

# IvyNet: A Testbed for Multi-Hop Wireless Network Research

Yang Su  
Computer Science  
Department  
ETH Zurich  
8092 Zurich Switzerland  
ysu@inf.ethz.ch

Markus Heule  
Computer Science  
Department  
ETH Zurich  
8092 Zurich Switzerland  
markus.heule@inf.ethz.ch

Thomas Gross  
Computer Science  
Department  
ETH Zurich  
8092 Zurich Switzerland  
thomas.gross@inf.ethz.ch

## ABSTRACT

This paper describes IvyNet, a miniaturised 802.11-based multi-hop wireless network testbed, that is currently being developed for controllable wireless testing and evaluation of wireless protocols and applications. IvyNet requires only a small space and allows users to set up a multi-hop wireless network with minimal effort. We present the key elements of the IvyNet testbed architecture, including its hardware and software platforms. We also demonstrate the fidelity of IvyNet by extensive experiments. Finally, using a case study we show the usefulness of IvyNet for wireless protocol evaluation.

## 1. INTRODUCTION

Wireless multi-hop networks have become very popular over the past several years. Most of the time, researchers rely on wireless simulators for evaluating new protocols/applications for wireless networks. Because simulators are often based upon idealised and simplified models for radio propagation and the physical environment that fail to capture the complex channel characteristics of a real system faithfully, the fidelity of the simulation results remain a concern [1], [13]. There is an increasing need in the research community for research testbeds that allow users to conveniently perform controllable experiments with protocols and applications using real-world wireless devices. Researchers have built several full-scale operational wireless network testbeds [12], [14]. Because the long transmission range of the wireless radios, a real multi-hop testbed requires a large space to operate, and the deployment and management of these testbed are expensive. Due to the cost and complexity of building wireless testbeds, most researchers still rely on simulators to conduct wireless network research in spite of the known inaccuracy of the simulation results.

As an effort to reduce the barrier for building wireless testbeds, we developed IvyNet, a miniaturised wireless multi-hop network testbed. IvyNet consists of a set of

wireless nodes built with off-the-shelf hardware. The wireless nodes communicate with each other using a wireless interface. By attenuating the radio signals on the transmitter and the receiver, IvyNet dramatically reduces the space requirement of a wireless testbed. With IvyNet, users can easily set up a multi-hop wireless network in a laboratory environment. In this paper, we present the design and implementation of the IvyNet prototype. We highlight the IvyNet software toolkit for managing and automating experiments. For any experimental platform, fidelity is always a concern. We verify the fidelity of the miniaturisation approach used in IvyNet through extensive experiments. We show that in the scaled-down version of networks, links preserve the similar distribution of packet level link quality as links in large scale networks.

The rest of the paper is organised as follows. Section 2 discusses related work. In Section 3, we describe the overall architecture of IvyNet. Section 4 contains the validation of the testbed. Section 5 presents the detailed design of the IvyNet software toolkit. At the end, we present a use case in Section 6.

## 2. RELATED WORK

There are a number of related testbed projects for wireless networks.

ORBIT [4] uses a combination of 802.11 and cellular telephone (3G) equipment. For testing 802.11 networks, it implements a laboratory-based wireless network emulator that uses a two-dimensional grid of 802.11 radio nodes which can be dynamically interconnected into specified topologies with reproducible wireless channel models. Kansei [2], [3] is a hybrid sensor network testbed that consists of heterogeneous testbeds with different wireless technologies. It also has an indoor grid of wireless nodes with 802.11 radio interfaces. In addition, it supports hybrid simulation over the physical testbed. IvyNet is very similar to ORBIT and Kansei, in the sense that it is also built on a collection of stationary

miniaturised 802.11 wireless nodes. IvyNet differs in the design of its various management software components. The MiNT testbed [5] is a miniaturised 802.11 testbed for mobile wireless networks. It reduces the area required to run a multi-hop 802.11 testbed, and integrates ns-2 simulation with emulation. They achieve mobility through the use of antennas mounted on mobile robots. In contrast to MiNT, in the current IvyNet implementation, all the nodes are stationary. IvyNet provides more flexibility to manage the node state. It stays compatible with ns-2 by adopting the ns-2 experiment description language for the experiment setup.

Researchers also built some full-scale testbeds. The CMU testbed [14] was built for evaluating the Dynamic Source Routing (DSR) protocol. The testbed is made up of 5 mobile nodes and 2 static nodes in an area of 300m by 700m. The mobile nodes were carried by cars moving around. The roofnet [12] project at MIT has built a 37-node testbed spread over four square kilometres of an urban area. Roofnet provides a platform for studying the behaviour of wireless mesh networks. Both testbeds were built for specific applications. They do not support flexible experiment control and management. Furthermore, they require large space, and they are expensive to operate.

### 3. SYSTEM DESIGN

In this section, we present the overall IvyNet architecture and the design of the individual testbed components.

