UNIVERSITY OF NICE - SOPHIA ANTIPOLIS
Doctoral School STIC
Information and Communication Sciences and Technologies

THESIS

to fulfill the requirements for the degree of :
Doctor of Philosophy in Computer Science
from the University of Nice - Sophia Antipolis
Specialized in: WIRELESS NETWORKS

by

Shafqat Ur REHMAN

Research team: Planete, INRIA, Sophia Antipolis Méditerranée

BENCHMARKING IN WIRELESS NETWORKS

Thesis supervised by: Dr. Thierry TURLETTI
Dr. Walid DABBOUS

defended on 30 January 2012, in front of the committee composed of:

Président: Guillaume Urvoy KELLER CNRS/UNSA
Advisers: Thierry TURLETTI INRIA
           Walid DABBOUS INRIA
Reviewers: Andre-Luc BEYLOT ENSEEIHT
           Marcelo DIAS DE AMORIM UPMC
Members: Hossam AFIFI University of Paris XI
        Denis COLLANGE Orange Labs
Université de Nice - Sophia Antipolis
École Doctorale STIC
Sciences et Technologies de l’Information et de la Communication

Thèse

Présentée pour obtenir le titre de :

Docteur en Sciences

de l’Université de Nice - Sophia Antipolis
Spécialité : INFORMATIQUE

par

Shafqat Ur Rehman

Équipes d’accueil : Planete, INRIA, Sophia Antipolis Méditerranée

Benchmarking in Wireless Networks

Thèse dirigée par : Dr. Thierry Turletti
Dr. Walid Dabbous

Soutenance le 30 Janvier 2012, devant le jury composé de :

Président : Guillaume Urvoy KELLER CNRS/UNSA
Directeurs : Thierry TURLETTI INRIA
Walid DABBOUS INRIA
Rapporteurs : Andre-Luc BEYLOT ENSEEIHT
Marcelo DIAS DE AMORIM UPMC
Examinateurs : Hossam AFIFI Université Paris-Sud 11
Denis COLLANGE Orange Labs
Thèse

Benchmarking dans les réseaux sans fil

Benchmarking in Wireless Networks

Shafqat Ur Rehman
January 2012
Acknowledgments

First and foremost, I would like to express my gratitude to my supervisors, Dr. Thierry Turletti and Dr. Walid Dabbous, for their continuous support, guidance and patience. This thesis is greatly influenced by their encouragement, helpful discussions and keen interest. It has been a real pleasure to work with them.

I would also like to thank colleagues at INRIA, Sophia Antipolis for providing a fun and stimulating environment. I am especially grateful to Naveed Bin Rais, Shehryar Malik, Aswhin Rao, Imed lassoued, Yuedong Xu, Mohamed Jaber, Alina Quereilhac, Stevens Le Blond, Diego Dujovne, Cristian Tala Sánchez, Amir Krifa, Anais Casino, Lauri and Thierry Parmentelat. The Planète group has been a source of friendships as well as good advice and collaboration.

There are a lot of friends and mentors who have helped in countless ways to climb the ladder of higher studies. I cherish their company and support.

Finally, and most importantly, I wish to thank my parents who have taken great care of me through the thick and thin of my academic journey. They have been a great source of support, love and inspiration. To them I dedicate my thesis.

Shafqat UR REHMAN
Sophia Antipolis, France
DEDICATED TO MY PARENTS,
FAZAL AND MEHTAB
# CONTENTS

List of Figures ........................................... xii
List of Tables ............................................. 1

## 1 Introduction: Overview and Contribution

1.1 Context .................................................. 5
  1.1.1 How Wireless Networks differ from Wired Networks? ....... 5
  1.1.2 Impact of Wireless Channel on Network Performance ........ 6
  1.1.3 Evaluation Techniques .................................. 6

1.2 Wireless Benchmarking .................................... 9
  1.2.1 Problems ........................................... 10
  1.2.2 Fair comparison: Is it within sight? ...................... 11
  1.2.3 Performance Criteria: Metrics .......................... 13
  1.2.4 Full Disclosure Reports (FDR) .......................... 14
  1.2.5 Benchmarking Framework ............................... 14

1.3 Thesis Contributions ..................................... 16
1.4 Thesis Outline .......................................... 17

## 2 State of the Art ........................................ 19

2.1 Classical Evaluation Approaches ......................... 21
  2.1.1 Network Simulation .................................. 21
  2.1.2 Wireless Experimentation .............................. 23

2.2 Enhanced Wireless Experimentation ...................... 26

2.3 Metrics, Tools and Data Management ..................... 30
  2.3.1 Performance Metrics .................................. 30
  2.3.2 Wireless Tools ....................................... 30
  2.3.3 Data archiving and management ....................... 31

2.4 Requirements for Protocol Benchmarking ................. 33
3 Pitfalls of Wireless Experimentation

3.1 Avoiding Pitfalls

3.1.1 Antennas and sniffers can miss packets

3.1.2 Improper calibration of the sniffer

3.1.3 Traffic generators can be buggy

3.1.4 Multipath fading

3.1.5 Channel interference

3.1.6 Multiple antennas

3.1.7 Power control

3.1.8 Common problems with packet transmissions

3.2 General Advice

4 Benchmarking Methodology

4.1 Food for Thought

4.2 Background

4.3 Major Steps

4.3.1 Plan: Terms of Reference

4.3.2 Investigate: Existing best practices

4.3.3 Engineer: Benchmark tools

4.3.4 Deploy: Resource and Experiment control

4.3.5 Configure: Wireless experimentation scenario

4.3.6 Experiment: Undertake experiment execution and data collection

4.3.7 Preprocess: Data cleansing, archiving and transformation

4.3.8 Analyze: System performance

4.3.9 Report: Benchmarking score

4.4 Benchmarking Methodology in a Nutshell

5 Benchmarking Framework

5.1 Objective

5.2 Key Features

5.3 Design and Architecture

5.3.1 Architecture diagram

5.3.2 Control flow

5.4 Detailed Description

5.4.1 Server side modules

5.4.2 Client side modules

5.4.3 Running an experiment: User’s perspective

5.5 Deployment
### CONTENTS

5.6 Performance ................................................................. 69  
5.7 Conclusion ................................................................. 70  

6 Benchmarking Case Studies  ............................................. 71  
6.1 Case Study I: Wireless Channel Characterization .................. 72  
   6.1.1 Plan: Terms of Reference .................................... 72  
   6.1.2 Investigate: Existing best practices ......................... 72  
   6.1.3 Engineer: Benchmark tools .................................. 73  
   6.1.4 Deploy: Resource and experiment control .................. 74  
   6.1.5 Configuration: The Wireless experiment scenario .......... 77  
   6.1.6 Experiment: Undertake experiment execution and data collection .... 83  
   6.1.7 Preprocessing: Data cleansing, archiving and transformation ... 84  
   6.1.8 Analyze: System performance ................................ 84  

6.2 Case Study II: Multicast Video Streaming over WiFi Networks .... 92  
   6.2.1 Background ....................................................... 93  
   6.2.2 Experiments for multicast video streaming: Configuration and Execution .... 94  
   6.2.3 Method .............................................................. 95  
   6.2.4 Video Streaming Scenario ...................................... 95  
   6.2.5 Analysis: Multicast streaming performance ................. 98  
   6.2.6 Measurements during office hours ......................... 99  
   6.2.7 Measurements during non-office hours ..................... 102  
   6.2.8 Conclusion ....................................................... 103  

6.3 Metrics ................................................................. 106  
   6.3.1 Channel interference and RF activity in 2.4 GHz ISM band .......... 106  
   6.3.2 Ricean K-factor ................................................. 106  
   6.3.3 Packet Loss ratio .............................................. 107  
   6.3.4 Bit Error Rate (BER) .......................................... 107  
   6.3.5 Packet Error Rate (PER) ...................................... 107  
   6.3.6 Goodput ......................................................... 107  

