the central component of our architecture where packet relay is concerned. We have evaluated it with regard to packet throughput in three different scenarios, since it is potentially the single point of failure.

All measurements were performed on a Debian Lenny system equipped with a 3.4 GHz processor running 15 virtual OpenVZ nodes. In all three cases, we measured the number of packets processed by the server during a flood ping from a single node with a packet size of 100 bytes.

The first scenario uses the “Hub” oracle, which simply forwards the packets to all other nodes. We observe a throughput of 30,000 packets/s. As expected, about 1/15 are receptions (read) and 14/15 are transmissions (write). The packet transmission rate is 2,200 packets/s.

In the other two cases, the fading oracle was used together with static nodes arranged as depicted in figure 2.

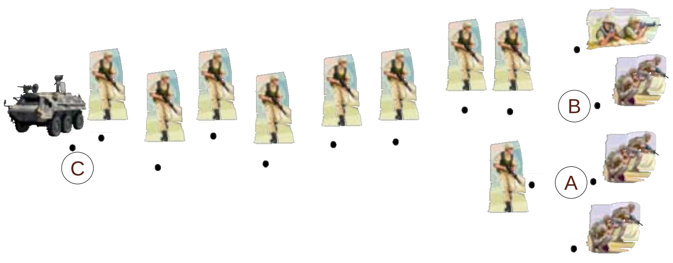

Figure 2: Hostage rescue scenario snapshot

In the first fading scenario (“Fading – S”), the ping is sent from node to the neighbor node B (single-hop). The overall performance of the emulation server is slightly reduced (29,000 packets/s), but the rate at which the client sent packets is increased by 65 % to 3,300 packets/s.

During the last scenario (“Fading – M”), the ICMP packets were sent from node A to node C. Since this involves a multi-hop route, the server has to forward the packet multiple times. The overall server relay rate is increased to 30,500 packets/s, and the rate at which packets were sent by node A has also increased to 3,700 packets/s.

In general, we observed that the overall performance of the emulation server is bounded by the number packets that it its host operating system can relay. In situations with high traffic volume, the CPU resources required by the operating system are close to 100 %. This is due to the fact that currently the relaying of packets in the server cannot be spread to multiple CPU cores. Thus, the performance of the emulation server is not dependent on the speed of the entire processor, but rather that of a single core.

Another observation we have made is that the use of the more complex oracle increased the maximum throughput measured by the sender. This is because the geographic locations of the nodes were further apart, and the packets sent by each node had fewer neighbors within radio range.

6 DISCUSSION

The use of our MANET emulator permits real-time hardware and software operation without additional patches to applications or the operating system. This ensures a high level of realism on the application level and does not carry the risk that simulation results (as in event-based simulators such as NS-2 [17]) are distorted or even falsified due to simplifications or abstractions within the simulation engine or the protocols implemented therein.

Our emulation framework was intentionally designed as a lightweight tool to make the emulation network independent of the underlying backbone network. This facilitates the inclusion of different kinds of external devices via different transmission technologies as IEEE 802.11, USB, or Ethernet, even dynamically switching between different backbone connections. This is especially useful in demonstration environments, where node mobility is an important aspect.

The fact that all packet processing and distribution logic is concentrated at a central server gives rise to multiple further advantages. This includes an easier maintenance and debugging procedure. Moreover, the server can easily be extended with different, more complex, or scenario-specific radio propagation models. This is especially useful from a military point of view where troop deployment scenarios in different locations, e.g. indoors or outdoors, warrant different radio propagation characteristics. The emulation server also provides a central logging point which can be used for the monitoring of traffic flow. Combined with the scripting of remote executing applications on all clients, this supports the validation of protocols and/or applications under development through extensive and repetitive emulation runs.

All nodes' geographic coordinates are specified at discrete locations in a motion scenario file. Location updates are sent as waypoints from the server to its clients in a synchronized manner. Clients interpolate their current location by interpolating between these waypoints and feed a local GPS daemon. Thus, applications running on the clients are constantly aware of their precise geographic coordinates when connected to the server.

When analyzing potentially very large motion scenarios, it is often desirable to focus only on specific, arbitrarily short parts of these scenarios. Therefore, our emulation solution supports fast "fast-forward" and "rewind" operations so that only the scenario's portions of interest are emulated. An additional feature is the fast or slow motion execution of the emulation. This allows the scenario to be "played" at different speeds. From a military perspective, this is especially useful, for example to demonstrate the effect of troop movement speed on connection properties. To the best of our knowledge, our approach is unique in this fashion.

Unfortunately, despite the above-mentioned numerous advantages, our approach also has certain disadvantages.