#### 3.1 Overall Architecture

IvyNet consists of a set of experimental nodes, monitor nodes, and control servers. The experimental nodes and the monitor nodes are managed by the control servers. The experimental nodes communicate with each other using an IEEE 802.11 wireless NIC. To reduce the transmission range, the wireless NIC is connected to a low-gain antenna through RF attenuators. The monitor nodes are dedicated nodes that set their wireless interface into monitor mode. The monitor nodes sniff the wireless traffic and collect packet traces. The IvyNet control servers oversee the operations of all experimental nodes. The experimental nodes and the monitor nodes communicate with the control servers through a dedicated network interface that can be either a wired Ethernet or a wireless network; this dedicated control network does not interfere with the wireless transmission in the testbed. To support user experiments, the IvyNet toolkit has been developed to facilitate system management, experiment setup, and experiment analysis. It consists of management/control softwares as well as application software for the wireless nodes.

#### 3.2 Wireless Nodes

The collection of wireless nodes is the key component of the testbed. In our prototype implementation, we use Soekris 4826 as the wireless nodes. Soekris 4826 is a PC-compatible embedded low-cost and low-power communication computer. It has a Geode SC1100UFH-266 CPU, 64 MB on-board flash storage, and 128 MB RAM storage. For communication, each box has one fast Ethernet controller. There are 2 mini-PCI slots for extensions. There is also a Compaq ZFM Micro USB 1.1 controller on the board for external persistent storage. Each wireless node is equipped with one EnGenius NL-5354MP plus ARIES2 Mini PCI wireless network card. It uses the Atheros 5213 chipset and supports 802.11 a/b/g modes. A low-gain omni-directional antenna is connected through an RF attenuator to the wireless card. For any miniaturisation technique, the fidelity of the spatial scaling of the wireless network's behaviour is always a concern. With the experiments in Section 4, we show that it is possible to scale down the wireless radio transmission to fit the testbed into an laboratory setting, at the same time, maintaining high fidelity in terms of packet level link quality. We use 30dB fixed attenuators for reducing both the transmission and the reception power, as a consequence, the maximal transmission range of the wireless interface amounts to less than 5 meters. For ease of management, we keep all nodes stationary. Because the wireless NIC supports multiple txpower settings, even with the same set of stationary wireless nodes we can still get different connectivity characteristics by using different txpower levels.

The wireless nodes run Linux. As part of the IvyNet toolkit, a node agent runs on wireless nodes, listening to commands from the control server. The node agent can start and stop applications, dynamically pass parameters to the applications, and report the experiment status to the control server.

#### 3.3 Control Servers

Control servers provide system services for system management and experiment management. The IvyNet toolkit implements multiple control servers: an experiment control server, a trace collection server, and a node management server. The experiment control server is responsible for interpreting a user's experiment descriptions, translating them to experiment scripts, and disseminating the experiment setup to wireless nodes. It also schedules multiple experiments to enable the sharing of the wireless testbed. During the execution of an experiment, it keeps track of the state of the experiment execution by processing the events reported by the node agent. The trace collection server collects traces from the monitor and wireless nodes during the experiment execution and stores them for post-processing. The node management server manages the system con-

figuration of the wireless node. It provides an interface to query and set system parameters. The node management server also manages the system images of the wireless nodes. It enables quick imaging of hard disks on the wireless nodes as required by the user. Control servers either reside on one host or are distributed over multiple hosts.

## 4. VALIDATION OF IVYNET

Like Mint [5] and Kansei [3], IvyNet uses RF attenuators to reduce both the transmission power and the reception power of the wireless interfaces. We target to shrink the wireless network into a space that is hundred times smaller than the full-scale deployment space. In this section, we present the validation results. They show that the miniaturisation technique based on attenuation can effectively reduce the wireless NIC’s communication range. At the same time, it is still possible to get high-fidelity experiment results with the testbeds. All the experiments shown in this section use only the IEEE 802.11b mode, and they are carried out in an office building where there is interference also from the existing infrastructure WLANs.

### 4.1 Effectiveness of Miniaturisation

In IEEE 802.11 wireless networks, several wireless hosts share one wireless channels under the control of the CSMA/CA MAC protocol. Only when two hosts are out of the carrier sense range of each other, they can transmit packets concurrently. In this experiment, two wireless nodes with 40dB attenuation broadcast packets with 800 Bytes payload as fast as they can. We record the broadcast throughput of two nodes when they broadcast at the same time. The experiments are carried out with different transmission powers. Figure 1 shows the measured broadcast throughput of the nodes when the distance between them is varied from 0.25 m to 5 m. We can observe that the carrier sense range of the two nodes is reduced to a very small distance. With 19dbm transmission power, the carrier sense range is 2.5 m. Two nodes can broadcast simultaneously with the full rate, when they are more than 2.5 m away. The figure also shows that reducing the transmission power effectively reduces the carrier sense range. When transmission power is reduced from 19 dbm to 0 dbm, the carrier sense range decreases from 2.5 m to 1.5 m. With the dramatically reduced carrier sense range, it is possible to set up a multi-hop wireless network in relatively small spaces.

