MiNT: A Miniaturized Network Testbed for Mobile Wireless Research

Pradipta De Ashish Raniwala Srikant Sharma Tzi-cker Chiueh
{prade, raniwala, srikant, chiueh}@cs.sunysb.edu
Department of Computer Science, Stony Brook University
Stony Brook, NY 11794-4400, USA

***Abstract*—Most mobile wireless networking research today relies on simulations. However, fidelity of simulation results has always been a concern, especially when the protocols being studied are affected by the propagation and interference characteristics of the radio channels. Inherent difficulty in faithfully modeling the wireless channel characteristics has encouraged several researchers to build wireless network testbeds. A full-fledged wireless testbed is spread over a large physical space because of the wide coverage area of radio signals. This makes a large-scale testbed difficult and expensive to set up, configure, and manage. This paper describes a *miniaturized* 802.11b-based, multi-hop wireless network testbed called MiNT. MiNT occupies a significantly small space, and dramatically reduces the efforts required in setting up a multi-hop wireless network used for wireless application/protocol testing and evaluation. MiNT is also a hybrid simulation platform that can execute *ns-2* simulation scripts with the link, MAC and physical layer in the simulator replaced by real hardware. We demonstrate the fidelity of MiNT by comparing experimental results on it with similar experiments conducted on a non-miniaturized testbed. We also compare the results of experiments conducted using hybrid simulation on MiNT with those obtained using pure simulation. Finally, using a case study we show the usefulness of MiNT in wireless application testing and evaluation.**

Keywords: Wireless experimentation testbed, System design/implementation, Hybrid simulation, Miniaturization, Measurements on IEEE 802.11b.

I. Introduction

A commonly accepted practice in network research community is to use simulation tools for testing and evaluating new protocols. The *ns-2* simulator [1] is among the most widely used simulators for networking research. In the wireless networking domain, the wireless extension of *ns-2* [2] has been developed and widely used by the research community. Ideally, the results of a wireless simulation study should closely approximate the measurements on a real wireless network of similar set-up. However, this is not always the case because of the inadequacies of the models used in simulations. It is always challenging to come up with a computationally efficient, and at the same time accurate, model that captures various aspects of wireless channels, such as radio propagation and error characteristics. Hence, when simulating wireless channels many existing non-commercial simulators incorporate an idealized and simplified radio propagation model that fails to capture the channel characteristics faithfully [1], [3].

Researchers have studied the inadequacies of existing wireless simulation tools and their impact on protocol development, especially the ones that are closely tied to the signal propagation and interference characteristics of the wireless radio channels [4], [5]. Recent interest in cross-layer protocol optimizations raises the concern further because higher layer protocol decisions are now more closely tied to lower layer feedbacks [6]. Examples of cross-layer protocol optimizations include hop-by-hop error control in multi-hop wireless networks, channel state-dependent packet scheduling, and signal strength-aware packet routing on ad hoc networks. Without high confidence in the accuracy of the wireless network simulation tools, it is difficult to make concrete progress in cross-layer protocol optimization research. In spite of the known limitations of simulation, the lack of access to real testbeds that are inexpensive and easy to set up, has forced the wireless community to depend on simulations.

Several wireless protocol implementation projects chose to validate their systems by setting up and running their protocols on custom-built wireless network testbeds [7]–[9]. Most of these testbeds are tailored toward specific research projects. These require large space and extensive management infrastructure. Moreover these testbeds are mostly inaccessible to a broader research community.

In this paper, we address some of the key inadequacies of existing simulation tools and wireless network research testbeds by developing *a miniaturized mobile multi-hop wireless network testbed* called *MiNT*. MiNT serves as a platform for evaluating mobile wireless network protocols and their implementations. Like a generic wireless network testbed, MiNT consists of a set of wireless network nodes that communicate over one or multiple hops with one another using wireless network interfaces. A key feature of MiNT is that it *dramatically reduces the physical space requirement* for a wireless testbed while providing the fidelity of experimenting on a large-scale testbed. For example, using MiNT it is possible to set up an IEEE 802.11b-based 3-hop wireless network with up to 8 nodes on a 12ft by 6ft table. This space reduction is achieved by attenuating the radio signals on the transmitter and the receiver. Through this miniaturization it is possible to substantially reduce set-up, fine-tuning, and management efforts required for a wireless network testbed. Additionally, attenuation on the transmitters reduces the interference of the testbed with the production wireless networks operating in its vicinity.

MiNT is also a *hybrid* testbed platform that enables one to run *ns-2* simulations with its link, MAC and physical layers replaced by real hardware and driver implementations. The large number of wireless network protocols and traffic models already coded for *ns-2* can thus be directly used on MiNT. MiNT allows unmodified *ns-2* scripts to be executed on a set of physical nodes. Since the effects of radio signal propagation, like multipath fading and interference, are better captured while executing simulations in the hybrid mode, it produces much more realistic results for simulation experiments.

Although the miniaturization approach has been discussed

0-7803-8968-9/05/$20.00 (C)2005 IEEE

in past, this paper is the first comprehensive study on use of this technique. More specifically, we make the following research contributions in this paper:

- We present the architecture and implementation of a miniaturized wireless network testbed, called MiNT that features mobile multi-hop ad hoc networking on a tabletop. The testbed additionally incorporates comprehensive remote management, traffic monitoring, and fault injection facilities.
- We develop one of the first hybrid simulation platforms that can run unmodified *ns-2* simulations with its link, MAC and physical layers replaced by real components.
- We verify the fidelity of the miniaturization approach and point out its limitations through extensive experimentation on an operational MiNT prototype.

The rest of the paper is organized as follows. We present the architecture and design of the key components of MiNT in Section II. In Section III, we discuss the features necessary to execute an experiment on MiNT. In Section IV, we discuss how MiNT works as a hybrid simulation platform. We prove the fidelity of MiNT and discuss its limitations in Section V. In Section VI, we compare the results of hybrid simulation against pure simulation. We study UDP Lite protocol [10] to show the usefulness of MiNT in implementing and evaluating protocols for multi-hop wireless network in Section VII. In Section VIII, we review the existing literature in testing and evaluation of wireless network protocols. We summarize the contributions and outline our future work in Section IX.

II. System Design of MiNT

In this section, we discuss the overall MiNT architecture and the design of individual testbed components.

A. Overall Architecture

MiNT consists of a collection of *core nodes* managed remotely by a *controller node*, as shown in Fig 1. A core node communicates with its peers in the testbed using an IEEE 802.11b wireless NIC that is connected to a low-gain external antenna through radio signal attenuators. The antenna is mounted on a mobile robot to enable mobility. Each core node has another optional wireless interface for the purpose of sniffing traffic and collecting packet trace. The central controller oversees the operations of all the core nodes. A core node communicates with the controller node through a dedicated network interface, that can be either wired Ethernet, or any other wireless technology, that does not interfere with the 802.11b transmissions in the testbed. Fig 1 shows a typical MiNT set-up.

B. Core Node

A collection of core nodes constitutes a MiNT testbed. As our goal is to build a multihop wireless testbed, the design of a core node is at the heart of the overall testbed design. A typical wireless testbed spans a large geographical area because the radio signal can be received over a large radius of the order of few hundred meters. In order to build a testbed that can fit on a tabletop, it is imperative to restrict the radio signal within a small space. This will enable us to set up several nodes on a table and still establish multiple collision

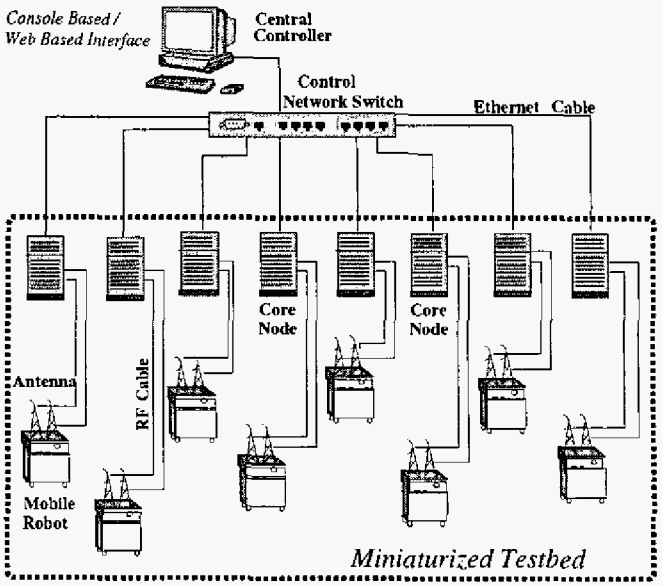

Fig. 1. The overall MiNT architecture. The core nodes communicate with the peers using a wireless NIC that is connected to a low-gain external antenna through radio signal attenuators. The antenna is mounted on a mobile robot. Another optional wireless NIC on a core node is used for monitoring the channel around the core node. A controller node is used to remotely manage the operations of the core nodes, issue execution commands, and collect statistics from the testbed. The core nodes communicate with the controller node through a dedicated network interface, that can be either wired Ethernet, or any other non-interfering wireless technology.

domains. The next important thing in the design of the core nodes is the mobility of nodes. In large-scale testbeds, mobility is introduced through use of cars or volunteers carrying the nodes. Since we are designing a testbed that can be placed on a tabletop, the space over which the nodes must move is small. Hence we improvised on the core node design to make them mobile. In this subsection, we present the design of the core nodes with respect to miniaturization of the overall testbed and mobility of the core nodes.