6.4 Full Disclosure Reports ................................................ 108  

6.5 Fair Comparison .......................................................... 108  

7 Conclusion ................................................................. 111  
7.1 Overall Closure Comments ........................................... 112  
7.2 Perspectives of Future Work ......................................... 114
# Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Packet drops because of under-provisioned hardware/software</td>
<td>37</td>
</tr>
<tr>
<td>3.2</td>
<td>Improper calibration settings</td>
<td>38</td>
</tr>
<tr>
<td>3.3</td>
<td>Anomalous received power as measured UDP traffic generated by IPerl</td>
<td>39</td>
</tr>
<tr>
<td>3.4</td>
<td>Impact of small displacements on multipath fading</td>
<td>40</td>
</tr>
<tr>
<td>3.5</td>
<td>Impact of antenna orientation on multipath fading</td>
<td>41</td>
</tr>
<tr>
<td>3.6</td>
<td>Channel interference in 2.4 GHz</td>
<td>42</td>
</tr>
<tr>
<td>3.7</td>
<td>Multiple distinct received power levels when antenna diversity is enabled</td>
<td>43</td>
</tr>
<tr>
<td>4.1</td>
<td>Wireless network benchmarking</td>
<td>48</td>
</tr>
<tr>
<td>4.2</td>
<td>An instance of wireless benchmarking process</td>
<td>54</td>
</tr>
<tr>
<td>5.1</td>
<td>WEX toolbox design</td>
<td>60</td>
</tr>
<tr>
<td>5.2</td>
<td>Experiment workflow</td>
<td>61</td>
</tr>
<tr>
<td>5.3</td>
<td>Flow of data management process of WEX toolbox</td>
<td>62</td>
</tr>
<tr>
<td>5.4</td>
<td>Component-oriented architecture of WEX toolbox</td>
<td>63</td>
</tr>
<tr>
<td>5.5</td>
<td>Experiment description, its parsing and task scheduling</td>
<td>69</td>
</tr>
<tr>
<td>5.6</td>
<td>WEX toolbox performance</td>
<td>70</td>
</tr>
<tr>
<td>6.1</td>
<td>Indoor experimentation setup and placement of nodes</td>
<td>79</td>
</tr>
<tr>
<td>6.2</td>
<td>Displacement of probes</td>
<td>81</td>
</tr>
<tr>
<td>6.3</td>
<td>Timeline of events for each run</td>
<td>83</td>
</tr>
<tr>
<td>6.4</td>
<td>Temporal K factor measurements from 300 experiment runs during 3 weeks</td>
<td>85</td>
</tr>
<tr>
<td>6.5</td>
<td>Impact of orientation and displacement of receivers on K factor</td>
<td>86</td>
</tr>
<tr>
<td>6.6</td>
<td>Histogram of received power at the probes during one session</td>
<td>87</td>
</tr>
<tr>
<td>6.7</td>
<td>Temporal BER measurements from 300 experiment runs during 3 weeks</td>
<td>88</td>
</tr>
<tr>
<td>6.8</td>
<td>Impact of orientation and displacement of receivers on BER</td>
<td>88</td>
</tr>
<tr>
<td>6.9</td>
<td>Temporal PER measurements from 300 experiment runs during 3 weeks</td>
<td>89</td>
</tr>
<tr>
<td>6.10</td>
<td>Impact of orientation and displacement of receivers on PER</td>
<td>89</td>
</tr>
<tr>
<td>6.11</td>
<td>Temporal Packet loss ratio from 300 experiment runs during 3 weeks</td>
<td>90</td>
</tr>
</tbody>
</table>
6.12 Impact of orientation and displacement of receivers on packet loss ............... 91
6.13 Wireless testbed setup and placement of nodes ........................................... 97
6.14 Spectrum analysis in test case 6 ................................................................. 99
6.15 Average signal power per channel per run in test cases [1,6] ......................... 100
6.16 K-factor averaged over 5 runs ................................................................. 101
6.17 Received power recorded at Probe 2 in test cases [1,4] ................................. 102
6.18 RSSI averaged over 5 runs ................................................................. 103
6.19 Goodput averaged over 5 runs ................................................................. 104
6.20 Packet loss ratio averaged over 5 ............................................................... 104
6.21 Quality of video received at each client in cases [1,6] ................................. 105
6.22 K-factor and Packet loss ratio averaged over 5 runs .................................. 105
6.23 Full Disclosure Report for K factor ............................................................. 109
# TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Major requirements for a benchmarking framework for wireless networks</td>
<td>14</td>
</tr>
<tr>
<td>6.1</td>
<td>Planning for wireless benchmarking</td>
<td>73</td>
</tr>
<tr>
<td>6.2</td>
<td>State of the art: Literature and Tools</td>
<td>74</td>
</tr>
<tr>
<td>6.3</td>
<td>WEX Toolbox</td>
<td>75</td>
</tr>
<tr>
<td>6.4</td>
<td>LAN setup (Hardware requirements)</td>
<td>76</td>
</tr>
<tr>
<td>6.5</td>
<td>LAN setup (software requirements)</td>
<td>77</td>
</tr>
<tr>
<td>6.6</td>
<td>WEX cluster (server side)</td>
<td>78</td>
</tr>
<tr>
<td>6.7</td>
<td>WEX cluster (client side)</td>
<td>78</td>
</tr>
<tr>
<td>6.8</td>
<td>Wireless channel characterization: Four test cases</td>
<td>79</td>
</tr>
<tr>
<td>6.9</td>
<td>Wireless scenario (Stations and their role)</td>
<td>80</td>
</tr>
<tr>
<td>6.10</td>
<td>Timeline of experiment sessions on each workday</td>
<td>80</td>
</tr>
<tr>
<td>6.11</td>
<td>Multicast video streaming scenario</td>
<td>95</td>
</tr>
<tr>
<td>6.12</td>
<td>Wireless scenario (Hardware specifications)</td>
<td>96</td>
</tr>
<tr>
<td>6.13</td>
<td>Wireless scenario (software specification)</td>
<td>96</td>
</tr>
<tr>
<td>6.14</td>
<td>Multicast video streaming: Test Cases</td>
<td>98</td>
</tr>
</tbody>
</table>
1

**INTRODUCTION: OVERVIEW AND CONTRIBUTION**

Wireless has a big footprint in the home, office and public areas. It is incorporated in laptops, tablets, printers, smart cellular phones, VoIP phones, MP3 players, Blu-Ray players, and many more devices. Because of the proliferation of WiFi devices and applications, hundreds of millions of people across the globe have increased their online presence and stay connected for longer periods of time. It is becoming common to have at least one smart device (e.g., music player, smart phone, tablet, etc.) in addition to one primary computing device (e.g., personal computer). This has led to dramatic increase in the density of WiFi users. Note that most of the smart devices in the marketplace support WiFi only at 2.4 GHz. ISM (Industrial, Scientific and Medical) 2.4 GHz is license-free radio band that can be used by anybody in most countries. In the face of heavy utilization of 2.4 GHz radio band, WiFi faces the challenge of meeting performance requirements of data (such as file sharing, browsing and emails), audio (i.e., Voice over WiFi) and audio/video streaming applications.

To meet the needs of end users and enterprises, *performance and stability of WiFi systems and protocols must be thoroughly tested in the ‘real’ environment using real hardware and software. In science, a test result becomes accepted or trusted only if it can be verified (i.e., reproduced) by others. In wireless networks, *peer-verifiable or trustable experimentation is crucial for fair comparison. However, it is highly challenging.*

In this chapter, we will, first, provide necessary background information on wireless protocol
evaluation in the *real-world*. Then, we will elaborate the notion of *fair comparison* and *scientific rigor* in wireless experimentation. We will further discuss experimentation in the context of benchmarking and its relevance to protocol evaluation in wireless networks. Benchmarking is the key consideration in this thesis which also addresses the issues of scientific rigor and fair comparison. Finally, we will present our contributions and layout of the dissertation.
1.1 Context

Life without Internet is unthinkable for those who take it for granted. Wireless networks enable users-on-the-go to remain connected to the Internet without tethering to the network using messy cables. As of now, the most widely used wireless technology is WiFi technically referred to as IEEE 802.11, a cross-vendor industry standard. Originally dubbed as Wireless Ethernet, 802.11 WLAN (wireless LAN) technology has evolved to be more elaborate over the time than its 802.3 wired counterpart. It currently exists in the form of a set of standards such as 802.11a, 802.11b/g, 802.11n, etc., which implement a series of over-the-air (OTA) modulation techniques. The most popular are those defined by the 801.11b and 802.11g protocols. Therefore, we restrict ourselves to the evaluation of 802.11b and 802.11g protocols in this thesis. However, contributions of the thesis are not limited to these two standards and can be generalized easily to other wireless technologies.

1.1.1 How Wireless Networks differ from Wired Networks?

From technical point of view, LLC (Logical Link Control) and upper TCP/IP layers in both Wired (IEEE 802.3) and Wireless (IEEE 802.11) networks are same. However, wireless networks differ fundamentally from wired networks due to the nature of the wireless channel. Wireless channel is unpredictable. It fluctuates because of multipath fading. Small changes in position and orientation of the devices, mobility and human traffic can result in big changes in fading and power of received signal. Wireless channel must be shared by numerous WiFi Devices given the fact that it is a scarce resource. Unlike IEEE 802.3, IEEE 802.11 does not employ any collision detection algorithm. Rather it uses Carrier-Sense Multiple Access with Collision Avoidance or CSMA/CA as the primary access mechanism. To deal with the complexities arising from all this, IEEE 802.11 has become much more complex and elaborate.

Against single frame type used in IEEE 802.3, IEEE 802.11 defines 3 major frame types: management, control and data. Management frames are used by the network to join and leave the basic service set (BSS). Stations can join BSS through either passive scanning (by scanning beacon frames from APs) or active scanning (by sending probe requests). They are not necessary on the wired network since physically connecting or disconnecting the network cable performs this function. Control frames are used for the delivery of data frames. Control frames are also used to clear the channel, acquire the channel and provide acknowledgments. Most data frames carry actual data passed down from upper layers. Frames such as Null, CF-Ack, CF-Poll, though considered data frames, are utilized for purposes other than carrying data from upper layers.

IEEE 802.11 physical layer (PHY) provides an interface between the MAC and the wireless medium (air) which serves as a conduit for the transport of frames. The PHY provides three functions. First, the PHY provides an interface to exchange frames with the MAC layer for trans-
mission and reception of data. Second, the PHY employs signal carrier and spread spectrum modulation techniques to transmit data frames over the media. Thirdly, the PHY provides a carrier sense indication back to the MAC to verify activity on the media.

In short, Wireless networks are much more complex than their wired counterparts. In order to transmit data reliably over an unreliable wireless channel, IEEE 802.11 employs many control, management and data frames, and various coding and modulation techniques. It also uses intelligent MAC and PHY layers adaptive schemes (e.g., adaptive power control, adaptive rate control, adaptive modulation and coding (AMC), etc.) to adapt them to the channel conditions (e.g., path loss, interference, etc.). Because of these complexities, one might not know the detailed conditions of the experiments and therefore, may not be able to have correct performance evaluation.

1.1.2 Impact of Wireless Channel on Network Performance

In wireless networks, physical layer impacts operation at all layers of the protocol stack, although it can often be neglected in wired networks. Radio waves or WiFi signals propagating through air are very variable. Different environments have different propagation properties such as reflection, refraction, fast fading (or multipath fading), slow fading (or shadowing), path loss (or attenuation), etc. Even with the best wireless protocols, the best chipsets, the best RF design, the best software and the smartest antennas, wireless performance is going to vary a lot. Performance can also vary significantly, even in a fixed location, due to motion in the environment, interference and background noise in the WiFi spectrum.

1.1.3 Evaluation Techniques

Performance evaluation of wireless networking systems and protocols has recently witnessed tremendous research activity. Evaluation techniques employed range from mathematical modeling to in-field experimental evaluation each with its own pros and cons. A succinct overview of each of these is provided in order to put them in the right perspective in relation to benchmarking.

Modeling

Usually, the design of a new protocol typically begins with a model in someone’s head about the potential benefits of an algorithm in a certain application. Mathematical modeling allows for fast and easy evaluation. The behavior of the system under study is represented by a mathematical model used to derive the system’s expected performance. However, the models represent in general an approximation of the real system and are often too simplistic. They do
1.1 Context

not take into account many system aspects. For example, it is hard to model what path a radio signal will follow during propagation.

Simulation

In simulation, a software program mimics the behavior of the real system and uses mathematical modeling to represent parts of the target system. Statistics about the system’s performance metrics are obtained by running a sufficient number of simulations. Simulation provides a controlled environment for testing in network scenarios that can be scaled up to thousands or even millions of nodes. It enables fast exploration of the design space of the studied protocol under different network conditions, i.e. different environment parameters values. However, using simulation one must remember that the models may not reflect reality.