### 4.2 Fidelity of Spatial Downscaling

The wireless links in the testbed should behave with reasonable fidelity compared to the wireless links in large scale networks so that the behaviour of the protocols and applications in one network can predict their

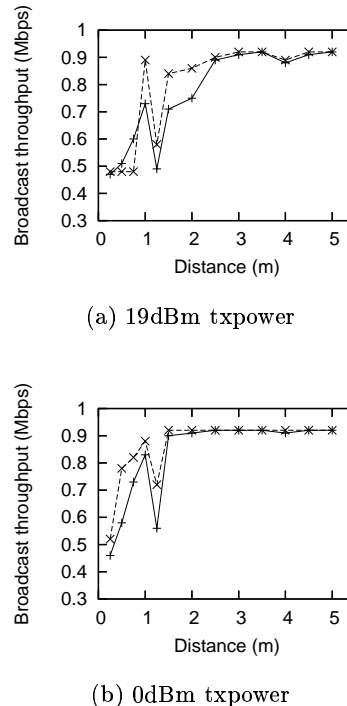
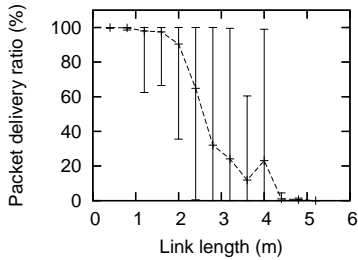


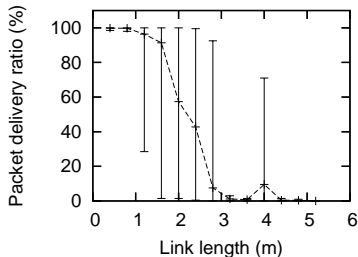
Figure 1: Carrier sense range

behaviour in another network. It is difficult to compare the physical signal quality from two networks, because the propagation of wireless radio is affected by many factors and the detailed behaviors of large scale wireless networks are very expensive and difficult to measure. In this paper, instead of trying to compare the detailed physical characteristics of two networks, we compare the link quality in terms of packet delivery ratio and one-hop TCP throughput. These parameters determine the performance of many important applications and protocols. We show, by experiments, that in IvyNet we can get a wide range of link qualities that cover all the possible link qualities in large-scale networks. In IvyNet, short links have good link quality with high probability, medium-length links experience medium link quality with a high variance, and with high probability, long links show poor quality. In all experiments in this section, 30dB fixed attenuators are used.

Figure 2, shows the average packet delivery ratio of links with different length. The data are collected from a network with 14 nodes placed in a chain. The nodes are put 0.4 m away from each other. For a node  $A$  and a node  $B$ , we count the link from  $A$  to  $B$  and the link from  $B$  to  $A$  as two different links. There are 182 links in the network. Each data point in the graph is the average of the measurements from multiple links with the same length. There are more short links than long links. For example, we have 26 0.4 m links while only



(a) 19dBm txpower

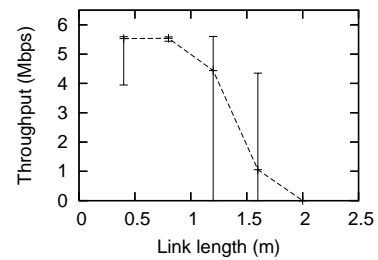


(b) 0dBm txpower

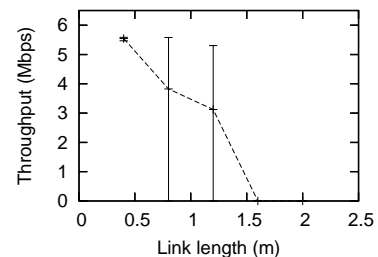
**Figure 2: Packet delivery ratio**

two 5.2 m links. To measure the packet delivery ratio, each node, in turn, broadcasts 200 1 KByte packets and the other nodes measure the number of the received packets. We can observe that with a link length of less than 1 m, we can always get over 99 % delivery ratio, no matter which transmit power level was used. When the transmit power is 19 dBm, links with length from 1 m to 4.2 m show a medium delivery ratio. When the transmit power is 0 dBm, a medium delivery ratio is only achieved over links shorter than 3 m. Medium areas show very high variance in packet delivery ratio. For links longer than 4.2 m with 19 dbm and links longer than 3 m with 0 dbm, link quality becomes very poor and most of the packets get lost.