1) Miniaturization: The key to miniaturization of the testbed lies in limiting the radio signals within a small space. The simplest technique is to *adjust the transmit power* on the wireless interface card. One can use a laptop or a PDA with a commercially available PC card that allows setting the transmit power to different values, like 100mW, 50mW, 10mW, 5mW, 1mW. Since we are aiming to minimize the space as much as possible, we tried using a Cisco Aironet 350 series card that allows us to reduce the transmit power of the card to the smallest value possible in a commercially available card (1mW). However, experiments revealed that this transmit power setting is still too high to carry the radio signals across two mid-sized rooms. This defeats our goal of miniaturizing the testbed to the scale of a table.

The alternative choice is the use of *radio signal attenuators*. Radio signal attenuators are available in two different types, *viz.* fixed signal attenuators and programmable attenuators. However, there is a stark price difference between the two: the fixed signal attenuators are priced in tens of dollars, as opposed to the programmable attenuators which are usually close to $1000 a piece. Therefore, we choose fixed signal attenuators to design a low-cost core node. We determine the extent of

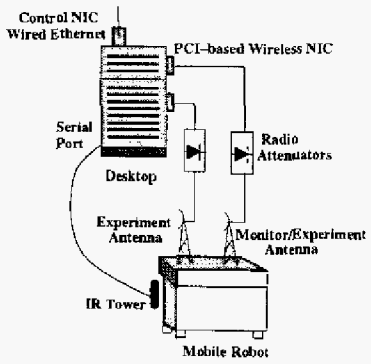

Fig. 2. Every node is equipped with at least one wireless PCI card. This PCI card does not have any internal antenna. It is connected to an external antenna through fixed radio signal attenuators. The antenna(s) is mounted on a mobile robot, which introduces node mobility.

attenuation (dB rating) based on the desired range of signal propagation. Use of attenuator, which is external to a wireless card, requires the use of an external antenna. The attenuator is connected between the PC card and the external antenna using RF cables with suitable connectors. The problem with this approach is that most commercially available PC cards come equipped with an internal antenna. The internal antenna is not fully disabled upon attaching an external antenna, and radiates significant RF energy, thus defeating the goal of miniaturization.

There are two ways to overcome this problem: one is to desolder the internal antenna, and the other is to use a card that does not have an internal antenna. It is hard to find a PC card without internal antenna; therefore, the remaining choices are using a mini-PCI or a PCI card. As most laptops/PDAs do not provide a mini-PCI socket, we finally opted for *a PCI card that does not have internal antenna*. This choice ties us currently to the use of desktop PC as the platform for a core node, but provides the flexibility to design the core nodes, such that they can be placed near each other. Moreover it is possible to place additional wireless interfaces on each node for experiments requiring multiple interfaces [11].

Our final design of a core node is shown in Fig 2. We use a desktop PC with a NetGear MA311 wireless PCI card. We connect the PCI wireless NIC to a radio signal attenuator that in turn connects to an external antenna using an RF cable. The cost breakup of the equipments per node, that is completely designed from commercial off-the-shelf equipments, is: desktop PC ($300), fixed radio signal attenuators ($100), wireless NICs ($100), and connectors ($50).

2) Mobility: We implement node mobility using mobile robots. As the desktop node itself is not mobile, we *place only the external antenna on the mobile robot.* This limits the robot movement by the length of the cable connecting the external antenna to the wireless card. In the next prototype of MiNT, we plan to use a small form-factor PC, specifically Soekris Board [13], that can be mounted on a robot. This will provide unrestricted mobility to the nodes. Current version also poses the problem of cables getting entangled during robot movement. At present, we choose a non-overlapping space of movement for each robot, thus avoiding cable entangling.

Our requirements from a mobile robot are: (i) low-price, (ii) easy assembly, and (iii) remote controllability. Hobby robots provide an inexpensive option, but require extensive assembly. We choose LEGO Mindstorms robots [12] that are priced at $200 a piece and are easily assembled. A LEGO robot is controlled from the desktop PC using an Infra-Red (IR) Tower. The IR Tower is attached to the robot so that Infra-Red signal from one tower does not interfere with another robot's movements. This is shown in Fig 2.

C. Control Node

The control node enables centralized control and management of the testbed through a console-based/Web-based remote access. The functionalities provided by the control node are used by both the administrator, as well as the users (experimenters). The administrator is primarily concerned with status monitoring and routine maintenance, e.g. software upgrades, of the testbed nodes. On the other hand, a user accessing a shared MiNT testbed deployment, requires other functionalities that let him configure each node, monitor the status of individual links, set up scripts on different nodes, and control experiment execution on the testbed. A remote management system is the underlying mechanism to enable this remote operability of MiNT.

The remote management system of MiNT is based on the Simple Network Management Protocol (SNMP) [14], where each testbed node is treated as a managed device. MiNT implements different SNMP components as follows. A software agent running on each testbed node queries various wireless NIC parameters such as transmit power and statistics such as corrupted packet count, using wireless tools [15], and updates these values into the a Management Information Base (MIB). The Network Management Station (NMS) residing on the central controller, queries these parameters using GET requests. Upon receiving GET requests from the NMS, the software agent responds by reading the stored values from the MIB. The SET requests are similarly handled except that the software agent now updates the specified parameters using wireless tools.

In order to ensure that we can collect management data while an experiment is in progress, we use a control network that is separate from the wireless network used for experiments. The control network operates on a non-interfering channel, in the current prototype over wired Ethernet. One can also use 802.11a for control channel since it does not interfere with 802.11b channels used for experiments. The wireless control interface is not attenuated, enabling each node to communicate with the control node over a single hop. This is unlike a full-scale testbed, where the control network also needs to operate over multiple hops [9].

III. Experimentation on MiNT

MiNT is a platform for testing and evaluating wireless application and protocol implementations. This requires a user to have control in configuring an experiment, executing the experiment, and finally analyzing the results using traces collected during the experiment. In this section, we present the control and analysis features we provide in MiNT for managing experiments.

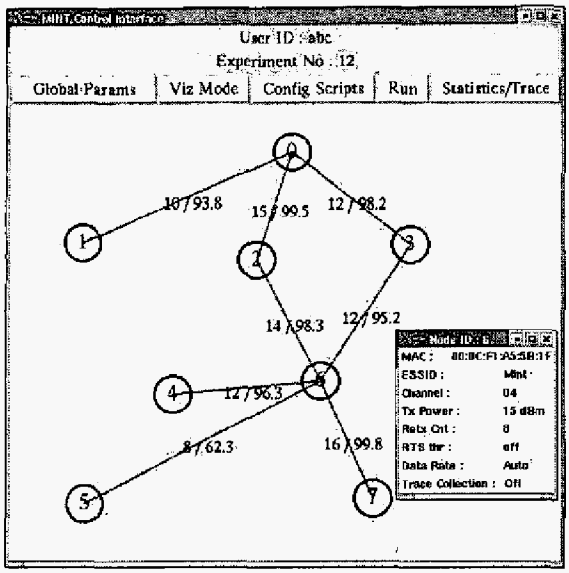

Fig. 3. Experimenter's GUI to MiNT. The position of the core nodes in the testbed is changed by dragging their associated icons in the GUI. During topology set-up signal quality and delivery rates of links are displayed with respect to the chosen node. This figure shows the signal quality and delivery rates of links relative to Node 0 and Node 6. Default node parameters can be set using "Global Params" button, and can be overwritten on a per-node basis by clicking on the associated node icon. Traffic scripts, mobility scripts, and fault injection scripts can be loaded through the "Config Scripts" button. Merged traces and network statistics can be viewed through "Statistics/Trace" button. Finally, double-clicking on a node icon opens a console window to it, that can be used to install protocol software/modules.