Due to the centralistic nature of our design, the server is a potential bottleneck. In situations with large amounts of traffic this could result in MANET nodes observing considerable packet delays or even packet loss if the server receives packets faster than they can be processed. Scalability could be affected by this potential single point of failure.

Furthermore, the entire packet distribution and reception mechanisms are implemented in the user memory space. The reason for this was to facilitate portability between different platforms, since no specific kernel modules need to be created. However, due to this, extra time is required to copy packets back and forth between the user and kernel memory space.

Thus far, Motion-Emu^{NG} does not emulate layers 1 and 2 of IEEE 802.11 directly, but replaces layer 1 effects with a stochastically approximated packet drop. Notably, it does not emulate layer 2 control traffic, e. g. IEEE 802.11 beacons and acknowledgements, radio signal strength, packet collisions and wireless driver runtime parameters. These aspects should be incorporated into future versions of our emulation framework.

When compared to our previous motion emulation solution, we observe several significant differences.

The filtering of packets which now takes place solely on the server was formerly performed by each client individually, which removes the need for clients to install custom-made kernel patches or modules, e.g. the NetFilter-IPTables module for packet filtering in OpenVZ.

Additionally, the need for peer-to-peer connections between communicating clients is removed, since a client needs a connection only to the emulation server. This lifts a burden on the network design, as clients can now be distributed throughout the network, possibly even in different subnets.

In our previous implementation, the connection between two clients was severed during the emulation if these clients were not reachable according to their geographic location and the selected radio propagation model. Via the backbone network, all nodes can still be accessed through a separate interface than the one carrying the emulation traffic. This makes it easy, for example, to perform maintenance tasks or manually reconfigure devices should the need arise.

The NCTUns network simulator and emulator [22] can provide similar functionality as Motion-Emu^{NG}. However, due to its requirement of a dedicated Fedora Linux server, we observe deployment issues. Also, this server does not distribute GPS coordinates and thus simulated and emulated nodes as well as attached hardware nodes are not location aware.

Other MANET emulation tools [1, 11, 15, 19] were implemented with similar goals in mind. Single Motion-Emu^{NG} features are present in many of these but lack others and thus the resulting comfort and flexibility.

The feature set of the NRL MANET emulator *MANE* [14] is very close to that of Motion-Emu^{NG}, but is more optimized for clustered virtual machines and not for combination with real hardware.

Taking into consideration the aforementioned advantages and disadvantages, we feel that our motion emulation solution fulfils the requirements for a MANET testbed. Although the effects of OSI layers below the network layer are not explicitly taken into account, they can be approximated to a certain extent by randomized dropping of packets to represent small-scale fading and calculating total success rates for multiple retransmissions of unicast traffic.

7 CONCLUSION AND FUTURE DIRECTIONS

Due to shortcomings of existing MANET emulation and simulation environments for tactical MANETS, we have developed Motion-Emu^{NG}, a platform-independent and highly flexible, motion-capable MANET emulation framework for IP-based network traffic. It can easily integrate arbitrary numbers of real and/or virtual MANET nodes into a single emulation in a dynamic manner. Node positions and movement are specified at a central server which determines connectivity between sending and receiving nodes according to their geographic locations and a pre-loaded radio propagation model.

We have carried out emulation runs for a variety of very MANET-specific evaluation and measurement scenarios for components of a MANET intrusion detection system developed in our lab and have established the viability of our emulation solution for these tasks.

For our purposes, this new framework is superior to other comparable environments in that it enables us to use as-is software components in our test cases. Combined with the ability to conduct real-time observations of running protocols and software, this increases the achieved level of realism. The addition of the GPS location distribution functionality relieves us of the need to expend resources for the distributed, synchronized generation of exact geographic coordinates, as this is performed by the central emulation server component. What is more, the server-side scriptable starting and stopping of applications facilitates the exact automated, time-controlled supervision of MANET nodes. Complex scenarios with intricate communication patterns can thus be emulated without the need for manual intervention.

Future work on Motion-Emu^{NG} includes the integration of more precise radio propagation models and the inclusion of non-MANET emulation participants such as nodes to which one or more MANET nodes has a WAN connection. Furthermore, we aim at integrating layer 2 effects such as medium collisions. On an architectural level we are planning the use of UDP channels between the emulation server and its clients, through which the use of broadcast transmissions can significantly increase the packet relay rate.

8 ACKNOWLEDGEMENTS

This work was sponsored in part by the Federal Office for Information Management and Information Technology of the German Bundeswehr, ITAmtBw.