Figure 3, presents how the link quality (expressed as one-hop TCP throughput) changes with different link lengths. The experiments are also carried out over a network with chain topology. We use a fixed bit rate of 11 Mbps. For each pair of nodes, we measure the TCP throughput for both directions. Each measurement was repeated 20 times. Each data point is the average of multiple measurements from multiple links with the same length. The figure indicates that TCP can get high throughput reliably over short links. When the link length increases, the average throughput drops down. The TCP connection does not work over links longer than 2 m for 19 dBm transmit power and 1.6 m for 0 dBm transmit power. Figure 4 compares the



(a) 19dBm txpower



(b) 0dBm txpower

**Figure 3: One-hop TCP throughput**

TCP throughput with different bit rate. The transmit power is set to 19dBm. We can observe that although 11 Mbps is the highest transmit bit rate supported by IEEE 802.11b that needs good link quality to operate efficiently, there is a large range (up to 1.5 m) where the link quality is good enough for TCP to achieve higher throughput with 11 Mbps than with 5.5 Mbps.

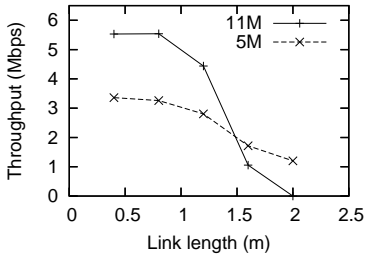
The experiments in this section show that with RF attenuators, we can effectively reduce the scale of a multi-hop wireless network into a much smaller area, which is more convenient for conducting experiments. At the same time, the links in the scaled-down version of networks preserve the similar distribution of packet level link qualities as links in the large scale networks. By adjusting the degree of attenuation, the length of the link, and the transmission power we can get all the possible link qualities that might be observed in large-scale networks.

## 5. KEY SERVICES AND SOFTWARE COMPONENTS

In this section, we present the IvyNet software toolkit which helps users to set up the testbed, manage testbed, and run the experiments efficiently over the testbed.

### 5.1 Carrier Sense Range and Link Quality Measurement

In IvyNet we integrate tools to measure the carrier



**Figure 4: One-hop TCP throughput with different bit rate**

sense range of the wireless nodes and the wireless link quality. The 802.11 protocol requires the sender to do carrier sensing on the radio channel, before transmitting a packet. If two nodes are within the carrier sense range of each other, only one of them will transmit at a time. Otherwise, they can spatially reuse the channel concurrently. Information about carrier sense range is essential for the multi-hop wireless testbed setup and the experiment configuration. We use *CSR* [8] to measure whether two nodes are in the carrier sense range of each other. For example, we have nodes A and B. First, they broadcast alone as fast as they can. Then they broadcast together. Consider the ratio of their send rates when they are broadcasting together ( $S_{ab}, S_{ba}$ ) to the send rates, when they are broadcasting alone ( $S_a, S_b$ ). *CSR* is defined as:

$$CSR = \frac{S_{ab} + S_{ba}}{S_a + S_b}$$

If two senders are within the carrier sense range of each other, then only one of them would be able to send at a time, resulting in a *CSR* value of 0.5. If the senders are not within each other's carrier sense range, *CSR* will be 1. Intermediate values can result from noise, differences in sensitivity of antennas, signal strength fluctuations due to environmental factors, or combinations of such events.

We use *ETX* [7] as the metric for the quality of wireless links. *ETX* is based on packet delivery ratios, which are easy to measure, and reflect the link quality experienced by higher layers. Given a pair of nodes that communicate using the 802.11 protocol, the packet delivery ratio in both directions matters, since a unicast packet transmission is considered successful only if the sender successfully receives the ACK sent by the receiver. *ETX* takes into account the packet delivery ratio in both directions. For two nodes A and B, let  $P_{ab}$  be the packet delivery ratio from A to B and  $P_{ba}$  be the packet delivery ratio from B to A. Then *ETX* is given by:

$$ETX = \frac{1}{P_{ab} \cdot P_{ba}}$$

To measure the packet delivery ratio, we let each node

in turn broadcast 200 1K-byte packets. Only one node is active at a time. Other nodes measure the packet reception rate. This information is then sent to the control server where each link's *ETX* is calculated.

## 5.2 Wireless Node Management

In IvyNet, wireless nodes are kept free of persistent configuration state. Hence their memory and local disks are considered to be volatile soft state. After each experiment, the state of the wireless nodes is reset. The hard state of the wireless nodes, such as system images and system configurations, are kept at the node control server. To retain local disk modifications, wireless nodes must save them on the remote persistent storage of the node control server. In our current implementation, wireless nodes boot diskless and stateless over the network and mount the root file system from a NFS server. The NFS export for the root file system is set to be read-only. Since all wireless nodes share the same system image on the NFS server, any changes we make to the system image are immediately visible to all the wireless nodes. Since most OS distributions are built to run with writable media, a read-only root file system is not convenient for them. We solve this problem by integrating unionfs [9] file system into our system images. During system booting, a memory file system is created on each wireless node to store their temporal changes to their root file system. We use unionfs to combine the read-only root file system mounted from NFS and the memory file system into a read-writable root file system. Applications can write to the root file system, but all the changes are actually stored temporarily in the memory file system and will be lost once the node powers down. In addition, each wireless node has a private writable directory in the NFS server, this directory is mounted at the */root* directory and serves as the private persistent storage for the wireless nodes. To share data among each other, wireless nodes mount */share* to the same NFS directory which is writable to all the nodes. The modification to the */share* directory are visible to all the wireless nodes.