A. Experiment Control

Defining an experiment on any testbed involves several steps – configuring network topology, setting up applications, defining mobility patterns, and setting the required per-node parameters. MiNT facilitates this configuration through a graphical user interface (GUI) (shown in Fig 3) that can be used by an experimenter to set up and manage her experiments. In this subsection, we explain the challenges in setting up experiments and the use of the GUI.

1) Topology Configuration: In configuring a wireless network topology, an experimenter is primarily interested in the radio-connectivity between different node-pairs. This is achieved by placing the nodes in such a way that each node-pair satisfies specified link properties, like SNR or link error rate. In manual topology configuration, the user determines correct location of all the nodes to satisfy the link properties. However, with large number of nodes this method quickly become tedious. Ideally, the user should declaratively specify the topology constraints, and the node positions should be automatically calculated based on a priori measurements done on the testbed. For automated topology configuration, one can start by calculating approximate node positions from relative signal strength using multi-hop trilateration [16]. The initial positions can then be improved by iteratively changing the node locations and measuring the signal quality to achieve the desired pair-wise configuration.

Our current solution provides manual configuration facility. In the GUI, each node in the testbed is represented using an icon. A user can move the testbed nodes by dragging their associated icons in the GUI. Such a movement generates a request to change the corresponding node's position in the testbed. The testbed nodes provide continual feedback about pair-wise signal strength statistics, as well as the node coordinates. As the user moves the nodes, she can monitor the connectivity among them and re-configure the positions accordingly. For indoor position estimation an indoor positioning system, like Cricket [17] can be used.

2) Application Configuration: This involves setting up the traffic generators and traffic sinks, and can be done in two ways. The user can write her own applications. Alternatively, the user can choose from MiNT-supported library of ready-made applications, similar to the traffic sources/sinks provided by the ns-2 simulator.

3) Mobility Configuration: A user can configure node mobility by specifying (i) node trajectories, (ii) target locations, and (iii) mobility models (such as the random waypoint model and the random walk model). Mobility scripts are installed on each node using the *Config Script* button on the GUI. Since multiple nodes could be moving at the same time, the nodes could collide. The script must be validated to avoid such node collisions. At present, limited mobility of nodes prevents collision avoidance. In a fully mobile testbed methods for collision avoidance will be incorporated.

4) Setting Node/Card Properties: Changing node/card configurations, as well as installing kernel modifications are typical requirements of a user. Network-wide parameters, such as nodes' default transmit power and retransmission count, can be set using the *Global Params* button in the GUI. These parameters can be overridden on a per-node basis through the same GUI by right-clicking on the node icon. For application/protocol code that require kernel modification, we allow kernel module installation. It is also possible to remotely login to each node by double-clicking on its associated icon.

Current prototype provides users with privilege access, which is required for accessing many of the functionalities such as raw socket and broadcast socket. Providing privileged access to users makes it necessary to be able to restore vanilla conditions on all nodes once an experiment is completed. One can use Frisbee-like set-up for performing entire disk re-imaging after experiments [18]. An alternate approach to providing priviledged access is to support limited access programming interfaces providing similar functionalities.

5) Experiment Execution: The next step in experiment control is providing the user with ways to fine-tune an experiment by observing the results during execution. In addition to simultaneous start/stop of an experiment on all the testbed nodes, an ability to pause the experiment, modify parameters on the fly, and then continue the experiment, could substantially reduce experimentation time.

6) Application/Protocol Debugging: MiNT is a distributed experimentation platform, and hence an experimenter faces all the difficulties of debugging distributed applications and protocols while using MiNT as well. To address this problem, MiNT incorporates a fault injection and analysis tool, which was earlier implemented for wired network protocol testing [19]. The tool helps a developer generate realistic network

faults, like dropping, delaying, or corrupting of specific packets, using a simple scripting language. It is also possible to check for violations of protocol conditions, expressed using the same scripting language, and thus catch implementation bugs. Such facility is also useful in understanding the behavior of wireless protocols like AODV in presence of multiple errors such as control packet losses

Once an experimental configuration is finalized, the user can save all the configuration settings, such as node coordinates, applications and mobility scripts. A saved configuration can be then used to quickly and automatically set up the experiment next time onwards.

B. Experiment Analysis

A crucial component of an experiment life-cycle is its analysis stage. A network application/protocol is usually analyzed by looking at various packet dynamics during the experiment execution. MiNT incorporates a full-scale packet trace collection, aggregation, and visualization facility to aid this analysis.

1) Trace Collection: Network sniffers, such as *tcpdump* and *ethereal*, are standard tools for Ethernet-layer packet capture. One can additionally switch a wireless card to the *RF monitor mode*, where it can capture all *802.11 link-level* transmissions including 802.11 protocol headers and control frames.

In a distributed environment multiple monitor nodes are required to collect the entire network trace [20]. In MiNT, each core node also performs the monitor function using an additional wireless interface. This approach is most accurate in reconstructing each testbed node's view of the wireless channel during an experiment. It is also possible to separate the monitoring facility from experiment nodes. This requires strategically placing the nodes to completely cover the signal space of all the nodes. Additionally, the packets observed by a monitor node could be different from those seen by an experiment node.

2) Trace Aggregation: The trace collected on each node is sent to the central node over the control network. Here all the traces are merged based on timestamps. This merge step requires that all nodes be synchronized at the beginning of any experiment. It is possible for the same packet to be captured by multiple monitor nodes. The duplicate packets are eliminated to create the final trace.

3) Trace Visualization: Trace visualization shows the transition of packets with respect to time. Visualization could be real-time or offline, depending on whether the collected trace on individual nodes are transported and aggregated while the experiment is running, or at the end of the experiment. Real-time visualization requires that parse, collate and display operations be done in real-time. Display of the network-wide packet dynamics must show the packet exchanges over time for each node. Also, different frames, like control, management and data frames, must be highlighted separately for ease of understanding. The current MiNT prototype supports offline analysis, and uses Ethereal for visualizing the aggregated trace.

4) Data Filtering: Another useful element of experiment analysis is set of filters used to reduce the amount of trace collected on each node. This aids the online visualization of trace by reducing the amount of traffic that must be transferred in real-time. The user could not only specify the network layer at which the packets are collected, but also the types of packets (e.g. HELLO packets) that are collected at each node. A similar filter is available with the visualization tool to further aid the trace analysis.

IV. Hybrid Simulation

MiNT can be a crucial platform to test, debug, and evaluate protocol implementations before going for their larger-scale deployment. Simulations, however, will still provide an important first step in any protocol development and evaluation. MiNT provides a way to conduct the same simulations in realistic settings. In this section, we discuss the *hybrid simulation* technique that MiNT implements. We focus our discussion to one specific simulator, namely *ns-2*, which we modified to support hybrid simulations on MiNT.

A. Overview

Raising doubts about the veracity of simulation results is not uncommon [4], [5]. The drawback is mostly attributed to the lack of detailed models for the physical layer properties such as signal propagation and error characteristics. A common practice in most academic research to date is to use simplistic physical layer models. This is one of the prime reasons for the lack of simulation fidelity. With growing interest in cross-layer designs of protocols, it becomes imperative to provide accurate results at different layers in the protocol stack. Hybrid simulation alleviates some of these problems faced by pure simulation.

We define *hybrid simulation* as a technique where some layers of the simulator's protocol stack are replaced with their real implementations. It is well-known that majority of the inaccuracies in simulations stem from incomplete physical layer models. In our design, we replace the link layer, the MAC layer, and physical layer of the simulator with wireless card driver, firmware, and real wireless channel respectively.

The benefit of the hybrid simulation approach is that it requires minimal change to the already existing simulation code and scripts. The same simulation experiment can be used to obtain results in a realistic setting. The questionable effects of the physical layer models in simulation are removed through use of real wireless channel.

B. Implementation Issues

In this section, we discuss the challenges involved in implementing hybrid simulation capability into a standard discrete-event simulator, and and detail the techniques we use to overcome these challenges for the *ns-2* simulator.