For example, in simulations, there is no one standard or general environment to adjust to, but there are classes of environments which represent typical situations for users, like conference halls, offices, trains, buses, airports and coffee shops. These typical places can be classified into Indoor, Outdoor and Indoor/Outdoor environments, using a LOS (Line Of Sight) or NLOS (Non Line Of Sight) criteria. An Indoor wireless environment is rich in signal reflections from different surfaces of objects, which gives an alternative path to signals to overcome an obstacle, while outdoors there are fewer reflections, giving a smaller chance to obtain an alternative path. Furthermore, on simulations there is also no movement around the devices which affect the signal level variability. Every time an object blocks the path between a transmitter and a receiver, the LOS path disappears, and the only path is the NLOS, if this path exists. Also, interference models in simulators are not complex enough to take into account the always increasing number of devices trying to share the same wireless band. A survey of the wireless networking literature reveals that majority of articles embrace simulation as a convenient approach for the performance evaluation of such systems [53]. However, lack of realism because of simplistic radio models in stochastic simulations can lead to misleading analysis [53] [60].

Validation, a process that determines whether a simulation model accurately represents the target system or not, is therefore required. This process is particularly difficult to perform because not only must the implementation of a simulated protocol be verified against its design specification, but the model must also be able to capture lower-level characteristics of the environment with a proper level of abstraction. The validation problem is alleviated in case an existing implementation of the protocol code can be compiled with the simulator. In this ap-
proach called direct execution simulation, the protocol’s logic is executed within the simulator and is driven by the simulator’s time-advancing mechanism.

**Emulation**

Network emulation is a hybrid approach that combines real elements of the network such as end hosts and protocols with simulated elements such as the network links, intermediate nodes and background traffic. Which elements are real and which are partially or fully simulated will often differ, depending on the experimenter’s needs and the available resources. An important difference between simulation and emulation is that while the former is mostly based on discrete time, the later must run in real time. Another difference is that it is impossible to have an absolutely repeatable order of events in emulation because of its real-time nature.

Emulation provisions repeatable but semi-realistic evaluation of protocols. Advantages are reconfigurability and control. However, it employs virtual/wired links instead of real wireless links and important channel characteristics such as path loss, packet errors, etc., are controlled and, therefore, 'not real'.

**Experimentation**

The best way to achieve realism is to perform in-field experiments using 'real' hardware and software. Availability of low-priced network equipment has made it feasible for research groups with modest resources to deploy network testbeds. Unfortunately, wireless experimentation is not a smooth process and configurability and management of even a small number of nodes is cumbersome. The behavior of the network is tightly coupled with the networking conditions. Since networking conditions vary with time, experimental results obtained will also vary. To take into account these varying conditions, it is important to know in which conditions the experiment was run so that we are able to interpret the results correctly.

Experimentation provisions non-repeatable but realistic evaluation of protocols. It is 'realistic' because experiments are conducted in the real environment.

Networking factors/conditions that lead to the variability of the results are traffic load, channel conditions (i.e., interference and fading), device configuration (including stations, access points, probes). In order to be able to make fair evaluation of protocols, it is necessary to document these conditions. Documenting the analysis process itself is required to enable others to repeat the analysis procedure on the data to which it is associated. A record of networking conditions together with analysis procedure (consisting of steps and tools/scripts) constitutes metadata. To achieve all this, it is highly desirable to have a standard methodology to first
establish the experimental setup, then executing the experiment, capturing, processing, archiving/sharing data/results and access control portal.

Provided that a pre-defined methodology is followed, it would be easier for researchers to compare between experiments which were executed under the same setup from one instance of an experiment to the following. Archiving and management of packet trace data, metadata, and tools in the form a shared database will enable others to correctly interpret/recalculate the results, reuse for another purpose or even reproduce (at least statistically) the scenario.

1.2 Wireless Benchmarking

Benchmarking is the act of measuring networking protocols, applications, devices and systems, under reference conditions. The goal of wireless benchmarking is to enable fair comparison between two or more protocols of the same category or between subsequent developments of the same system to determine ‘which one is better’.

Benchmarking in wireless networks is of great interest to the networking research community because it enables fair testing and validation of experimental results by independent experimenters. Independent experimental verification of results using the scientific method is precursor to scientific advancement.

Performance is judged using a small set of carefully selected metrics and the result is presented in the form of benchmarking score. Benchmarking score is a single number based on one or more metrics. It is also highly desired to be able to compare benchmarking scores contributed by independent researchers against a use case scenario. This can be achieved if all the contributors follow the same methodology.

Note that mechanisms described in Section 1.1.3 can be combined to form a federated testing framework for the evaluation of network protocols. An hybrid experiment then can leverage scalability of simulation, repeatability of emulation and realism of experiment to provision thorough evaluation. Even then several major issues prevent to have a scientifically rigorous evaluation process that would be close to the definition of benchmarking chief among them being lack of methodology and tools for unified description, execution and management of experiments. However, such an undertaking is beyond the scope of this thesis. We, rather, focus on benchmarking in wireless experimentation which is non-trivial because of several challenges.
1.2.1 Problems

In wireless networks, radio propagation is highly dependent upon the physical environment including geometric relationship between the communicating nodes, movement of nodes, movement of objects and the type and orientation of the antennas used. Unlicensed ISM (Industrial, Scientific and Medical) bands of the spectrum available are being shared by an increasing number of different devices making wireless medium more interference prone. There are up to fourteen channels on 802.11 b/g worldwide out of which only 3 channels are non-overlapping. In most cases, density of wireless nodes and a small number of non-overlapping channels make it impossible to ensure innocuous co-existence of different WLANs. Increased channel interference leads to degradation in network performance. Therefore, wireless channels are unpredictable (random), error-prone and could vary over very short time scale (order of microseconds).

Moreover, wireless networks are becoming more and more sophisticated. Modern APs can dynamically alter power levels and channel assignments in response to changing conditions. The rapid evolution in wireless technologies (e.g., introduction of smart antennas, directional antennas, reconfigurable radios, frequency agile radios, MIMO systems, multi-radio/multi-channel systems) makes benchmarking more complicated.

Reproducibility is at the core of benchmarking but factors mentioned above coupled with volatile weather conditions, ever-shifting landscape of obstructions, network equipment aging, software/firmware bugs, etc. make network retesting, reconfiguration and hence reproducibility a big challenge [29].

Off-the-shelf wireless cards are not designed as ‘test instrument’ and open source wireless tools and drivers are not free from bugs. Therefore, calibration is required to ensure reliability of the tools and correct interpretation of the results.

Data collected can be fairly large. Depending on the number of flows, data rates, duration of the experiment, and number of probes; collected measurement data can run into hundreds of Giga bytes. Synchronizing, merging and managing wireless traces is time consuming. In order to do the analysis, one needs to combine them into one coherent time-ordered sequence. It is costly in terms of time, computational and storage resources.

In short, erratic channel conditions sabotage experiment repeatability. Coupled with complexities of wireless technologies, network reconstructions and data management, reproducibility becomes an elusive target. Therefore, fair performance comparison in wireless networks is a big challenge.
1.2 Wireless Benchmarking

1.2.2 Fair comparison: Is it within sight?

Why it is difficult?

A major challenge to comparability in wireless experimentation is *uncontrollable* factors/parameters. They include station workload (memory/cpu usage, etc.), network traffic load, multipath fading, path loss (attenuation), channel interference, power spectral density (PSD), etc. Then there are parameters which can be configured to fixed values. These configurable network parameters are considered as *controllable*. Controllable parameters entail scenario configurations (topology, traffic, wireless card configurations). Controllable parameters can also include meta-data such as system hardware/software specifications. Note that *uncontrollable parameters are not the only obstacle in the way of comparability*.

It may seem trivial to keep track of all the controllable and uncontrollable parameters, but ensuring correctness and soundness of the measurement data can prove to be tricky. Some parameters and metrics concerning channel characteristics are influenced by, among other things, calibration settings. For example, if the impact of power/rate adaptation, noise floor calibrations and interference is ignored, it can lead to misleading results [71] and hence compromise the trustability of the results.

Experimental data sets, code, and software are crucial elements in scientific research; yet, these elements are noticeably absent when the research is recorded and preserved in the form of a scholarly article. Furthermore, most researchers do not deposit data related to their research article [69]. In [70], authors conducted an informal study of 33 of last year’s accepted articles from the prestigious ACM SIGCOMM 2010 conference. The study indicated problems with 70% of the accepted papers related to proper description of methodology, experiments, and analysis. This makes it difficult for peers and reviewers to confirm the results and hence reduces trust in the findings. This boils down to ‘repeatability’ of experiments.

Repeatability and Reproducibility

A key factor for comparability is *repeatability*. It means evaluation of system under test (SUT) at different moments in time should lead to the same result. Repeatability is a pre-requisite for reproducibility.
Reproducibility is the ability of an experiment to be replicated by independent reviewers on their own. It is not necessary for others to get exactly the same results. Usually, variation in the measurements is unavoidable. If the variation is smaller than some agreed limit, the experiment would be deemed reproducible. Reproducibility is often confused with repeatability which is being able to do the same experiment over and over at the same site by same experimenters and get the same results (i.e., with variation smaller than the agreed limit) each time.

When it comes to simulation, repeatability is easily achievable. Reproducibility, however, may still require more care and effort. In any case, it may be sufficient to maintain provenance (i.e., chronological record of measurement and analysis steps) of results (figures, tables, graphs, etc.) with all the parameters, data, source code and scripts. However, in real-world wireless experiments, both repeatability and reproducibility are non-trivial and elusive.

Reproducibility has been at the core of research in most fields of science [12]. An experimental result would be worthwhile only if it can be reproduced by peers. In natural sciences, an experiment is considered reproducible if the protocol/procedure used is described in sufficient detail along with reagents, specifications of the equipment, times, temperatures, etc. [13]. However, networking experiments cannot be reproduced by such measures because the distributed software is much more complex. Indeed, wireless experiments include additional complexities such as volatile radio spectrum, software/hardware imperfections/calibrations, configurability, management of resources and data, etc. [30]. It is impossible to ensure same channel conditions until and unless the experiment setup is insulated from outside interference using shielded enclosure. That may be the reason why rigorous peer verification of experimental results is not becoming a culture in networking field as yet [23]. In this dissertation, we restrict ourselves to the repeatability of wireless experiments. However, the roadmap presented is valid for reproducibility as well.

Because of uncontrollable experiment conditions, it would be impractical to repeat a real-world (non-shielded) wireless experiment the way experiments are repeatable in other fields of science.