The current IvyNet implementation also supports an alternative way to boot the wireless nodes. Before each new experiment, the system image is distributed to the wireless nodes. The wireless nodes then boot from the local disk. Frisbee [10] is used to distribute the system image to the disk of the wireless nodes. Frisbee works over a reliable multicast session and is highly efficient and scalable. In the prototype, we only provide system images based on Linux but other operating systems could be supported as well.

IvyNet provides software tools for querying and managing the configurations of the wireless nodes during the experiment execution. The node control agent running on each wireless node, queries and records various

wireless NIC parameters using the Linux wireless tools. When the node management server receives query requests from users, it sends queries to the wireless nodes. After receiving the queries, the node control agent responds by reading the recorded parameter values. The users can also change the wireless interface configuration through the node control server. When receiving update requests from the server, the node control agent updates the specified parameters.

### 5.3 Experiment Control

IvyNet provides experiment control facilities to help users to run experiments automatically. Since ns-2 is the most popular experimental platform for wireless network research, IvyNet uses ns-2 scripts for the experiment description. The experiment control daemon resides on the experiment control server. It receives experiment specifications from users, interprets them, generates experiment configurations, and schedules experiments to execute. The execution of the experiments is directly controlled by the experiment execution daemon on the wireless nodes. The experiment execution daemon is implemented as part of the node control agent and it takes the experiment configurations received from the experiment control daemon and executes the experiments following the configurations. It also collects the status information about the experiment execution and reports them to the experiment control server. The prototype of this testbed can handle experiments only in a single-task and single-user manner.

The experiment control daemon consists of two parts: the experiment configurator and the experiment scheduler. The experiment configurator parses and verifies the user experiment specification and generates IvyNet specific experimental configuration files. Users specify experiments using *ns* scripts. Written in *Tcl*, the IvyNet parser recognises a subset of *ns*. It operates by overriding and interposing on standard *ns* procedures and *Tcl* primitives. Unrecognised *ns* commands abort the execution. The current implementation does not support topology auto-mapping. Nodes are assigned based on their preconfigured coordinates. The experiment scheduler reads the experiment configurations generated by the experiment configurator and schedules the execution of the experiments. It first notifies the trace collector to start the trace collection and then transfers the per node experiment configuration file to the wireless nodes after encoding them into XML format. The experiment execution daemon on the wireless nodes parses the experiment configuration and synchronises its local clocks using NTP. After all the wireless nodes are ready, the experiment scheduler chooses a time stamp in the near future to start the experiment. This time stamp is transferred with another XML mes-

sage to the wireless nodes. Then at the given time, the experiment execution daemons start the experiment. The execution status is sent finally back to the experiment control server in XML format.

Figure 5 shows the experiment description files for an experiment to measure the TCP fairness between two flows. The scene file in (a) describes the location of the experiment nodes. The traffic file, given in (b), specifies the behavior of the experiment nodes. In this example, the node `$_node(0)` connects at 0 s to the node `$_node(1)`. At 30 s the node `$_node(2)` connects to node `$_node(3)`. These two flows compete for the shared wireless link. At 90 s the node `$_node(2)` finishes its transmission. The other flow now uses the channel exclusively. The parameter `maxpkts_` specifies the number of data packets sent to the receiving host or -1 for an unlimited number. The parameter `packetSize` specifies the size of data packet. Figure (c) shows the command line used to execute the experiment on IvyNet. The command line arguments `-n` and `-stop` specify the total number of nodes used in the experiment and the total runtime of the experiment. The arguments `-cp` and `-sc` select the used traffic and scene files. The flag `-tc` enables the automatic trace collection.

### 5.4 Trace Collection

Various packet dynamics during the experiment execution are very important for researchers to analyze network applications and protocols. IvyNet incorporates packet trace collection facilities to aid this analysis. There are multiple trace collectors distributed in the testbed. They switch the wireless card to the RF monitor mode so that it can capture all 802.11 link-level transmissions including 802.11 protocol headers and control frames. Network sniffer programs, such as *tcpdump* and *ethereal*, are used by trace collectors for capturing packets. There is one central trace collection server which collects packet traces captured by trace collectors. It then merges the traces based on the timestamps and produces the final trace.