Event Scheduler: Two key design components in a simulator are – (a) the way to model execution logic of different entities based on either events, activities or processes, and (b) the way the simulation time is advanced. *ns-2* is a discrete-event simulator, where the execution logic is based on events, and the time is advanced at the pace of event execution time using a global virtual clock. In a hybrid simulation, all

packet communication is carried over real wireless medium. This leads to inconsistency between the *virtual clock* that determines the dispatch rate of simulation events, and the *real-world clock* that determines the transmission rate of packets over actual wireless channel.

To overcome such issues, we use system clock on all the nodes, that are synchronized at the beginning of each experiment, to update the simulator's virtual clock. Events are now dispatched according to their real execution time instead of being executed as soon as the previous event has finished execution. We use *ns-2*'s built-in RealTime Scheduler with the following modification. *ns-2*'s RealTime Scheduler yields execution control to the kernel while waiting for the next event timer to expire. Most operating systems however only implement a coarse *process-level* scheduling granularity (10 ms). Because of this limitation, the control comes back to the *ns-2*'s scheduler only after 10 ms. In order to schedule events at a finer granularity, we implement a busy wait solution, and can process each event as soon as its timer expires.

Limiting the Number of Events: The correctness of hybrid simulation requires that events should not be scheduled " in the past". For instance, if the amount of time spent in processing the simulator's execution logic is too large, then an event dispatching a packet to another node could be delayed and may be dequeued by the scheduler when the real time has advanced past its scheduled execution time. We prevent such delayed event execution by reducing the number of events that the scheduler needs to process.

In our implementation, we have made a simplifying assumption that only one virtual node is mapped to a physical node. However, since we execute unmodified *ns-2* script on each physical node, it instantiates all the virtual nodes, including their traffic sources/sinks, on each physical node. Since we are binding only one virtual node to a physical node, therefore we prevent traffic sources on any other virtual node from generating any packet on this physical node. In our implementation of hybrid simulation, we identify the virtual node that is mapped to the physical node, and only allow traffic generators, like FTP and CBR, associated to this virtual node to schedule events.

Transmission/Reception of Packets: The internal packet format used in a simulator does not conform to the exact specifications of the real protocols. Hence, a packet from the simulator needs to be modified before it can be sent over the wireless medium.

Current *ns-2* implementation does not contain the protocol header fields needed for transmission over the wireless channel. In order to transmit an *ns-2* packet sent from the routing layer onto the link layer, we implement a wrapper that encapsulates the *ns-2* packets in a UDP packet payload, and delivers it to the destination node using standard socket layer. The address of the virtual node in the *ns-2* packet is mapped to a core node's IP address to which the packet is destined. Upon receiving the UDP packet carrying the *ns-2* payload, the receiver node decapsulates the packet and inserts it into the local event queue. The logic for distributed execution of hybrid simulation over wireless channel is shown in Fig 4.

Changes to ns-2 script: Our goal is to require minimal changes to the existing *ns-2* scripts to execute them on

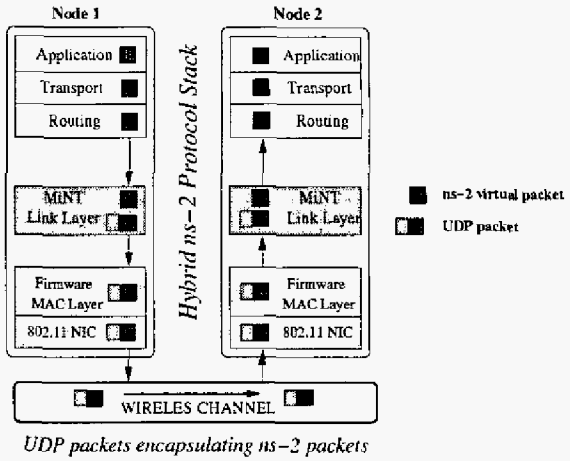

Fig. 4. The diagram shows the passage of a packet from one simulated node to another in hybrid simulation on MiNT. All event packets for other nodes generated in *ns-2* are encapsulated in a UDP packet payload and given to the wireless card for actual transmission. The receiving node decapsulates the packet and inserts the event into the local event queue.

the hybrid simulation platform. To provide a single-script abstraction, we kept the required changes independent of the individual core nodes. All changes are composed at the central distribution node, and *same* script is loaded on all the core nodes.

The changes to an existing script are: (i) the script must point to the MiNT link layer implementation instead of the *ns-2* link layer, (ii) each testbed node is assigned a physical node-id that is used in the *ns-2* script. The physical node-id for each node is preassigned and the *ns-2* script reads it from an environment variable local to each node.

Limitation: In our current design, only one virtual node is mapped onto a physical node. This might limit the the size of the network that can be tested in hybrid simulation by the number of physical nodes available. Careful observation reveals that it could be fundamentally impossible to share a physical node for multiple virtual nodes. This is because if each of the virtual nodes sharing a physical node is sending enough traffic to saturate the channel, then multiplexing the wireless card would be impossible using real clock. Also, it would be impossible to capture real MAC-level interaction, or effect of transmission over real wireless medium, for the virtual nodes that are mapped to the same physical node. For instance, assume a string topology of 3 nodes, where the first and the last node are out of each other's sense range. There are two flows, one between N1 and N2 (flow-1) and other between N3 and N2 (flow-2), active at the same time. Given two physical nodes, if N1 and N2 are mapped onto the same physical node, we fail to capture effects of real wireless medium on flow-1's packet transmissions; whereas, if N1 and N3 are mapped to the same physical node, then it would not be possible to capture the MAC layer interaction between N1 and N3.

V. Fidelity of MiNT

In this section, we prove that the miniaturization technique based on attenuator does not affect the fidelity of the experimental results. We first compare results of experiments conducted on the testbed with and without the use of attenuators

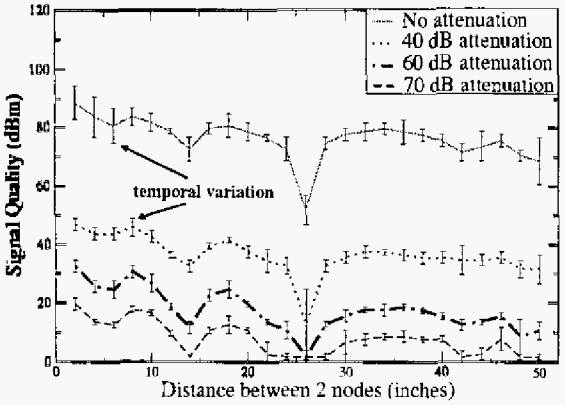

Fig. 5. Graph showing variation of signal quality at different overall attenuation on a link. It also shows the extent of temporal variation of the signal quality at each sample point. The signal quality varies non-monotonically over distance because of multipath fading. The variation of signal quality for the attenuated and the non-attenuated case follows the same pattern.

on the signal path. We present this comparison that verifies that miniaturization technique does not alter the behavior of any layer in the network stack; it only shrinks the physical space used by the testbed. Next, we discuss the limitations of MiNT.

Physical Layer: Signal propagation is a key aspect of the wireless physical layer. We study the impact of attenuation on signal propagation characteristics in MiNT. In this experiment, we use 2 nodes connected in ad hoc mode and apply different levels of attenuation. We compare the resulting spatial distribution of signal quality (SNR) with that of the non-attenuated case. Fig 5 shows the variation of signal quality reported by the card firmware, when signal attenuation on the path is varied from 40 dBm to 70 dBm. The signal quality is measured at 2 inches granularity. The same graph also shows the extent of time variation of signal quality at each sample point.

The figure shows that the signal quality variation is non-monotonic. There are intermediate regions where the signal is weaker relative to the neighboring regions, or even fades completely. These regions of weak signal quality, termed *dark spots*, are primarily a result of multipath fading. When the attenuation is removed completely, the signal quality improves, but the nature of its variation is preserved. The IEEE 802.11-1999 standards [21] also show similar non-monotonic distribution of signal quality. Furthermore, signal quality at any point for the attenuated and the non-attenuated cases show similar temporal variations.

Fig 5 also indicates how to configure a topology in MiNT. For example, when 70 dB of attenuation is applied, within a radius of 4ft (48 in) there are regions of good connectivity (16 dBm) and complete disconnectivity (2 dBm). Reducing signal attenuation and keeping the space unchanged makes the entire space better connected. By adjusting attenuation level to a specific research task's needs, one can trade off the minimum signal quality with the physical space requirement of the set-up. As the maximum communication range of a node at 70 dB attenuation is 4 ft, it should be possible to set up a multihop 16-node mesh network in a 12 ft x 12 ft space.