Proposed Approach

We, therefore, focus on getting around this obstacle via conducting large number of runs, and clustering them according to the similarity of experiment conditions. Essentially, the entire procedure entails following steps: define the scenario precisely, conduct large number of runs
with fixed and recorded controllable parameters, measure all the uncontrollable conditions (i.e., parameters/metrics), cluster the runs as per the conditions, and perform an apples to apples comparison. This would provide a level playing ground for the fair comparison of networking protocols. This will also lead researchers to archive and share data and code and hence enable future researchers to compare their experimental results with the previous ones \[13\].

The above procedure requires a highly systematic and scientifically rigorous experimentation approach which is often challenging due to the cost, complexity, and scale of the involved experimental resources, and some potential limitations in the training of the research investigators \[70\]. Generally, an experimenter is required to deal with a host of issues such as testbed setup, installation of hardware/software tools, calibration and instrumentation of tools, sanity checks, imperfections and limitations of tools, scenario description, scheduling and management of experiment runs, meta-data collection, data cleaning, synchronization and merging of traces, data normalizations/transformations, analysis, reports, data management (measurement data, meta-data, code, assumptions, archiving, sharing, etc.). In this thesis, we address the issue of scientifically rigorous wireless experimentation 'in the wild'.

1.2.3 Performance Criteria: Metrics

As the objective of benchmarking is to learn whether a proposed networking protocol outperforms existing ones, it requires the definition of a set of criteria or indicators. This is achieved by the definition of metrics to characterize what performance means for the specific goal of the measurement study. We need to agree upon definitions of how the metrics are determined from packet traces, and by ensuring that traces themselves were measured according to standardized methodologies.

We categorize the metrics into groups: primary and secondary metrics. Secondary metrics are concerned with the channel characteristics such as multipath fading, RSSI (Received Signal Strength Indication) and channel interference. These metrics can undergo high variations depending on the channel conditions. Primary metrics are user-oriented network performance metrics, e.g., goodput, video quality, delay, etc. They depend on secondary metrics. Benchmarking score for the selected metrics is desired to be an average over sufficiently large number of runs. One or multiple metrics can be combined to form one or multiple benchmarking scores.

Ideally, benchmarking scores should not only be comparable to other scores obtained using the same testbed, but also with scores obtained from different testbeds with similar capabilities but potentially running different operating systems, or based on different types of hardware. The success of a specific benchmark may very well depend on whether this interoperability aspect is satisfied or not.
1.2.4 Full Disclosure Reports (FDR)

It is the complete documentation of a benchmark's score along with the meta-data (relevant scenario configuration and network conditions). There should be sufficient detail and coverage for someone else to be able to correctly interpret and compare the tests. FDR is desired to be a succinct one page performance report that is dynamically generated on-demand from the data repository and preferably published on the web.

1.2.5 Benchmarking Framework

Conception of a generic, workable and easily implementable benchmarking methodology is the first step, enforcing it on realworld experiments is not possible without necessary support tools. A methodology is expected to be applied over and over by researchers to benchmark the network performance in different use cases and scenarios. One pass through the methodology to test a single use case scenario constitutes one benchmarking cycle or benchmarking instance. A benchmarking cycle entails describing experiment, specifying configurations, executing experiment, taking measurements, computing benchmarking score, preparing reports and sharing benchmarks. A thorough performance analysis may require tens, hundreds or even thousands of tests to be scheduled for different scenarios of a use case. Note that a scenario represents a specific set of configurations and changing one factor/parameter, e.g., fragmentation threshold, beacon interval, node displacement, antenna orientation, human traffic, etc., would constitute a new use case scenario. A sustainable framework is required to facilitate reconfiguration of scenarios and carry out the aforementioned tasks. We present important requirements for such a framework in Table 1.1. The check list intended to be used as criteria for judging the competitiveness of benchmarking framework.

Benchmarking places three main requirements on benchmarking tools: reconfigurability, manageability and fair comparison.

**Table 1.1:** Major requirements for a benchmarking framework for wireless networks

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genericity</td>
<td>Is it easy for other research groups with moderate resources to use the framework on their own site using their own equipment?</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.1 – continued from previous page

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reconfigurability</strong></td>
<td>Is it easy to deal with configurations and reconfigure network devices and tools for each use case scenario usually requiring multiple experiments and runs?</td>
</tr>
<tr>
<td><strong>Expressiveness</strong></td>
<td>How flexible is the experiment description language (EDL)? How convenient it is to describe workflow, topology, applications, device configurations, traffic sampling, and meta-data?</td>
</tr>
<tr>
<td><strong>Sanity Checks</strong></td>
<td>Sanity checks are a set of conditions needed to be satisfied before an experiment can be expected to yield meaningful results. These may include time synchronization, power control, data rate, channel occupancy, RTS/CTS, antenna diversity, noise calibrations, sampling accuracy, etc.</td>
</tr>
<tr>
<td><strong>Re-execution</strong></td>
<td>It means that it is convenient and fast to run an experiment over and over conveniently. It can be used to verify the results, to debug the experiment and diagnose anomalies (Calibration problems and/or bugs), to refine the monitoring (traffic sampling, collection of meta data, etc.), to improve the methodology and to adjust topology, configurations, packet size, rate adaptation.</td>
</tr>
<tr>
<td><strong>Scheduling</strong></td>
<td>This means being able to schedule multiple (possible tens or hundreds or even thousands of) experiment runs in bulk. This is a 'key' requirement for scientifically rigorous experiments. It provides wide space for clustering of runs based on networking conditions.</td>
</tr>
<tr>
<td><strong>Resource Management</strong></td>
<td>This is concerned with the management of testbed resources including setup, configurations and monitoring.</td>
</tr>
<tr>
<td><strong>Resource Sharing</strong></td>
<td>Can the testbed resources be shared by other remote users? This is useful to cross-check experiments and validate the results.</td>
</tr>
<tr>
<td><strong>Experiment Control</strong></td>
<td>Is it easy to monitor experiment runs and tasks, and control their execution? Does it allow suspending or aborting runs.</td>
</tr>
<tr>
<td><strong>Trace Aggregation and Indexing</strong></td>
<td>Collection and possibly synchronization of the data, indexing of traces and meta-data corresponding to an experimentation campaign.</td>
</tr>
<tr>
<td><strong>Data Repository</strong></td>
<td>A central repository that holds everything that is concerned with network measurements and analysis.</td>
</tr>
<tr>
<td><strong>Channel Analysis</strong></td>
<td>Adjacent and co-channel interference</td>
</tr>
</tbody>
</table>
Table 1.1 – continued from previous page

<table>
<thead>
<tr>
<th>Channel Characterization</th>
<th>Path loss, multipath fading, SNR, BER, PER, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet Analysis</td>
<td>Packet loss, packet errors, goodput, delay jitters</td>
</tr>
<tr>
<td>Full Disclosure Reports</td>
<td>These reports anything that has impact on reproducibility. This includes measurements, results, scripts, meta-data, etc.</td>
</tr>
<tr>
<td>Data Sharing</td>
<td>Traces, scripts(scenario, analysis), meta-data, FDR</td>
</tr>
<tr>
<td>Methodology</td>
<td>Does the platform support a benchmarking methodology?</td>
</tr>
</tbody>
</table>

1.3 Thesis Contributions

The objective of this dissertation is to define a benchmarking methodology for wireless experimentation in the ‘real-world’, and demonstrate feasibility and workability of the methodology through the implementation of a benchmarking framework and case studies.

In this thesis, we present and explain the notion of benchmarking and present a methodology for realistic and fair evaluation of networking protocols. We identify and elaborate some common pitfalls for the unwary that were discovered during the implementation of the methodology and were used to improve it. The methodology is applied to two case studies: Wireless channel characterization and Video streaming over WiFi in indoor OTA environment. The case studies demonstrate workability of the methodology and provide insight into the technical details involved. We also present the toolkit designed for the purpose. The thesis makes following contributions:

- Identification and description of pitfalls that if ignored can lead to wrong interpretation of results.

- **Benchmarking methodology** which supports scientific rigor and promotes fair comparison. The methodology provides a general purpose well-defined evaluation procedure for wireless technologies particularly those operating in 2.4 GHz. It is designed to be comfortably implementable and deployable locally by researcher groups using their own testbeds and tools.

- **Software tools** which we developed to verify the workability and practicality of the proposed methodology. Although the methodology is, for the most part, experimentation
oriented, it is adaptable to emulation and hybrid evaluation approaches as well. Salient features of the tools are as follows:

- **XML-based experiment description language (EDL)** which allows easy description of wireless experiments.
- Remote **reconfiguration, scheduling and automatic running** of multiple experiments and runs.
- **Data repository** of traces, metadata and scripts.
- **Indexing and management** of large number of runs. Packet traces are merged, synchronized and stored in MySQL database.
- **SQL based statistical analysis.** Benchmarking score and corresponding confidence intervals for the metrics.
- Scripts to generate result reports and plots. We also propose a sample **full disclosure reports (FDR)** for the metrics.

Two case studies to demonstrate the methodology. First case study undertakes empirical measurement and estimation of WiFi channel characteristics. Second case study undertakes video streaming over WiFi networks and investigates impact of interference and multipath fading over goodput and video quality in indoor OTA test environment.

### 1.4 Thesis Outline

The rest of the thesis is organized as follows:

In Chapter 2, we investigate contemporary simulation and experimentation approaches in the wireless networks. We are especially interested in efforts made in the direction of scientifically rigorous experimentation. The goals is to understand existing approaches and what can be borrowed or reused from them so as to avoid reinventing the wheel.

In Chapter 6, we discuss some pitfalls of wireless experiments that we learned though experience. Some pitfalls are not straightforward to decipher and may take up considerable time to uncover. The goal is to help the unwary avoid repeating mistakes of the past, save time and be more productive.

In Chapter 4, we present a ground up methodology for benchmarking of wireless networks. The methodology covers steps from planning to measurements to data management to sharing benchmarks. The goal is to design methodology which brings scientific rigor to wireless experiments and provides guidelines to enable fair comparison in the real world.

Finally, the thesis concludes with Chapter 7 in which we provide sum and substance of our work and provide direction for the future.
As stated earlier, we have witnessed an explosion in the amount of data hauled by Internet to and from wireless end users equipped with WiFi enabled laptops, tablets and smart phones. A large part of the information exchanged by end users results from their activities on social networking and video-sharing websites. VoIP tools such as Skype are also hugely popular and generate a significant amount of traffic. This leads to increased load on 2.4 GHz ISM band which is shared by bluetooth, cordless phones, microwave ovens, game controllers, media players to name a few. Therefore, WiFi networks have become a hot topic for networking research community. This is evident from the development of enhanced models for wireless simulations, new emulation tools and experimentation platforms. The purpose has been to cope with the performance evaluation challenges brought forth by wireless networks which are distinct from their wired counterparts.