In a distributed environment multiple trace collectors are required to collect the entire network trace [15]. There are various possible approaches to deploy trace collectors in a testbed. In Mint [5], the trace collectors are set up directly in the wireless nodes. Each wireless node performs the monitoring 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. But the wireless nodes are required to have one additional wireless NIC, which increases both the cost and the complexity of the wireless nodes. The wireless card driver we are using (madwifi-ng [16]) allows to operate multiple virtual interfaces over one physical wireless interface. Different

```

$node_(0) set X_ 1
$node_(0) set Y_ 1
$node_(0) set Z_ 0
$node_(1) set X_ 1
$node_(1) set Y_ 2
$node_(1) set Z_ 0
$node_(2) set X_ 2
$node_(2) set Y_ 1
$node_(2) set Z_ 0
$node_(3) set X_ 2
$node_(3) set Y_ 2
$node_(3) set Z_ 0

```

(a) Scene File

```

set tcp_(0) [new Agent/TCP]
$ns_ attach-agent $node_(0) $tcp_(0)

set null_(0) [new Agent/TCPSink]
$ns_ attach-agent $node_(1) $null_(0)
$ns_ connect $tcp_(0) $null_(0)

set ftp_(0) [new Application/FTP]
$ftp_(0) attach-agent $tcp_(0)
$ftp_(0) set maxpkts_ -1
$tcp_(0) set packetSize_ 1024

$ns_ at 0.000 "$ftp_(0) start"
$ns_ at 120.000 "$ftp_(0) stop"

set tcp_(1) [new Agent/TCP]
$ns_ attach-agent $node_(2) $tcp_(1)

set null_(1) [new Agent/TCPSink]
$ns_ attach-agent $node_(3) $null_(1)
$ns_ connect $tcp_(1) $null_(1)

set ftp_(1) [new Application/FTP]
$ftp_(1) attach-agent $tcp_(1)
$ftp_(1) set maxpkts_ -1
$tcp_(1) set packetSize_ 1024

$ns_ at 30.000 "$ftp_(1) start"
$ns_ at 90.000 "$ftp_(1) stop"

```

(b) Traffic File

```

wtbadmin -cp tcpfairsharing.traffic \
-sc tcpfairsharing.scene \
-n 4 -stop 120 -tc

```

(c) Command Line

Figure 5: Experiment setup script

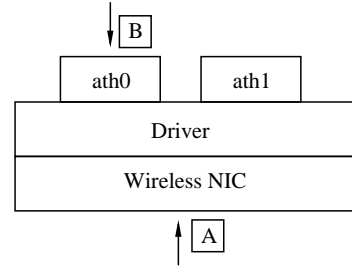


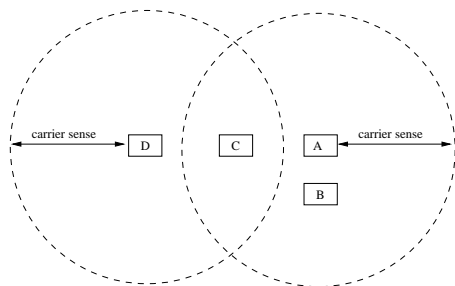
Figure 6: Packet reordering.

Table 1: Packet Capturing Ratio at  $C$

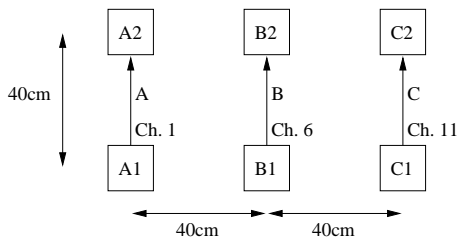
Distance (m)	Interference	% from A	% from B
1.5	no	97.4	97.8
0.3	no	99.3	99.5
1.5	yes	23.8	11.7
0.3	yes	95.5	98.2

virtual interfaces can work in different modes simultaneously. It seems that the virtual interface provides the possibility to collect traces on the wireless nodes without introducing an additional monitoring wireless NIC. But there are two limitations which make the virtual-interface-based packet capturing infeasible. First, with the current off-the-shelf hardware, MAC protocol logics are mainly implemented in the firmware. Many link layer frames, such as link-layer ACK packets, retransmitted data packets, and 802.11 control frames are managed directly by the card. If those frames are sent by the local wireless card, they will not be delivered to the driver, hence they are invisible to the sniffing programs. As a result, with the virtual monitor interface, a trace collector will never capture those MAC layer frames. Furthermore, capturing packets with the virtual monitor interface introduces a reordering problem. In Figure 6, two virtual interfaces are created. *ath0* works in ad hoc mode while *ath1* works in monitor mode. There are two packets *A* and *B*. Packet *A* is received by wireless NIC, at the same time packet *B* is ready to be sent out through *ath0*. Since the wireless card first receives *A* then transmits *B*, *ath1* should capture *A* before it captures *B*. Because the packet capturing is done by the driver instead of the wireless card, it is very likely that packet *B* is reported to *ath1* before packet *A*. Therefore in the packet traces captured from *ath1*, no correct packet order is guaranteed for the packets sent via *ath0*.

In IvyNet, we use dedicated monitor nodes to capture traces. At the cost of additional nodes, this approach separates the monitoring facility from the wireless nodes and makes wireless nodes simpler and more efficient. However, due to the high density of wireless nodes which are working simultaneously on the same channel, the effectiveness of this approach needs to be verified.