MAC Layer: In this experiment, we study the impact of

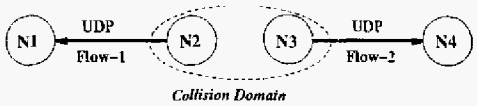

Fig. 6. String topology where the two sender nodes are in the same collision domain and contend for access to the shared wireless channel.

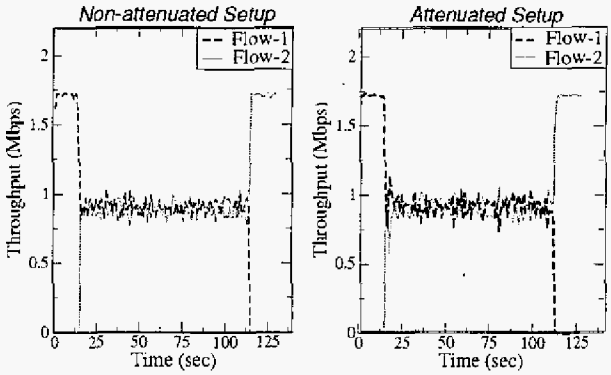

Fig. 7. This graph shows bandwidth sharing between two unicast flows when the senders are in the same collision domain, as shown in Fig 6, for an attenuated and non-attenuated set-up. The link quality between the two contending nodes is kept same across both set-ups. The channel is shared equally in both cases proving that the MAC layer is unaffected by introduction of attenuation.

attenuation on fairness property of channel access algorithm. We set up a string topology of 4nodes, as shown in Fig 6. Node $N2$ is sending unicast traffic to node $N1$, and node $N3$ to node $N4$. Since $N2$ and $N3$ are in the interference range of each other, they contend for access to the shared wireless medium. We compared two different set-ups – one with attenuators and the other without attenuators – while keeping the link quality same across both set-ups.

Fig 7 shows the instantaneous throughput of the two UDP flows for both the cases. As soon as the second flow starts, the channel is shared equally between the two contending flows. The bandwidth sharing behavior is same in the attenuated and the non-attenuated case.

Routing Layer: In this experiment, we show that the behavior of the routing layer protocols is not affected by introducing attenuators on the signal path. We use a 4-node network topology, where the end nodes are connected over 2 hops, as shown in Fig 8. In this experiment, we use AODV-UU [22] protocol to route packets between $N1$ and $N4$. The link quality is maintained same across the attenuated and the non-attenuated runs.

In each experiment, the route between node $N1$ and node $N4$ (chosen by AODV-UU) is made to fail by artificially failing the intermediate hop. Fig 9 depicts the time taken

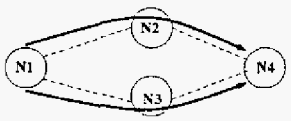

Fig. 8. The 2-hop topology used to run the AODV protocol experiment. The same topology was replicated with and without attenuation on MiNT keeping the link quality same.

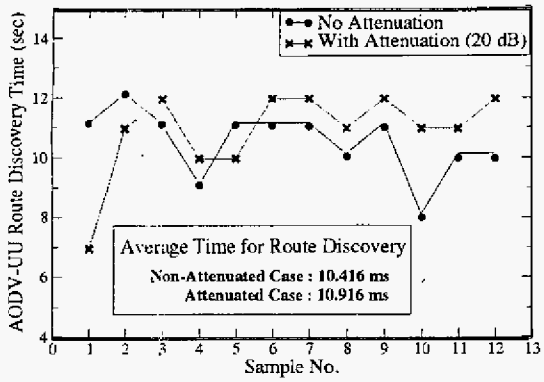

Fig. 9. This graph shows the comparison of AODV-UU [22] Route Discovery time in attenuated and non-attenuated set-up using the topology shown in Fig 8. The route discovery time varies between 7 ms to 12 ms, and the average time for attenuated and non-attenuated cases is 10.916 ms and 10.416 ms respectively.

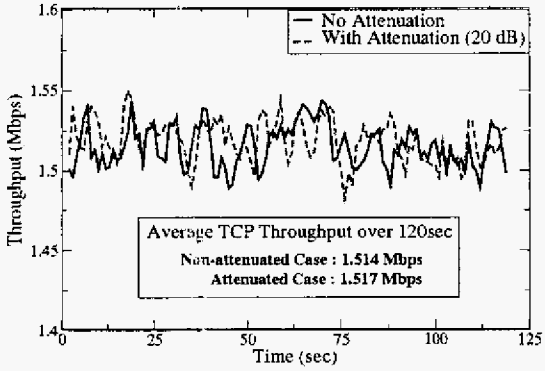

Fig. 10. This graph shows the throughput of a 1-hop TCP flow. The first set-up does not use any attenuator, while the second one uses a 20dB attenuator. The link quality is kept same in both the experiments.

for new route discovery when such a failure occurs. The time taken in both attenuated and non-attenuated cases varies between 7 ms to 12 ms, and the average over 12 samples is 10.416 ms and 10.916 ms for the non-attenuated and the attenuated case respectively.

Transport Layer: To prove that the transport layer is unaffected by attenuators, we use a 1-hop TCP experiment. We use 2 nodes connected in ad hoc mode and measure the throughput of a TCP connection between them. The link quality is again maintained same across the attenuated and the non-attenuated set-up.

Fig 10 shows the TCP throughput over 120 sec, averaged over 3 sec periods. The long-term average for the TCP flows are 1.514 Mbps and 1.517 Mbps for the non-attenuated and the attenuated case respectively. Even the instantaneous variations are similar in nature, suggesting that the transport layer behavior is not affected by use of attenuation.

Limitations: The key feature of a MiNT testbed is its ability to limit the signal propagation range between two nodes to within a few feet through use of attenuators. However, the attenuation approach has certain limitations. In this section, we discuss the pitfalls of MiNT and explain their impact on the final outcome of experiments.

Selective Attenuation: The most prominent change in MiNT

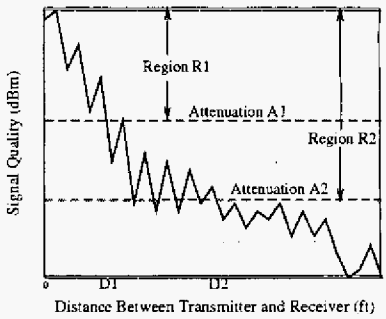

Fig. 11. Representation of relationship between signal quality and distance. Adding radio signal attenuators pushes the X-axis of the graph up, effectively reducing the extent of signal quality variation. At attenuation A1, the graph is confined to region R1. In region R1, the signal becomes 0 when the distance between transmitter and receiver is greater than D1.

from a typical full-scale testbed is that in MiNT the radio signals are attenuated at the transmitter and the receiver ends. As we are not placing the core nodes in a noise-free environment, the nodes operate in presence of external noise sources, like microwave oven, cordless phones, and other interfering channels. The RF signals from these noise sources are attenuated only at the receivers. Additionally the thermal noise at the receiver is unattenuated because it does not go through the receiver antenna. Since the attenuation of signal is more than that of the noise, one might suspect that the signal-to-noise ratio (SNR) for a link in MiNT is lower than that of an unattenuated testbed. However, this effect can be overcome by reducing either the attenuation level or the distance between the nodes.

Near-field Effect: In MiNT, since the nodes (and hence the antennas) are placed in proximity of each other, *the receiver is in the near-field zone of the sender.* This is unlike a full-scale testbed, where the nodes are typically placed far from each other, hence the receiver is usually in the far-field zone of the sender. This difference is inherent to the MiNT approach due to shrinking of the space.

Spatial Variation of Signals: Multipath effects in signal propagation lead to small-scale variation in the signal strength. A qualitative representation of this variation of signal quality with distance is shown in Fig 11. Between two points, say 0 and D2 there are multiple crests and troughs in the signal quality. By adding the attenuator, we are effectively pushing up the X-axis in this graph by the dBm value of attenuation. As a result of this, the number of crests and troughs between the same two points, 0 and D2, is *smaller* than that of the non-attenuated case. Constructive and destructive interference resulting from the multipath effects are dependent only on the frequency of the signals. Hence a solution to this problem is to scale down the frequency of the signals which would make the number of crests and troughs same. However, changing the frequency would change the properties of the wireless medium under test, and hence is not a viable solution.

This limitation impacts the mobility-related experiments where the extent of signal quality variation encountered by a mobile node in MiNT will differ from that of full-scale testbed.