Being able to run a wireless experiment in the real-world is one thing but making the results trustable is quite another. An experiment result would become trustable only if it is based on sufficiently large number of runs and is statistically repeatable [Section 1.2.2]. Trustability is a prerequisite for fair comparison of protocols.

In this chapter, we will see as to what extent state of the art serves this goal. In this perspective, we will shed light on contemporary performance evaluation approaches especially experimentation. First, we will discuss simulation-based approaches and then look into contemporary experimentation approaches in greater detail. We will further investigate contributions in the direction of scientifically rigorous experimentation which we call enhanced experimentation. In addition, we explore wireless trace data repositories and some specialized experimentation
tools which can be leveraged easily in wireless experimentation platforms.
2.1 Classical Evaluation Approaches

Common approaches for the evaluation of wireless networking protocols are simulation, emulation and experimentation as introduced in Chapter 1. Simulation has been and is still the dominant of all. The reason is it is relatively easy and quick. It simplifies the complexities of the real-world and results in substantial productivity increase. For wired networks, it lives up to the promise and the results are generally considered reliable. However, the same is not true for wireless network simulations where sometimes results may deviate significantly from the reality [53]. This is because of the lack of the fidelity of the underlying simplistic communication models. Lately, simulation based approaches have been losing ground to emulation and particularly experimentation which provide greater realism. In this section, we investigate wireless simulation and experimentation approaches in conjunction with simulators and testbeds, and discuss their pros and cons for wireless networks.

2.1.1 Network Simulation

ns-2 [64] is a popular network simulator. It was primarily designed to simulate wired networks. The default implementation for simulation of IEEE 802.11 within ns-2 is strongly influenced by Sun Microsystems and the UC Berkeley Daedalus and CMU Monarch projects. Several publications point out the fact that wireless models implemented in ns-2 do not cover all the effects of realworld setup [53]. There are some reasonable propagation models in ns-2, ns-3 [90] and other simulators (e.g., OMNet++) which take into account the impact of interference, however, they make some simplifying assumptions. Moreover, most of the research uses simple two-ray ground-reflection or shadowing models. Some of the common assumption made by propagation models are circular transmission area, symmetric links, path loss based on distance which do not necessarily hold in reality. In realworld, angle of the source and receiver antennas also influences the reception of a packet [53]. Beyond this, there are several instances of oversimplifications or inaccuracies, e.g., wrong collision handling, no preamble and PLCP header modeling, wrong backoff handling, incomplete support for capture, etc [98]. Packet and bit error rates depend more on interference and placement of walls than on distance [96]. Furthermore, in ns-2 energy model, battery with linear charge and discharge characteristics is assumed. However, real batteries like lithium-ion cells show non-linear behavior.

ns-3 is a new network simulator written from scratch based on yans(yet another network simulator) [97]. The focus has been to improve the core architecture, software integration, models, and educational components of ns-2. In addition to natively-implemented protocols, it allows existing real-world protocol implementations to be reused and supports integration of real-world network protocols and stacks in the simulation [100]. ns-3 can also be leveraged in hybrid wireless evaluation. By using Network Experiment Programing Interface (NEPI)
researchers can mix real experiment nodes with simulation nodes and evaluate the performance of the network using real traffic either from a live network or unmodified applications. For example, in a multicast video streaming scenario, source and receivers can be real applications running on real (or virtual) machines and intermediate distribution network can be implemented in the simulation. This empowers the researcher to investigate the performance of system-under-test (SUT) with diverse test cases. However, despite all the improvements in simulation models, simulation cannot replace the realworld and always suffers from deficiencies. For example, basic simulation unit assumed in ns-3 and other simulators is frame (or packet), which omits considerations of the signal processing details at the physical layer, such as frame construction and reception.

J-Sim (formerly JavaSim) is a network simulator written in Java. On abstract level, J-Sim distinguishes two layers. The lower layer Core Service Layer (CSL) comprises every OSI layer from network to physical, the higher layer comprises the remaining OSI layers. J-Sim implements only basic level support for wireless/mobile network simulation. The only MAC protocol supported is IEEE 802.11.

There are several other simulators with some support for wireless network simulation. For example, GloMoSim was developed at UCLA (California, USA) and provides a library for parallel (SMP) simulation of large-scale wireless networks. OMNet++ offers support for wireless ad hoc network simulation. QualNet is a commercial ad hoc network simulator based on GloMoSim’s core. It extends GloMoSim by bringing support and a set of user-friendly tools. OPNet was first proposed by MIT in 1986. It is now the most widely used commercial network simulator with large number of wireless network models. SWANS, developed at Cornell university (New York, USA), is built atop the JiST discrete event platform.

Simulation simplifies the complexities of the real-world and results in substantial productivity increase. But for wireless networks, the real-world is more complex and often gets oversimplified in the simulation. Wireless simulation allows fair comparison but compromises the realism.

Despite the aforementioned problems that plague simulation, simulation often constitutes an important first step to verify the correctness and desired behavior of a solution. After ascertaining the preliminary efficacy through simulations, experimental evaluation is performed to finalize the solution. However, wireless experiments can be highly complex. A lot of research, therefore, simply ignores this vital step. The following section presents a critical analysis of existing experimentation facilities.
2.1 Classical Evaluation Approaches

2.1.2 Wireless Experimentation

Ad-hoc Protocol Evaluation (APE) testbed [63] is a well-known testbed for evaluating routing protocols in large scale Mobile Ad hoc Networks (MANETs). APE is a stripped down Linux distribution based on Red Hat Linux which can be booted directly from CD on regular notebooks. It incorporates mobility by strictly choreographing movement of volunteers carrying laptops. To a certain extent, this makes experiments repeatable. Authors have conducted indoor experiments using up to 37 nodes. They have developed virtual mobility metric based on measured signal quality. The idea is to use per packet signal quality to compute virtual distances between the nodes. These distances describe the topology of the network as it is perceived by the nodes and are used to determine how similar two repetitions of an experiment are with respect to connectivity. The authors ran several experiments with OLSR, TORA, DSR and AODV. By comparing the virtual mobility graphs of the distinct experiments the authors conclude that the choreographed approach is suitable to produce comparable test runs. They have integrated tools for choreography configuration, ethernet/IP level trace collection and uploading of these traces, at the end of an experiment, to a central machine. Also they provide perl scripts to calculate virtual mobility, connectivity, hop count, link changes, path optimality, etc. A basic level mechanism is provided to synchronize and organize trace files against each run. APE is limited to i386 compatible computers equipped with ORINCO IEEE 802.11 WaveLAN cards. It is specialized for MANETs but not designed for shared usage and diverse experiment scenarios. It lacks remote management, resource sharing and facilities to repeat/replay experiment easily.

EXC toolkit [117] focuses on simple scheduling of experiment runs, monitoring and management of nodes in MANETs, VANETs and mesh networks. It is quite similar to APE in the sense that each experiment is portioned into runs. However, it promotes the concept of semi-automatic experiments where each experiment run is launched manually but all the steps within the run are fully automated. This provisions an opportunity to the researchers to check for any problems before proceeding to the next run. Also, it is portable to other hardware and facilitates researchers to setup their own testbed using their own hardware with moderate effort. It is implemented using Ruby scripting language. At the core is an event handling framework. Configurations, scenario descriptions and movements (of nodes) are specified in XML. Each XML element corresponds to an event. However, it is cumbersome to repeat the same experiment multiple times which is important to improve confidence in the results. Control and management features are very basic and are not sufficient for large scale rigorous wireless experimentation. Furthermore, both APE and EXC are not remotely accessible to the broader research community as shared research testbeds. Instead, researchers are intended to setup their own testing facilities using these frameworks.

MiNT, a miniaturized mobile multi-hop wireless network testbed [118], is an effort to evaluate 802.11b-based multi-hop wireless networks with mobility using as little space as possible while
Chapter 2: State of the Art

providing the fidelity of experimenting on a large scale testbed. RF signal emanating from
the radios is artificially manipulating using attenuators and Digital Signal Processors (DSPs)
to shrink the full scale physical testbed to a compact and manageable size. This makes it
possible to implement a multi-hop mobile ad hoc network on a 12 ft by 6 ft tabletop with up
to 8 nodes. It is conceptually very similar to the mobile nodes in Emulab, although developed
independently [65]. Nodes have multiple wireless interfaces for various purposes; the ones
used for the protocol under test are highly attenuated to simulate the loss of much larger areas.
Additionally, the MiNT platform integrates with ns-2 to provide a hybrid simulation/emulation
environment (with its link, MAC and physical layers replaced by real hardware and drivers)
[118]. It allows unmodified ns-2 code to be executed on a set of physical nodes. In the initial
version, the mobile nodes were simple antenna platforms connected by RF cables to PCs where
the actual processing took place. The MiNT-m paper [119] describes improvements to dispense
with the stationary PCs, along with additional management tools. The testbed infrastructure
consists mainly of mechanisms for node tracking, positioning, control, state logging, and state
rollback. However, the testbed faces problems in ensuring that there is no radio frequency (RF)
signal leakage from the attenuators/connectors and adequate shielding is done to disallow the
radios to achieve their natural radio range. Even a small amount of RF leakage may induce
large errors in the testing and evaluation. MiNT is not aimed at repeatability. It does not
support replaying an experiment run arbitrary number of times and data management.

Emulab [59] (developed at the University of Utah) is a large network testbed which provides
integrated access to a range of experimental environments (simulation, emulation, wired area
networks, 802.11 wireless, etc.) in which to evaluate the system under test. It unifies all
these environments under a common user interface and an integrated common framework.
It provides remote access and resource reservation. Emulation experiments allow to specify
arbitrary network topologies through emulated network links provisioning a controllable and
repeatable environment. IEEE 802.11a/b/g wireless testbed is deployed on multiple floors of
an office building. All nodes have two wireless interfaces, plus a wired control network. Emulab
testbed can support experiments requiring very large number of nodes. This is made possible
by multiplexed virtual nodes (i.e., virtual machines) which allow an experiment to use 10-20
times as many nodes as there are physical nodes in the testbed. Applications are run in virtual
machines and communicate through virtual links. This approach is practical for experiments
with modest CPU, memory and network requirements.