**Figure 7: Monitor nodes**



**Figure 8: Interference between non-overlapping channels**

In Figure 7,  $C$  is a monitor node. It captures traces for the communication between  $A$  and  $B$ . At the same time, a remote node  $D$  works in the same channel.  $D$  and  $A(B)$  are out of carrier sense range of each other and they can transmit packets in the same channel simultaneously.  $C$  sits between  $A$  and  $D$ . The transmission of packets from  $D$  interferes the reception of packets at  $C$ . To reduce the communication range, a 35 dB attenuator was attached to the antenna of  $A$ ,  $B$ , and  $D$ . The monitor node  $C$  works without additional attenuation. Table 1 shows the packet capturing ratio at  $C$ , when  $C$  is put 1.5 m and 0.3 m away from  $A$ . With each distance, the capturing ratio with and without interference from  $D$  are measured. The capturing ratio for the packets sent from  $A$  and packets sent from  $B$  are shown in separate columns in the table. From the table we can observe that when there is no interference from  $D$ ,  $C$  can always capture over 95% of packets transmitted from both  $A$  and  $B$ . When  $D$  starts to send interference traffic, the capturing ratio drops down. When  $C$  is 1.5 m away from  $A$ ,  $C$  can only capture 23% of packets sent from  $A$  and 11% of packets sent from  $B$ . Moving closer to  $A$ ,  $C$  becomes more resilient to the interference from  $D$ . When it is 0.3 m away from  $A$ ,  $C$  can capture more than 95% of the packets even with the interference. This experiment shows that, although it is difficult for the monitor node to capture all the packets from wireless nodes, it is possible to get a high capture ratio by strategically placing multiple monitor nodes in the testbed. We are currently working on software tools that aid users to distribute the monitor nodes.

## 5.5 Experiment Scheduling

To achieve higher experiment throughput, we can schedule multiple experiments to use the testbed concurrently by separating them in spatial, frequency or temporal domain.

To enable sharing the testbed in the *temporal* domain, the experiment scheduler has to know when each experiment is going to finish. Since the experiment control daemon tracks the status of the experiment nodes during the whole experiment execution, this information is available. The nodes also have to be reset to a known state between two experiments as described in Section 5.2.

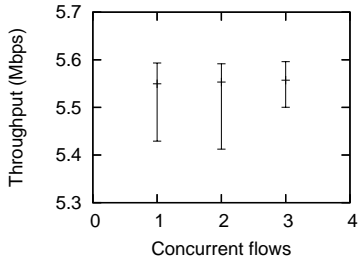
To enable sharing the testbed in the *spatial* domain, the experiment scheduler needs to know the carrier sense range of the experiment nodes. The carrier sense range can be determined using the software described in Section 5.1. The experiment scheduler ensures that no two experiments are running at the same time within each other's carrier sense range.

By running over multiple independent channels, experiments share the testbed in the *frequency* domain. There are 3 non overlapping channels for IEEE 802.11 b/g. The frequency-based schedulings are only valid if the communications on one channel do not interfere the communications on another channel. With experiment shown in Figure 8, we show that in IvyNet, communications on different non-overlapping channels are isolated to each other. In the experiment, all the nodes are within the transmission range of each other. They use fixed 30dbm attenuators and 11 Mbps bit rate. Three TCP flows run in parallel over three non-overlapping channels. Figure 9 shows the TCP throughput reported by nttcp of flow B with flow A and/or flow C running at the same time. The transmission power of A1 and C1 is always set to 19dBm. The transmission power of B1 is set to 19dBm and 0dBm respectively. We can observe that the throughput of flow B remains unaffected by the usage of the other non-overlapping channels.

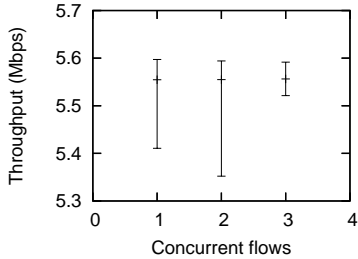
The prototype of IvyNet does not implement the experiment scheduler yet, but the results from this section indicate that experiments can be scheduled to run concurrently by sharing the testbed temporarily, spatially, or with non-overlapping channels.

## 6. CASE STUDY

We used IvyNet to study the performance of TCP on a 802.11-based multi-hop wireless network. In this experiment, we set up 5 IvyNet nodes in a chain within a 6 m x 6 m area. 30 dB attenuation is applied to each node. In Fig 10, the number beside the lines is the *CSR* of the two end nodes. The figure indicates that the neighboring nodes are well within the carrier sense range of each other. If two nodes are more than one hop away, they have *CSR* close or equal to 1. They