Non-repeatability: Finally like any other testbed, experiments on MiNT are not exactly repeatable because the external factors affecting signal propagation cannot be fully controlled

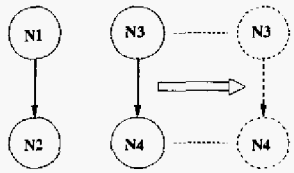

Fig. 12. Topology used for understanding the impact of signal propagation characteristics on channel access pattern determined by the MAC layer. Node pair $N1$-$N2$ is kept fixed at one position, while node pair $N3$-$N4$ is moved away from $N1$-$N2$.

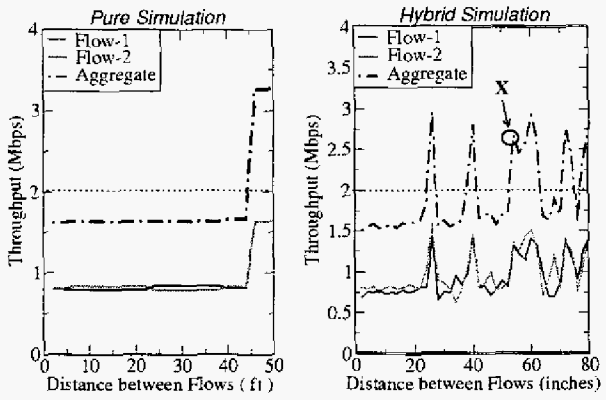

Fig. 13. This graph shows the difference in experimental results obtained from a similar set-up in two different environments – pure simulation and hybrid simulation on MiNT. It shows the impact of signal propagation characteristics on the behavior of the MAC layer. The graph shows the throughput variations of two unicast flows, shown in Fig 12, as they are moved away from each other. Use of two-ray ground propagation model in pure simulation leads to the MAC layers of the senders perceive the other sender's transmission till they are out of "sense range". In hybrid simulation, the signal quality variation is non-uniform, and the senders move in and out of sense-threshold, and hence the non-uniform throughput variation in hybrid simulation.

across experiments.

VI. Hybrid Simulation vs. Pure Simulation

This subsection presents the results of a comparative study between software-only *ns-2* simulations and hybrid *ns-2* simulation executed on MiNT. The main difference between pure *ns-2* simulation and hybrid simulation is that the latter replaces the simulated link, MAC, and physical layers with real implementations and real wireless channel. We study the impact of physical layer characteristics, *viz.* signal propagation and error characteristics, on data transfer rates for both the platforms.

A. Signal Propagation

In this experiment we demonstrate the impact of signal propagation on experimental results in pure simulation and hybrid simulation. We use 2 unicast flows, between nodes $N1$-$N2$ and $N3$-$N4$, as shown in Fig 12. The MAC layer on the senders $N1$ and $N3$ senses the channel before transmitting. Channel is perceived busy if signal from one active sender, say $N1$, reaches the other sender, say $N3$. If $N1$ cannot sense $N3$ then the two flows will be active simultaneously, giving higher throughput to both flows.

In our experimental set-up, we replicated the same topology in *ns-2* and MiNT. In *ns-2*, we use the two-ray ground propagation model, with a ratio of 1:2 for hearing range and

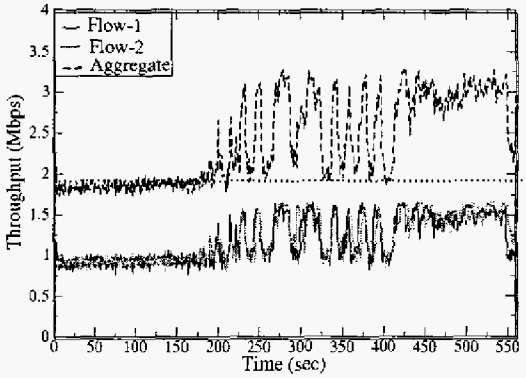

Fig. 14. The graph captures the impact of temporal variation of signal strength on MAC layer interaction between 2 nodes at a point (X in Fig 13). We use the set-up shown in Fig 12. Initially the senders can sense each other, hence the two flows are not active simultaneously. After a while, the signal quality drops, and the senders can no longer sense each other, resulting in two flows being active at the same time.

sense range (22ft : 44ft). In MiNT, the signal propagation is dependent on the environment, and this determines whether one node can hear/sense another node's activity. In *ns-2* the channel capacity is set to 2 Mbps. In MiNT, we set the card's transmission rate to 2 Mbps. For both cases we use a CBR traffic source on $N1$ and $N3$ to pump packets of size 1000 bytes at 2 Mbps, that ensures that both senders are constantly trying to access the channel.

Fig 13 shows the throughput of each flow as well as their aggregate in pure simulation using *ns-2*, and hybrid simulation using *ns-2* on MiNT. In pure simulation, till the point the two senders are within the sense distance (44 ft), the flows are constantly interfering. Therefore, the throughput of each flow is around 0.75 Mbps, giving an aggregate throughput around 1.5 Mbps. As soon as the senders move out of sense range, the flows stop interfering and the aggregate throughput shoots up to 3.2 Mbps. Unlike in pure *ns-2* simulations, where the throughput of each flow stays *uniform* at 0.75 Mbps till the distance exceeds the sense range, in hybrid simulation, there are distinct variations in throughput, especially at 26in and 40in distances, where the senders cannot sense each other.

The non-uniform distribution of throughput in hybrid simulation is explained with reference to the signal quality graph for 70 dB attenuation, shown in Fig 5. When the signal quality drops due to the presence of a " dark spot", the two senders fail to sense each others' transmissions. Therefore, the two flows can send packets at the same time.

We also observe that with increasing distance the number of spikes in throughput increases. At shorter distance, even if the senders fall in dark spots of each other and cannot communicate, they can still sense each other. However, with increasing distance, the dark spots completely isolate the two senders.

Additionally, there are points where the temporal variation of the signal quality is large. In Fig 13, we have marked one point X at distance 58in, where the aggregate throughput is less than the peak value. This is explained using Fig 14, that captures the temporal variation of flow throughput at a point using interaction between the two flows. The interference is initially higher leading to channel contention between the

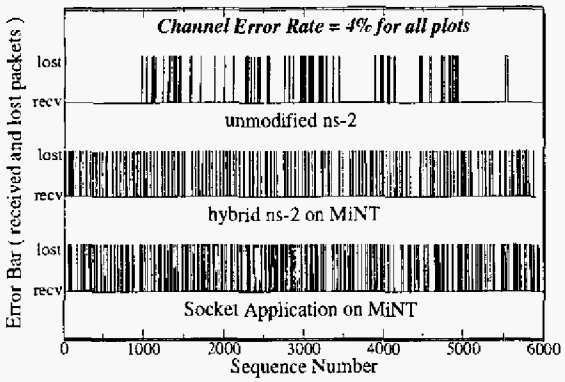

Fig. 15. The graph plots the packet errors encountered, represented as lost sequence numbers at the receiver, when a 1 Mbps CBR traffic source sends 1000 bytes data packets between 2 nodes for 1 minute duration in a software-only *ns*-2 simulation, and in a hybrid *ns*-2 simulation run on MiNT. It also shows the packet error rate for the same two nodes when data is sent using socket applications over UDP in real implementation. The packet error rate is fixed to facilitate comparison.

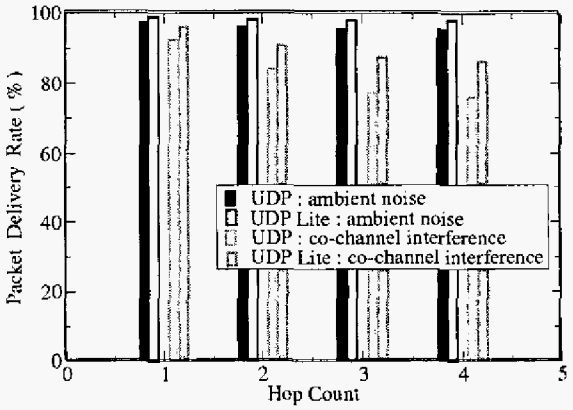

Fig. 16. The graph shows that UDP Lite can improve the packet delivery rate in lossy wireless channels. Moreover, the gain increases with the number of hops and with increasing co-channel interference. Video data like MPEG, which goes through (de)compression algorithm's built-in error correction mechanism, can improve their end-to-end throughput by accepting corrupted packets. This UDP Lite protocol implementation runs on a 4-hop set-up on MiNT.

senders, but later the interference fades, and both the flows can pump data simultaneously.