Emulab provides tools to describe a required experiment topology and map it to actual re-
ources. An extended syntax of Network Simulator (ns) [64] or alternatively, a GUI in Emulab’s
Web interface is used to specify the virtual topology. Emulab uses this specification to automati-
cally configure a corresponding physical topology. Using Testbed Event Client (tevc) and event
agents, some user events e.g., start/stop programs, start/stop/modify traffic generation, take
2.1 Classical Evaluation Approaches

links up/down, etc., can be dynamically controlled. Some control tools are also provided, but
they provide minimal features. Wex toolbox and Emulab share some features: Both facilitate
wireless experimentation through control and experimental networks, provide remote access to
the testbed, enable remote monitoring of the resources (nodes) and allow Linux images. How-
ever, WEX Toolbox [32] is more specialized for IEEE 802.11a/b/g WiFi networks and the two
platforms differ in terms of technological focus, hardware and target audience. Furthermore, it
lacks methodology, trace management/synchronization and post processing.

The cOntrol and Management Framework (OMF) is a framework for controlling and manag-
ing networking testbeds [31]. OMF was originally developed for the ORBIT wireless testbed at
Winlab, Rutgers University [23]. Now it is an open source framework, which supports heteroge-
neous wired and wireless resources. Work is under progress to federate OMF with existing and
upcoming testbeds in order to provide a unified control and management framework. Currently,
testbed resources (nodes) are identified by a 2D [x,y] coordinate scheme. OMF provides a set
of services to manage and operate the testbed resources, e.g., resetting nodes, retrieving their
status information, installing new OS images. It uses a Ruby-based domain specific language
called OEDL (OMF Experiment Description Language) [31] to describe an experiment. OMF
Experiment Description (ED) is basically an OEDL script that specifies the resources required
for an experiment, their required configuration, measurements to collect and state machine of
tasks to perform. OMF executes the experiment and collects the measurements as specified in
ED. Integration of existing third-party applications such as traffic generators is facilitated, al-
though, it requires instrumentation (inserting measurement points inside the source code) and
creating prototypes [31] on part of the user.

Traffic sampling and measurement collection is managed by OMF measurement library
(OML) which is based on client/server architecture and comprises OML Collection Server and
an OML Measurement Library (Client). Measurements and sampling rates are specified using
measurement points (MPs). Samples are collected at the OML server into a sqlite database
with one table for each distinct MP. The applications running on a given node forward the re-
quired measurements to their OML Client. The OML Client applies any defined pre-processing
filters on this data and sends the result to the OML Collection Server. For example, OMF has
an integrated measurement and instrumentation framework. However, OMF heavily depends
on instrumentation and measurement points. This makes it difficult to integrate existing appli-
cations into OMF. For example, using IPerf in conjunction with OMF requires instrumentation
of IPerf and creating a ruby-based prototype application for IPerf. Instrumentation of IPerf is
required to enable OML to collect user-defined measurement points from within IPerf. Fur-
thermore, it lacks trace management and post-processing. Support for dealing with wireless
headers is not clear. Repeating an experiment large number of times is cumbersome because it
involves manual intervention for each run. It has been developed and tested on ubuntu Linux
distribution only.

In all the above mentioned experimentation frameworks, there is no efficient mechanism to repeat an experiment arbitrary number of times and synchronize and manage traces at the end. Furthermore, automatically scheduling multiple experiments is not supported.

In the past decade, the main impetus for development of testbeds has been realism and large scale. Now that several testbeds have been developed, management of experimentation process and analysis of results has become complicated and hinders quality research systematically and efficiently.

2.2 Enhanced Wireless Experimentation

In order to make easier the management of experiments, configuration of scenarios and processing of packet traces, researchers have proposed various enhancements to classical wireless experimentation approaches discussed in Section 2.1.

In [79], after an extensive analysis of published work, authors conclude that lack of information about scenarios, methodologies, and meta-data hampers reproducibility and peer verification. They have proposed a web portal called LabWiki where experimenters can describe the experiments and store all the information about an experiment. Each of the artifacts of an experiment and the experiments themselves are identified by public URLs which can be linked easily from any LabWiki pages. Authors propose to use the R language to analyze measurements collected on the portal. After the experiment development process is finished, a Portal user may choose to open up his or her LabWiki permissions to specific reviewers or the general public. However, the data repository is not implemented as yet and the portal is still in development process [68]. More importantly, support for multiple runs, management of large experimentation campaigns is unclear. Statistical analysis and result reports are left for the user to deal with.

MyEmulab [59] is a web-portal to the Emulab network emulator testbed. It provides services to build experimental network topologies, upload an experiment description, automatically configure and execute the experiment. Furthermore, it provides wiki and versioning tools to allow collaboration between members of a given project. However, MyEmulab does not provide services to archive, access, analyze the measurements. Also it does not offer any services to report and share the results.

NEPI (Network experiment programming interface) [72] proposes a framework based on
a unified object model to describe a networking experiment which could subsequently be executed on different environments (e.g. simulations, emulations, testbeds). However, it is an ongoing work and differs in terms of focus. Currently, it does not handle real-world wireless experiments and lacks the support of multiple runs, collection of meta-data, data management, analysis, etc.

In [4], a comprehensive practice has been recommended by IEEE 802.11T Task Group for 802.11 wireless experiments. It contains essential information for setting up test scenarios in different wireless environments and metrics that should be considered in each scenario. The recommended test environments include calibrated over the air test (COAT) environment, conducted test environment, over the air (OTA) outdoor Line of sight (LOS) environment, OTA indoor LOS environment, OTA indoor non-line of sight (NLOS) environment and OTA shielded enclosure environment. The draft deals with two primary wireless issues namely interference and mobility. 802.11T has identified three principal use cases namely data, latency-sensitive and streaming media.

- **Data** use case covers data applications (such as web downloads, file transfers, file sharing, and email) which usually do not impose time constraints on the data delivery. Performance test metrics for data use case include Throughput vs. range, AP capacity, AP throughput per client.

- An example **latency-sensitive** use case is Voice over WiFi which is a time-critical application. In order to guarantee QoS requirements, following performance metrics are recommended: voice quality vs. range, voice quality vs. network load, voice quality vs. call load and BSS transition (roaming) time where voice quality is gauged by latency, jitter and packet loss.

- **Streaming media** use case includes audio/video streaming applications. This use case places the most stringent QoS guarantees. Performance metrics include video quality vs. range, video quality vs. network load where video quality represents throughput, latency and jitter.

IEEE 802.11T guidelines also provides a mapping between metrics and the test environments showing which metrics can best be measured in which test environment. For instance, measuring BSS transition or roaming time in OTA LOS/NLOS environment is not feasible for most of the researchers. Wherever mobility and range constraints come into play, IEEE 802.11T recommends measurements in conducted environment where path loss is controlled using attenuators. 802.11T recommendations offer valuable pieces of advice but putting these recommendation to practice is not straight forward. The challenge comes from testbed setup, testbed
management, experiment scheduling, trace and meta-data management, trustability and comparability of results. Furthermore, the recommendations focus on application level metrics and do not provide any guidelines for how to correlate channel characteristics and application level metrics.

An experimentation methodology for wireless networks is proposed in [105]. Further efforts were made to develop a platform which could make it easier to follow the methodology [116]. The proposed methodology envisages six steps for a wireless experiment namely layout definition, tasks and parameter configuration, run and capture, trace processing, analysis and storage. A brief overview of each step is as follows:

1. **Layout Definition** The first step is to document specifications of the wireless equipment, decide the placement/topology of nodes and record details about the wireless environment. According to the authors, this will help others to reconstruct an experiment in a similar test environment in order to reproduce the results.

2. **Tasks and Parameter Configuration** The second step is to describe the experiment. The tasks that make up an experiment consist of wireless interface configuration, traffic generation and traffic capture. Common configuration parameters for each task include task start time, task end time, target execution node. Each task can also have additional specific parameters, e.g., traffic generator may be configured with parameters such as traffic pattern, data rate, duration, etc. An experiment is defined using a set of shell scripts. There is one shell script file for each kind of task. For example, one task to configure AP, one task to configure probes, one task to run traffic generator, one task to run sniffer. Each task file is annotated with the parameters such as start time, end time and target node. Authors have used IPerf [84] as traffic generator and TShark [45] as packet sniffer.

3. **Run and Capture** This step launches the tasks defined in step II. It accomplishes two things: first, executing an experiment and second, collecting the packet traces from probes.

4. **Processing** This step is solely concerned with loading traces into a MySQL database. CrunchXML [42], a trace processing tool, is used to merge and synchronize traces captured at different probes as explained in [2.3]. It also serves as a filter between packet trace and database in the sense that it select only a subset of the packet fields based on the fields specified in the database schema.

5. **Analysis** As per the methodology [105], this step was aimed at calculating the selected metrics from the merged packet trace in the database. However, this step was largely ignored in the first version of the toolbox.
6. **Storage** As per the methodology [105], this step was aimed at storing layout information, node configuration, experiment description, processing and analysis scripts in the database. However, the first version ended up with storing only packet traces in the database.

As indicated above, last two steps of the envisaged experimentation methodology were still in infancy. We have identified some issues with the experimentation platform vis-a-vis methodology which are as under:

Experimentation description was based on shell scripts. One needed to describe the experiment using several shell scripts (depending on the number of tasks) one for each kind of task. Also the description needed to include attributes specific to Grid Engine [49]. This mechanism suffers from a few drawbacks. First, maintenance of several script files is cumbersome especially when one needs to perform large number of experiments. Second, requiring an experimenter to write tool-specific commands in the experiment description compromises simplicity. Third, large number of tasks means large number of scripts. This makes it difficult to find syntax errors and to debug scripts.

The third step (i.e., *run and capture*) did not support multiple runs of the same experiment. In order to run multiple runs, experimenter would need to wait for the first run to finish and then launch the next run manually and so on.

In the fourth step (i.e., *processing*), experimenter would need to create database schemas, export pcap trace to XML file and then launch CrunchXML on each XML file. This makes processing of large number of experiments cumbersome, time consuming and undermines productivity.

In a nutshell, although progress is being made to facilitate not only conducting wireless experiments easily but also designing scenarios, documenting configurations, etc., current experimentation methodologies suffer from several drawbacks. They don't support orchestrating and managing large number of runs of an experiment, recording network conditions (i.e., metadata), and combining packet traces and metadata for thorough analysis.

To fill this gap there have been efforts to develop metrics, specialized experimentation tools and data repositories some of which are discussed in Section 2.3.
Chapter 2: State of the Art

2.3 Metrics, Tools and Data Management

2.3.1 Performance Metrics

Selection of good metrics is necessary for producing useful benchmarks. In [12] [11], authors provide guidelines for the selection of performance metrics for benchmarking. However, these contributions are not complemented by a measurement methodology which is important for trustable results and meaningful comparison.