(a) 19dBm txpower

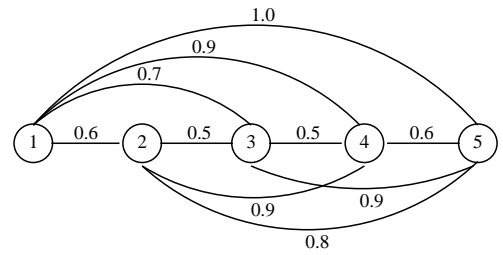


(b) 0dBm txpower

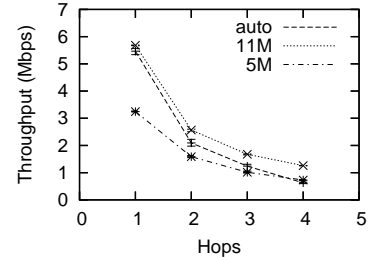
**Figure 9: TCP throughput with interference from non-overlapping channels**

are at the edge of the carrier sense range and can only weakly sense the existence of each other. Most of the time, they can transmit simultaneously over the same channel. TCP Reno sender resides at node 1. Routes were setup statically. Fig 11 compares TCP throughput over multi-hop routes when using different physical transmit rates. The sample rate algorithm is used for the auto-rate adaptation in the experiments. We can observe that over one-hop routes, using auto-rate we can get similar TCP throughput as when we use 11 Mbps bit rate. But with the length of route increasing, auto-rate adaptation becomes too conservative and fails to keep rate at the optimal value. For example, over 4-hop routes, TCP with auto-rate can only achieve half of the throughput compared to TCP with 11 M bit rate. The result of this specific experiment indicates that the current rate adaption algorithms for 802.11 wireless networks, which are evaluated mainly over single hop networks, interact poorly with TCP over multi-hop wireless networks and there is space for performance improvements.

This case study demonstrates the usefulness of IvyNet in protocol evaluations. With IvyNet we can easily set up a multi-hop wireless network and evaluate wireless protocols with real implementation in real network settings.



**Figure 10: Carrier sense ratio**



**Figure 11: TCP over multi-hop wireless networks**

## 7. SUMMARY

In this paper, we present the design of IvyNet, a novel multi-hop wireless network testbed. IvyNet is capable of providing multi-hop topologies in a laboratory setting with the fidelity of large-scale networks. We also discuss the design of the software toolkit for managing the testbed and facilitating the experiments. Such testbeds will be essential for efficient and realistic testing and evaluation of network protocols and applications. Given the significant interest in deploying wireless networks, such testbeds provide a cost-effective and manageable approach to go from laboratory setup to large scale networks.

## 8. REFERENCES

- [1] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and Chip Elliott: Experimental Evaluation of Wireless Simulation Assumptions. In Proceedings of the ACM/IEEE International Symposium on Modelling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), 2004
- [2] V. Naik, E. Ertin, H.W. Zhang, and A. Arora: Wireless Testbed Bonsai. The Second International Workshop On Wireless Network Measurement, 2006
- [3] E. Ertin, A. Arora, R. Ramnath: Kansei: A Testbed for Sensing at Scale, IPSN 2006
- [4] Raychaudhuri, D. and Seskar, I.: Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols, Proceedings of IEEE WCNC, 2005

- [5] Pradipta De, Ashish Raniwala, Srikant Sharma and Tzi-cker Chiueh: MiNT: A Miniaturised Network Testbed for Mobile Wireless Research, Proceedings of IEEE INFOCOM, 2005
- [6] Soekris Engineering: <http://www.soekris.com/>, World Wide Web
- [7] D.De Couto, D.Aguayao, J.Bicket, and R.Morris: A High-throughput path metric for multi-hop wireless routing, MOCICOM 2003
- [8] J.Padhye, S.Agarwal, V.Padmanabhan, L.Qiu, A.Rao, B.Zill: Estimation of Link Interference in Static Multi-hop Wireless Networks, IMC 2005
- [9] C.P.Wright, J.Dave, P.Gupta: Versatility and Unix Semantics in a Fan-out Unification File System, Technical Report, CS Department, Stony Brook University, October 2004
- [10] M.Hibler, L.Stoller, J.Lepreau, R.Ricci, C.Barb: Fast, Scalable Disk Imaging with Frisbee, USENIX Annual Technical Conference, 2003
- [11] B.White, J.Lepreau, L.Stoller, R.Ricci, S.Guruprasad, M.Newbold, M.Hibler, C.Barb, A.Joglekar: An Integrated Experimental Environment for Distributed Systems and Networks, OSDI 2002
- [12] J.Bicket, D.Aguayo, S. Biswas, and R.Morris: Architecture and Evaluation of an Unplanned 802.11b Mesh Network , Mobicom 2005
- [13] K.Pawlikowski, H.Jeong, and J.Lee: On credibility of simulation studies of telecommunication networks, IEEE Communications Magazine, 2002
- [14] D. A. Maltz, J. Broch, and D. B. Johnson: Experiences designing and building a multi-hop wireless ad-hoc network testbed. Technical Report, CMU-CS-99-11, 1999
- [15] J.Yeo, M.Youssef, A.Agrawala: A framework for wireless LAN monitoring and its applications, Proc. of ACM workshop on Wireless security, 2004
- [16] Madwifi-ng driver: [www.mawifi.org](http://www.mawifi.org), World Wide Web