Pure simulation fails to capture this non-uniform spatial and temporal variation of throughput, which is an artifact of signal propagation characteristics.

B. Error Characteristics

In this experiment, we show the difference in error characteristics captured using pure *ns*-2 simulation and hybrid simulation running on MiNT. We use a CBR traffic source to pump data from one node to another. In pure simulation, we choose an error model that is most commonly used in *ns*-2-based simulation studies, where each packet is corrupted based on a uniform random variable and pre-specified error probability. On the other hand, errors in hybrid simulation occur due to the ambient noise in the environment.

Fig 15 plots successful and unsuccessful packet transmission in simulation, hybrid simulation, and real-world communication. The results show that simple bit error models in simulation could produce qualitatively different behavior than those observed in real radio channels as seen on MiNT. Therefore, testing wireless protocols that depend on accurate bit error characteristics becomes much easier and produces realistic results with the use of hybrid simulation technique.

VII. UDP Lite Protocol Evaluation: A Case Study

We used MiNT to study the performance of UDP Lite protocol [10] on 802.11-based multi-hop wireless network. This exercise demonstrates the usefulness of MiNT in validating protocol implementations.

A. UDP Lite Protocol Description

To guard against bit errors, checksumming is used to verify the integrity of received bits at every layer in the protocol stack. When the checksum for a received packet fails, the packet is dropped. The UDP Lite protocol advocates that partially corrupted packets are still usable, especially for streaming media data. This protocol is particularly useful for wireless streaming applications as wireless channels tend to have relatively low bandwidth and high bit-error rate. UDP Lite allows application developers to specify a range of sensitive bytes in each packet. As long as the sensitive bytes of a received packet are correct, the entire packet is considered usable and not dropped. In essence, UDP Lite is a UDP-like protocol that applies checksum only to a specified part of each UDP packet.

B. UDP Lite Implementation

Implementation of the UDP Lite protocol on a multi-hop wireless network would require an intermediate node to be able to (1) receive a corrupt packet, and (2) relay it to the next hop. In the normal mode of operation, commercial IEEE 802.11 wireless cards simply drop corrupted packets as soon as the checksum fails at the firmware layer. To faithfully emulate the behavior of the UDP Lite protocol, we use two wireless NICs on each node – one operating in *RF monitor mode* and serving as a dedicated receiver, and the other in normal mode and serving as a dedicated sender. With this hardware set-up along with few device driver changes, the receiver card receives all packets, corrupt or not, and delivers them to a user-level application. The user-level application implements the UDP Lite protocol and selectively forwards the received packets through the sender card. The application uses the packet capture library, libpcap, to retrieve packets captured by the RF monitor card, then computes the checksum for the sensitive bytes, and forwards packets whose sensitive bytes are not corrupted. We use a separate NIC for transmitting because the card used in MiNT cannot transmit any packet when operating in RF monitor mode. With latest cards it could be possible to sniff the traffic in RF monitor mode, and at the same time transmit packets. Finally, to disable link-layer acknowledgement and retransmission mechanisms, we use broadcast instead of unicast primitive for transmitting packets in all UDP Lite related experiments.

C. Performance Evaluation of UDP Lite on MiNT

As UDP Lite is more tolerant to bit errors than base UDP, it can give better performance. We measure the improvement in the packet delivery probability at each hop in a 4-hop wireless

network. The first hop is relatively more noisy compared to the other hops. In the first round, the measurements are taken without any external noise source; so the packet corruption is mainly due to the ambient noise. In the second round, we introduce co-channel interference by generating traffic in an adjacent radio channel. The results are shown in Fig 16. The performance gain of UDP Lite over UDP increases as the channel noise increases. The gain also increases with the number of hops between the source and destination as bit errors accumulate across hops.

This case study shows that MiNT could be used for evaluating wireless protocols in real-setting. MiNT provided the flexibility to capture corrupted packets and also made it easy to set up a 4-hop network topology required for this study.

VIII. Related Works

In this paper, we describe the design of a miniaturized wireless network testbed, and its use in application testing and hybrid simulations. In this section, we look at the contributions of other researchers in building wireless network testbeds.

We first study some of the full-scale testbeds that are tailored to satisfy specific project's requirements. The *CMU testbed* was built for evaluating the Dynamic Source Routing (DSR) protocol for ad hoc wireless networks [7]. The testbed comprised of 5 mobile nodes and 2 static nodes spread over an area of 300m by 700m. The mobile nodes were implemented with rented cars carrying laptops acting as mobile ad hoc nodes. Since this testbed was meant for a specific application, it did not address issues of flexible experiment control and management. The *Ad Hoc Protocol Evaluation Testbed* (APE) is used for comparative study of different ad hoc routing protocols [8]. Node placement is done manually using a large physical space. Mobility of the nodes is explicitly managed by choreographing movement of volunteers carrying laptops. We believe that this technique of topology generation is not very flexible. The *Roofnet* project at MIT has built a 50-node testbed spread across rooftops of volunteers in Cambridge [9]. Roofnet is used for studying behavior of wireless mesh networks, and is not aimed at providing configuration flexibilities to user. Another testbed is being designed at Rice University to implement and evaluate *Transit Access Point* architecture [23]. The testbed uses custom-designed hardware, and may not be easy to replicate because of high cost.

Keeping in tune with the growing needs of the wireless research community for testbeds, some researchers are designing large-scale open platforms. Netbed is a shared wired network emulation platform [24], with a recent proposal to build its *wireless extension* [25]. The testbed plans to use a dense mesh of wireless nodes across the department building. Topology reconfiguration is achieved by selectively turning the wireless nodes on and off. The mobility of the nodes is captured by handhelds carried by student volunteers. Another shared testbed under development is *Open Access Research Testbed for Next-Generation Wireless Networks (ORBIT)* [26]. This testbed uses an indoor grid of 400 nodes in a 20m by 20m space. Unlike MiNT, ORBIT testbed nodes are custom-designed. The mobility of the nodes is simulated through a mobility server that activates different nodes at different times to represent the same emulated node. In MiNT, the mobility is introduced through use of robots carrying either the antennas or the nodes themselves. In a similar vein, the *WHYNET* [27] project is building a shared wireless testbed for mobile wireless technologies. WHYNET project plans to incorporate comprehensive hybrid simulation facilities in its testbed.

More akin to our work are the efforts that build testbeds that overcome large space requirement through miniaturization. Kaba and Raichle at *Sarnoff* designed a testbed on a desktop by restricting and controlling radio ranges and propagation effects of the PC cards [28]. This testbed demonstrates the idea of using fixed radio signal attenuators to reduce the radio range of wireless cards. However, they use RF cables to connect the communicating node pairs, thus shielding external interference. *RAMON* is another set-up using programmable attenuators for testing rapid mobility scenarios. The signal quality as seen by a mobile host is altered using the programmable attenuators. Sanghani *et al.* built an *Emulated Wireless Ad Hoc Network Testbed (EWANT)* with the goal of providing low-cost environment for wireless research [29]. Similar to the Sarnoff testbed, they also use attenuators and shielding to shrink the radio ranges. The mobility of the nodes is emulated by connecting 1 PC card to 4 external antennas through 1:4 RF demultiplexer, and switching the transmission through these antennas. Although these works demonstrate use of radio attenuators for wireless experiments in a limited space, MiNT does a comprehensive evaluation of this approach, and provides full support for experiment control (including node mobility) as well as post-experiment analysis. In addition, MiNT supports hybrid simulation using *ns-2*.

The usefulness of seamlessly migrating from simulation environment to field testing using actual implementation has been noted earlier. *ns-2* itself provides a network emulation facility where real applications can interact with simulated ones [30]. The nsclick project [31] attempts to bridge the gap between simulation and deployment by presenting a set-up where the code written for *ns-2* simulation can be used with minimal change in real implementation. We do not set reuse of code as our goal. Instead, we want to complement *ns-2* by giving it a stronger validation platform. We achieve this by enabling unmodified *ns-2* scripts to be executed, on the testbed nodes with MAC and physical layer functionalities from the real-world set-up.

A key advantage of simulation approach is repeatability, while trading off realism. Judd and Steenkiste perform *digital emulation* of signal propagation using an FPGA-based emulation platform [32]. They use coaxial RF cables to feed the signal from an RF device to the emulator. The emulator controls the emulation of signal propagation by taking into account the impact of external factors, like multipath interference, through use of signal propagation models. The main drawback of this approach is that external factors are still modeled and are not truly real. Like other testbeds, MiNT also does not aim to provide repeatability as a feature.