9 REFERENCES

[1] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim. CORE: A Real-Time Network Emulator. In Proc. of IEEE Military Communications Conference, 2008. MILCOM 2008., San Diego, CA, USA, November 2008.

[2] F. Bai, N. Sadagopan, and A. Helmy. The IMPORTANT Framework for Analyzing the Impact of Mobility on Performance of Routing for Ad Hoc Networks. Ad Hoc Networks Journal, 1(4):383-403, November 2003.

[3] D. M. Blough, G. Resta, and P. Santi. A statistical analysis of the long-run node spatial distribution in mobile ad hoc networks. Wirel. Netw., 10(5):543-554, 2004.

[4] J. Case, R. Mundy, D. Partain, and B. Stewart. Introduction and Applicability Statements for Internet-Standard Management Framework. RFC 3410 (Informational), December 2002.

[5] J. D. Case, M. Fedor, M. L. Scho_stall, and J. Davin. Simple Network Management Protocol (SNMP). RFC 1157 (Historic), May 1990.

[6] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), October 2003.

[7] NATO Coalition Warrior Interoperability Demonstration 2008. Euskirchen, Germany, June 2008.

[8] E. Gerhards-Padilla, N. Aschenbruck, P. Martini, M. Jahnke, and J. Tölle. Detecting Black Hole Attacks in Tactical MANETs using Topology Graphs. In LCN '07: Proceedings of the 32nd IEEE Conference on Local Computer Networks, pages 1043-1052, Washington, DC, USA, 2007. IEEE Computer Society.

[9] M. Jahnke, G. Klein, A. Wenzel, N. Aschenbruck, E. Gerhards-Padilla, P. Ebinger, S. Karsch, and J. Haag. MITE - Manet Intrusion Detection for Tactical Environments. In Proc. of the NATO/RTO IST-076 Research Symposium on Information Assurance for Emerging and Future Military Systems, Ljubljana, Slovenia, 2008.

[10] M. Jahnke, J. Toelle, A. Finkenbrink, A. Wenzel, E. Gerhards-Padilla, N. Aschenbruck, and P. Martini. Methodologies and Frameworks for Testing IDS in Adhoc Networks. In Proc. of Q2SWinet'07, Crete, Greece, October 2007.

[11] W. Jiang and C. Zhang. A portable real-time emulator for testing multi-radio manets. International Parallel and Distributed Processing Symposium, 0:145, 2006. 17 Literaturverzeichnis

[12] B. B. Luu, R. L. Hardy, and G. T. Tran. A technique for determining radio-signal propagation in an emulated wireless environment. In Proc. of IEEE Military Communications Conference, 2008. MILCOM 2008, San Diego, CA, USA, November 2008.

[13] D. Mahrenholz and S. Ivanov. Adjusting the ns-2 Emulation Mode to a Live Network. In Proc. of KiVS'05, 2005.

[14] Mobile Ad-hoc Network Emulator (MANE). http://cs.itd.nrl.navy.mil/work/mane/index.php, November 2008.

[15] M. Matthes, H. Biehl, M. Lauer, and O. Drobnik. Massive: An emulation environment for mobile ad-hoc networks. In WONS '05: Proceedings of the Second Annual Conference on Wireless On-demand Network Systems and Services, pages 54-59, Washington, DC, USA, 2005. IEEE Computer Society.

[16] The 14th Annual International Conference on Mobile Computing and Networking, Demo session. San Francisco, CA, USA, September 2008.

[17] The network simulator ns-2. http://www.isi.edu/nsnam/ns/, November 2008.

[18] OpenVZ - A container-based virtualization for Linux. http://wiki.openvz.org/, November 2008.

[19] M. Puzar and T. Plagemann. NEMAN: A Network Emulator for Mobile Ad-Hoc Networks. In ConTEL 2005. Proceedings of the 8th International Conference on Telecommunications, volume 1, pages 155-161, March 2005.

[20] J. Tölle. Intrusion Detection durch strukturbasierte Erkennung von Anomalien im Netzverkehr. Ph.D. thesis (German), University of Bonn, Germany, 2002.

[21] G. Valadon, R. Wakikawa, and H. Esaki. Emulating small scale MANET topologies. In Proc. of OLSR Interop2005, 2005.

[22] S. Y.Wang, C. L. Chou, Y. H. Chiu, Y. S. Tseng, M. S. Hsu, Y. W. Cheng, W. L. Liu, and T. W. Ho. NCTUns 4.0: An Integrated Simulation Platform for Vehicular Trac, Communication, and Network Researches. In 1st IEEE International Symposium on Wireless Vehicular Communications, October 2007.