2.3.2 Wireless Tools

Effort is also being made to develop supporting analysis tools [58] [56] [103]. A brief analysis of worthwhile wireless tools is provided here.

SWAT (Stanford Wireless Analysis Tool) [22] [102] automates gathering and analysis of network measurements. It provides an interface for configuring experimental parameters in a network. It collects raw packet statistics such as the received signal strength and chip error, and provides modules for calculating and visualizing various metrics derived from these statistics. However, it is tailored for low-power sensor networks.

CrunchXML [42] is a trace processing tool developed at INRIA which can load pcap packet traces into MySQL database. Usually a wireless experiment employs multiple probes to capture traffic at channel(s) of interest. For each experiment, majority of the packets are duplicate between packet traces captured at different probes. Merging the packet traces by storing duplicate packets only once can save a lot of disk space and greatly speed up packet analysis. However, as the probes work independently of a BSS or IBSS, MAC timestamps in RADIOTAP headers are not synchronized. This complicates merging of the traces. CrunchXML solves this very same problem. It implements an efficient synchronization and merging algorithm. A PCAP trace first needs to be exported to XML (or PDML) format using TShark/WireShark which is used as input by CrunchXML. It stores only those packet fields that have been marked as required by the user in the database schema. In the context of WEX toolbox, a database schema is a set of tables where each table corresponds to some protocol in the TCP/IP protocol stack.

WiPal [57] is relatively new software tool dedicated to IEEE 802.11 traces manipulation. A distinctive feature of WiPal is its merging tool, which enables merging multiple wireless traces into a unique global trace. This tool works offline on pcap traces that do not need to be synchronized. WiPal also provides statistics extraction and anonymization tools. Its key features are flexibility, and efficiency. WiPal features a wipal-stats command that analyzes a trace and outputs several statistics. As an example, one may use WiPal to extract the following information: total number of frames, number of occurrences for each frame type/subtype, traffic size for each frame type/subtype, estimated number of missed frames, available networks (SSIDs...
and BSSIDs), and network activity w.r.t. elapsed time. WiPal includes a trace merger, as well as several related tools. One may call the *wipal-merge* command on an arbitrary number of IEEE 802.11 PCAP traces and get a merged PCAP trace as result. Experiments show that WiPal is an order of magnitude faster than tools providing the same features [57]. In the context of WEX toolbox, WiPal is potential useful addition. Furthermore, it is possible to merge WiPal and CrunchXML to create a better trace manipulation tool.

The tools presented above are special-purpose and each of them tackles a specific experimentation problem. Generally, it is easy to develop and maintain specialized tools because it costs less time and money. Those working on a specific area have better understanding and are expected to propose better solutions. Requirements and cost-analysis of developing a benchmarking platform convinced us to exploit existing tools to the maximum in our toolbox as explained in Chapter 5.

### 2.3.3 Data archiving and management

In networking research, **archiving and management** of **measurements and metadata** using well-designed **data repositories** is a major corner stone for scientifically rigorous experimental research. **Metadata** is additional information about wireless trace data that can enable others to reconstruct an experiment and reenact the analysis steps leading to the reported results.

Data repositories offers tremendous benefits. They encourage collaborative research by making data and tools accessible to broader research community. They extend the value of measurements beyond a certain case study. They make it easier for others to revisit the analysis or calculate additional metrics without repeating the entire experiment(s) when existing traces can serve the purpose.

An interesting data repository is MOME (Monitoring and Measurement Cluster) measurement repository. It aims at collecting information about measurement data captured by different research projects. It is aimed at co-ordinating activities in the field of IP monitoring and measurement. The stored data includes the description of the assumed measurement scenario and environment as well as the results of data analysis. MOME Data Analysis Workstation allows for performing selected analysis tasks and storing the results directly in the MOME repository. The problem with MOME repository is that it stored only references to measurement traces which are expected to be hosted somewhere else by the researchers who captured the trace. In the longer run, the database might end up storing a lot of broken URLs. Most importantly, it not is not tailored for wireless measurement traces and does not employ tools for wireless network analysis.
DatCat [54] developed and run by CAIDA (The Cooperative Association for Internet Data Analysis) is an Internet Measurement Data Catalog (IMDC), a searchable database of information about network measurement datasets. It serves the network research community by offering three key features. First, it serves as a shared global database where anyone can find the data needed for network analysis. Second, instead of relying on the data contributor alone for documentation, it allows any researcher to annotate datasets with problems, features, or missing information they discover in the data thereby increasing the utility of the datasets. Third, by citing datasets (using IMDC handle) from DatCat in their published results, researchers can make it easier for peers to validate their results or perform alternate analysis on the same data. Note that DatCat stores only descriptions of the data and instructions to obtain it. It helps only with the first step: finding the data. DatCat is only a metadata repository. The data itself resides with the contributor. It allows researchers to store description and download information about their measurement data. It does not enforce any control over format and quality of traces and relies on user feedback to identify the problems. Like MOME repository, it is also not tailored for wireless networks and does not meet the requirements of benchmarking. It assumes that metadata repository can enable reproducibility, however, there is no guidance on how to achieve it. Limited reproducible analysis might be possible, but reproducing an entire experiment is a different story.

CRAWDAD [56] is a community resource at Dartmouth for archiving wireless trace data and tools from many contributing sources and locations. Trace data corresponding to a particular study with some temporal locality (i.e., without a long time gap) is considered as one datasets. A structured description for each dataset is provided in the form of metadata. Metadata provides four pieces of information: data (i.e., packet traces), tools, authors and papers. Tools contributed by the research community are categorized as collection, sanitization, processing and analysis tools. Traces collected using syslog, snmp, netflow and pcap format are supported. Sanitization is performed to reduce the risks to user confidentiality by anonymizing MAC addresses, IP addresses and AP hostnames. A specialized tool called AnonTool [106] is used for the purpose. Processing programs are used to extract from raw traces only the parts interesting for the research purpose. Notable processing tools are snmp-parser (a tool for processing snmp traces), WiPal (a specialized tool for IEEE 802.11 trace manipulation) [57] and pcapsync (a tool to time-synchronize pcap traces) [107]. Analysis tools are used to plot trace statistics. Interesting analysis tools are WScout(a tool to visualize huge traces (>10 GB)) [108], wifidelity(a tool to show completeness of wireless traces i.e., the fraction of transmitted packets caught by the monitor) [109], etc.

CRAWDAD metadata does not target traces collected from real (i.e., production) wireless networks. It focuses on reusability of traces rather than reproducibility. User is desired to download the traces on her machine and do the processing and analysis locally. It has two
drawbacks. First, current metadata is not sufficient to describe an experiment. Second, it does not provide the tools to support large experimentation campaigns consisting of hundreds of runs and detailed result reports.

In [79], authors have proposed a data repository for wireless experiments. Their approach seems to be promising, however, it is yet to materialize.

2.4 Requirements for Protocol Benchmarking

Nowadays, it is possible to setup a wireless experimentation testbed using commodity off-the-shelf hardware. However, wireless experimentation itself is not straightforward. Realizing fair evaluation and comparison is challenging. To overcome these challenges and bring scientific rigor to the results, networking community needs guidelines and tools. We propose benchmarking as an answer to these challenges.

Benchmarking is not an unfamiliar term at all. It is used in almost every walk of life where performance, efficiency and competitiveness matters. We invoke this term in the context of wireless networks to emphasize fair evaluation and fair comparison of protocols through experimentation.

A fair evaluation can be conducted by repeating the experiment several times to rule out the prospects of first results being just an accident. Then, we should group runs according to the similarity of conditions and analyze the protocol performance on per group basis. The approach can be extended to fair comparison of similar protocols or different versions of the same protocol.

To realize benchmarking in wireless networks, we have established following requirements:

Beware of Pitfalls Wireless experimenters, especially newcomers, can fall into the trap of numerous pitfalls. Therefore, wireless benchmarks need to describe the pitfalls and help others not to repeat mistakes of the past. Some pitfalls arise from the calibration issues with tools. Others arise from wrong mental model of the reality. Networking people often lack deep understanding of radio propagation and antenna theory. It is therefore, required to document pitfalls and facilitate experimenters to avoid them through sanity checks where possible. We can also formulate test cases to guard against pitfalls. See Chapter 6 for details.

Follow a Methodology We need a scientifically rigorous experimentation methodology. The methodology should be independent of any experimentation platform and easy to follow by others. Other than providing a detailed step-by-step account of the workflow of tasks, it should facilitate fair comparison of protocols. It should provide a roadmap from experiment definition
to *data repository* to *full disclosure reports*. We propose a benchmarking methodology in Chapter 4.

**Employ the right Tools**  Each step of the benchmarking methodology performs a bunch of tasks. Some of the tasks can be accomplished using existing tools, others need enhancements in the existing tools and yet others need brand new tools. We have developed a toolbox to implement the methodology as elaborated in Chapter 5.
3

PITFALLS OF WIRELESS EXPERIMENTATION

Wireless experimentation has numerous pitfalls. It is easy for the unwary experimenters to fall victim to them and draw wrong conclusions from the measurements. Therefore, in this chapter, we will report some pitfalls that were encountered during this research. The list of pitfalls reported is not exhaustive, rather it is intended to help others avoid mistakes of the past and contribute by documenting the pitfalls that they have uncovered. It is encouraged to archive the pitfalls in the trace data repository and share with the community.
3.1 Avoiding Pitfalls

Some pitfalls arise from wrong calibrations of the hardware and software tools. Others arise from wrong mental model of the reality. ‘Networking’ people often lack deep understanding of radio propagation and antenna theory, therefore, they are vulnerable to them. Some wireless experimentation pitfalls are briefly discussed as follows:

3.1.1 Antennas and sniffers can miss packets

It is known that antennas do not pick every transmission [122]. This can be more noticeable for off-the-shelf wireless cards. There can be various reasons. For example, these cards are not designed to serve as high accuracy ‘test instrument’ and trace collection hardware and software may be under provisioned. We observed anomalous packet drops frequently with Atheros Wireless cards and TShark on our probes. Note that application data rate is always constrained by physical bit rate. If PHY rate is set to 1 Mbps at the source, the effective transfer rate will always be less than 1 Mbps no matter what rate the application is trying to send data at.