IX. Summary and Future Work

Network researchers have a long tradition of using simulation tools in their study. In wireless network research, network simulator, *ns-2*, has been one of the most widely

used tools for validation and evaluation of protocols. However, a simulation tool is only as accurate as its models for the protocols and systems under study, such as MAC-layer interactions, signal propagation characteristics, and channel error patterns. Improving a model's accuracy proportionally increases its complexity, and eventually the total time required for individual simulation runs. As a result, researchers often can only afford simplified models when simulating complex behaviors, e.g. the radio propagation models used in *ns-2* simulator. Nonetheless, simulation tools are still very useful in providing a controlled environment for the initial design and tuning of wireless protocols with multiple parameters. In order to complement simulations, researchers have also built testbeds that help better understand and demonstrate the operational capabilities of specific wireless protocols. Unfortunately, a full-scale wireless testbed is expensive and time-consuming to build and maintain. Management of such a testbed is difficult because of the large physical area required to come up with any interesting multi-hop topology. It is also not easy to reconfigure such a testbed or replicate it for other projects.

MiNT is designed to address the shortcomings of existing wireless testbeds. Through use of radio signal attenuation, MiNT is able to miniaturize a multi-hop wireless network testbed to small physical space, e.g., an 8-node MiNT testbed can be set up within a 12ft by 6ft space. Because large physical space is no longer necessary, a MiNT testbed is easier to set up, re-configure, and administer. From the user's standpoint, MiNT provides a GUI to configure and monitor the testbed remotely through a centralized controller node. It also can be used as a platform for wireless application/system development, as well as a hybrid testbed that supports *ns-2* simulation. In the hybrid simulation mode, a MiNT testbed executes unmodified *ns-2* scripts, and generates more accurate results as it replaces the link layer and physical layer models in *ns-2* with real WLAN interfaces and the wireless medium. Through a working MiNT prototype, we demonstrate the feasibility of MiNT as a multi-hop wireless testbed by showing its fidelity when compared with a non-miniaturized testbed, and the results of a case study. We also demonstrate through examples the applicability of MiNT for hybrid *ns-2* simulations.

The current MiNT prototype only supports limited node mobility due to use of desktop PCs as the platform for testbed nodes. We are currently building the second MiNT prototype that will support unrestricted mobility. We plan to use Soekris boards [13], which are small form-factor battery-operated programmable devices, as the new platform for testbed nodes. The small size and battery-based operation allow the testbed node itself to be mounted on a robot. The robot's battery could also power the Soekris board. The battery needs to be re-charged periodically. For 24x7 automated operations, re-charging should not require human intervention. We are exploring using Roomba [33] for our robot platform, because of its controllability (over infrared), low cost, high payload capacity, and auto-charging feature. We also plan to incorporate mobile RF obstacles as components of the testbed. This could be in the form of water packs mounted on robots. This will add flexibility in configuring network topologies.

ACKNOWLEDGMENT

We would like to acknowledge technical assistance from Brian Tria and the people at HD Communications (http://www.hdcom.com/). We would also like to thank Prof. Hari Balakrishnan for providing useful insights that helped enhance MiNT design. This research is supported by NSF awards ACI-0234281, CCF-0342556, SCI-0401777, CNS-0410694 and CNS-0435373 as well as fundings from Computer Associates Inc., New York State Center of Advanced Technology in Sensors, NIST, Siemens, and Rether Networks Inc.

REFERENCES

[1] " The Network Simulator - ns-2." http://www.isi.edu/nsnam/ns/
[2] " The CMU Monarch Project." www.monarch.cs.cmu.edu/cmu-ns.html
[3] X. Zeng, R. Bagrodia, and M. Gerla, " GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks," in *Workshop on Parallel and Distributed Simulation*, May 1998.
[4] J. Heidemann, N. Bulusu, and J. Elson, " Effects of Detail in Wireless Network Simulation," in *Proc. of the SCS Multiconference on Distributed Simulation*, Jan 2001.
[5] M. Takai, J. Martin, and R. Bagrodia, " Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks," in *Proc. of MobiHoc*, Oct 2001.
[6] Y. Hu and D. Johnson, " Exploiting MAC Layer Information in Higher Layer Protocols in Multihop Wireless Ad Hoc Networks," in *Proc. of ICDCS*, Mar 2004.
[7] D. Maltz, J. Broch, and D. Johnson, " Experiences Designing and Building a Multi-Hop Wireless Ad-Hoc Network Testbed," in *CMU TR99-116*, 1999.
[8] H. Lunndgren, D. Lundberg, J. Nielsen, E. Nordstrom, and C. Tscudin, " A Large-scale Testbed for Reproducible Ad Hoc Protocol Evaluations," in *Proc. of WCNC*, 2002.
[9] B. A. Chambers, " The Grid Roofnet: A Rooftop Ad Hoc Wireless Network," MIT Master's Thesis, Tech. Rep., Jun 2002.
[10] L.Larzon and M. Degemark and S.Pink, " UDP Lite for Real-Time Multimedia Applications," in *Proc. of Sixth IEEE International Workshop on Mobile Multimedia Communications*, 1999.
[11] A. Raniwala and T. Chiueh, " Architecture and Algorithms for an IEEE 802.11-based Multi-channel Wireless Mesh Network," in *Proc. of IEEE Infocom*, 2005.
[12] " Programmable Mobile Robots." http://mindstorms.lego.com
[13] " Soekris Engineering." http://www.soekris.com
[14] " Simple Network Management Protocol." http://www.cisco.com/
[15] " A Wireless LAN API for the Linux Operating System." http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Linux.Wireless.Extensions.html
[16] D. Niculescu and B. Nath, " Ad hoc Positioning System (APS)," in *Proc. of GLOBECOM*, Nov 2001.
[17] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, " The Cricket Location-Support System," in *Proc. 6th ACM MOBICOM.*, Aug 2000.
[18] M. Hibler, L. Stoller, J. Lepreau, R. Ricci, and C. Barb, " Fast, Scalable Disk Imaging with Frisbee," in *Proc. of USENIX Annual Tech Conf.* 2003.
[19] P. De, A. Neogi, and T. Chiueh, " VirtualWire : A Fault Injection and Analysis Tool for Network Protocols," in *Proc. of ICDCS*, May 2003.
[20] J. Yeo, M. Youssef, and A. Agrawala, " A Framework for Wireless LAN Monitoring and Its Applications," in *Proc. of ACM Workshop on Wireless Security*, Oct 2004.
[21] " IEEE 802.11 Standards." http://standards.ieee.org/getieee802/802.11.html--Page28
[22] " Ad-hoc On-demand Distance Vector Routing - Uppsala University." http://user.it.uu.se/~henrikl/aodv/
[23] R. Karrer, A. Sabharwal, and E. Knightly, " Enabling Large-scale Wireless Broadband: The Case for TAPs," in *Proc. of HotNets*, 2003.
[24] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, " An Integrated Experimental Environment for Distributed Systems and Networks," in *Proc. of OSDI*, Dec 2002.
[25] B. White, J. Lepreau, and S. Guruprasad, " Lowering the Barrier to Wireless and Mobile Experimentation," in *Proc. of HotNets*, 2002.
[26] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, " Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols," in *Proc. of WCNC*, Mar 2005.
[27] " WHYNET: Scalable Testbed for Next Generation Mobile Wireless Networking Technologies." http://pcl.cs.ucla.edu/projects/whynet/
[28] J. Kaba and D. Raichle, " Testbed on a Desktop: Strategies and Techniques to Support Multi-hop MANET Routing Protocol Development," in *Proc. of MobiHoc*, 2001.
[29] S. Sanghani, T. X. Brown, S. Bhandare, and S. Doshi, " EWANT: The Emulated Wireless Ad Hoc Network Testbed," in *Proc. of WCNC*, 2003.
[30] K. Fall, " Network Emulation in the Vint/NS Simulator," in *Proc. of IEEE Symposium on Computers and Communications*, July 1999.
[31] M. Neufeld, A. Jain, and D. Grunwald, " Nsclick:: bridging network simulation and deployment," in *Proc. of ACM Intl Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2002.
[32] G. Judd and P. Steenkiste, " Repeatable and Realistic Wireless Experimentation through Physical Emulation," in *Proc. of HotNets*, 2003.
[33] " Roomba Discovery." http://www.irobot.com