At the probes (or receivers), goodput can be further less because antennas can miss packets and sniffers can drop (potentially large number of) packets without warning. There isn’t much that we can do to alleviate packet misses by antennas other than increasing the number of probes in the test environment. However, it is very important to calibrate the sniffer. Problems with the sniffer may not be discernible at low data rates. Therefore, experimenter should design sniffer test scenarios wherein sniffer is subjected to various data rates at varying distances from the source. Other than checking the throughput, difference between the number of packets transmitted and the number of packets received as well as tallying the total number of data packets received at the probes, will make it more obvious whether sniffer is dropping packets or not. In our measurement studies, abnormal packet drops by TShark became more obvious at higher data rates as shown in Figure 3.1. We solved the problem by replacing TShark with TCPDump which is light-weight and more efficient.

To avoid undesirable packet drops (caused by under provisioned hardware/software), it is recommended to use multiple probes (with good processing power) placed at various locations in the test environment, choose adequate wireless cards and employ lightweight sniffers.
### 3.1 Avoiding Pitfalls

#### 3.1.2 Improper calibration of the sniffer

Sniffer configurations can influence the ability of the sniffer to capture packets reliably. For example, if a sniffer, say TCPDump, is configured to capture packets with full payload, it is more likely to drop packets. Experimenter may never suspect the sniffer if packet drops by it are not huge. During our initial experiments, we noticed large packet drops at regular intervals even with the lightweight TCPDump as shown in Figure 3.2. However, one needs to be watchful as packet drops may not be obvious at the first sight.

```
It is important to calibrate the sniffer so that it does accurately what is desired of it. Ensuring sane working of the sniffer will ensure that packet loss or goodput calculations are accurate.
```

#### 3.1.3 Traffic generators can be buggy

Bugs or improper configurations cause traffic generators to misbehave. Take Iperf [84] for example. It is a well-known traffic generator often used as throughput testing tool. For UDP
3.1.4 Multipath fading

Small changes in the distance between transceiver stations can have a big impact on the signal stability at the receiver. Same happens when orientation of the receiver or the transmitter changes even slightly. The phenomena is known as multipath fading. As the geometric relationship between the transmitter and the receiver changes, depth of fading at the receiver also changes. The received signal envelope either becomes less stable or more stable. We estimated multipath fading using ricean $K$ against two test cases. First, the receiver was incrementally moved 4 times by 1 foot in the same direction and $K$ factor was estimated time by executing 5 runs. The result is shown in Figure 3.4. Second, receiver was rotated clockwise 4 times by 90°.
3.1 Avoiding Pitfalls

Five runs were conducted against each orientation. The resulting K factor estimates are shown in Figure 3.3. As multipath fading can vary significantly with a small change in the placement or the orientation of stations, network performance will also vary and, if this impact is ignored, comparability of the results will become difficult.

![Figure 3.3: Anomalous received power as measured UDP traffic generated by IPerf](image)

It is important to control the placement of nodes with great care. This would be error prone if done manually. Machine controlled displacement or rotation of nodes are desirable. Otherwise, experimenter MUST estimate ricean K factor for any meaningful comparison.

3.1.5 Channel interference

Channel interference in 2.4 GHz is an ever increasing phenomena. There is no escape from interference caused by other WiFi networks, devices operating in 2.4 GHz band and tropospheric ducting, etc., 'in the wild' (non-shielded environment). A glimpse of RF landscape across 2.4 GHz is shown in Figure 3.6. More interference means more channel utilization (or power spectral density or less availability of time slots). As consequence, network performance will suffer because of increased packet loss.

The solution is to conduct experiments either in the shielded environment or use a reasonable spectrum analyzer to record channel utilization or power spectral density (PSD).
Chapter 3: Pitfalls of Wireless Experimentation

3.1.6 Multiple antennas

Spatial diversity is a technique where multiple antennas are used at the transmit/receive ends, in order to improve signal to noise ratio (SNR) and throughput. Wireless card drivers such as MADWiFi have diversity capabilities enabled by default. This results in unexpected changes in the received power values due to a change in the antenna used, caused by a switched diversity algorithm \[114\]. According to this algorithm, only one antenna is chosen at any given time. The switch of antenna occurs when the perceived link quality falls below a certain threshold \[114\]. This switch may even occur when there is only one antenna on the cards, resulting in a second phantom antenna \[115\]. When a single antenna is used, the captured signal strength indicator (SSI) values must follow a statistical distribution from those reported in \[111\]. The empirical and theoretical probability density functions (pdf) of the received power should be compared. When a switched diversity algorithm is used, fictitious power fluctuations can be observed. Indeed, since there is no second antenna, this algorithm generates attenuated versions of the power received by the only antenna, and the received power values reflect a strange behavior. This can cause significant errors in the interpretation of data, which is even more critical when using adaptive rate algorithms dependent on the RSSI, SSI, or SNR. An example of this phenomenon is shown in Figure 3.7, observing two virtual pdfs instead of one. This phenomenon is also mentioned in \[114\]. It is worth noting that the data is being transmitted from a single source antenna; if both transmit and receive diversity were enabled,
3.1 Avoiding Pitfalls

Figure 3.5: Four Orientations: Multipath fading experienced by a node changes if it is rotated

during the day. A straightforward analysis of the histogram would also reveal more than a single pdf.

3.1.7 Power control

Power control algorithms can generate very different results according to the type of card used. Actually, many power control solutions are not efficiently implemented in 802.11-based chipsets, and there are only a few cards where they operate properly [113]. When performing measurements, a calibration procedure should reveal if the algorithm is working properly. In case of anomaly detected, it should be switched off.

3.1.8 Common problems with packet transmissions

We conducted a large number of wireless experiments with different transmission power values, packet sizes, data rates and distances using different packet sniffer to estimate the transmitted and received number of packets. Depending on network and local system load, the transmit or receive end can silently drop packets without leaving any trace about possible reasons for packet losses. In our case, this was solved by changing the size of the buffer of the acquisition software. However, this may also be related to external factors and special care should be taken when performing measurements in order to avoid packet losses not related to collisions or to wireless channel instabilities.

On the other hand, packet injection is an important mechanism for research and analysis of Wi-Fi networks, especially security aspects [112]. For instance, we found MadWifi driver to
be performing 11 retries at MAC layer for each and every packet although the retry attribute was turned off when configuring the wireless interface. The problem was noticed by examining the sequence numbers embedded in the injected packets after reception at the probes. For our specific case, we solved the problem by modifying the driver by setting the retry value to 1 when the interface is operating in monitor mode.

### 3.2 General Advice

Experimenter should investigate the literature for reported problems or abnormal situations that contradict the expected results from theory. Sanity checks can help greatly to avoid some of the pitfalls. A sanity check is a set of tasks to ensure that the hardware and software behave as expected. In particular, we should pay attention to the following points:

**Time synchronization:** Time synchronization is relevant to perform simultaneous measurements at different locations. The experimenter needs to ensure time synchronization up to the desired granularity, using NTP or another time synchronization protocol.

**Antenna diversity:** When enabled, it causes the driver to choose the most convenient antenna for reception. This might cause sudden rise or fall in the received power. It should be clear to the experimenter whether antenna diversity is enabled or not for a particular wireless scenario and its impact on the metrics.
3.2 General Advice

Figure 3.7: Two distinct power levels observed at the receiver when wireless NIC has two radio antennas and antenna diversity is enabled

Packet sniffing: Verifying accuracy of the packet sniffer is very important. Sniffer when used with wireless NICs can silently drop packets mainly because of under-provisioned hardware/software. This can lead to wrong interpretation of the results.
In this chapter, we provide a general purpose benchmarking methodology for scientifically rigorous wireless experimentation. Benchmarking requires provisioning of a level playing ground for fair apples to apples comparison of protocols and applications relative to a reference evaluation. The seemingly simple task of benchmarking is surprisingly complex in wireless networks because it requires an in-field evaluation of the system to ensure real world operational conditions. The complexity of such an evaluation is compounded by the lack of control on experimental conditions and lack of tools. Given the distributed nature of a testbed prepared using real hardware (wireless cards, network equipment, etc.) and real software (network stack, drivers, OS) in the real wireless environment, implementing a wireless benchmarking methodology can be challenging. It requires know-how of not only computer networks but also distributed computing, antenna theory, EM radio signal propagation, signal processing, and advanced (sometimes drive-level) programing skills.

The purpose is to propose a benchmarking methodology which could be applied using existing testbeds. However, part of the methodology addresses testbed setup issues as well.

In the following sections, we provide a detailed step-wise account of our proposed methodology for wireless experimentation and benchmarking, and present our benchmarking framework in Chapter 5.
4.1 Food for Thought

Identifying, improving and deploying superior communication standards and protocols adds to the business value of a network and benchmarking provides the foundations. Some of the reasons that benchmarking is becoming more and more promising are outlined as follows:

Benchmarks can be published and used by interested research groups who then can contribute with relevant metrics, models and test scenarios. They enable increased reproducibility of results and provide a common base for fair and consistent comparisons. Standardized workloads, run rules and benchmark tools can speed up the performance estimation and hence organization's ability to make improvements in a more efficient way. The use of different metrics, test scenarios and measurement methodologies complicates comparison. Benchmarks help overcome these problems and promote healthy competition.

Benchmarking helps identify areas of cost reduction, enables a more detailed examination of efficiency and facilitates value-add. Benchmarks are also used to prepare proposals for product selection and system development. They can be employed to investigate how well an initial installation is performing. It is helpful in debugging a given configuration, determining where additional equipment needs to be installed and it can go a long way in providing most cost effective and functional installation in a given environment.

Basically, benchmarking is greatly useful in planning, testing and evaluating network performance. It is of great interest to engineering, marketing and executive level personnel. Staff can better prioritize which network problem needs to be addressed and "how good is good enough?"

4.2 Background

Benchmarking is a well known concept in many domains of computing and natural sciences. It is often applied to measure quantitative advantage of one system over another similar system. More precisely, benchmarking is evaluation of the performance of a system relative to a reference performance measure. It can be applied to virtually any system (business processes, progress reviews, software/hardware tools, protocols, etc.) which exhibits quantifiable performance indicators. It is proven means of improving performance, efficiency, cost-savings and competitiveness of a system [1]. It facilitates "learning from the experiences of others" and thus enables the identification, adaptation and deployment of protocols that produce the best results. However, the potential of benchmarking hasn't yet been utilized in networking and particularly in wireless networks. If realized, benchmarking can go a long way in alleviating the critical issues such as data rate enhancements, cost minimization, and user security in future wireless networks.
4.3 Major Steps

In the field of network computing, benchmarking is common for network interconnection devices [3]. Recently, IEEE LAN/MAN Standards Committee prepared a recommended practice for the evaluation of 802.11 wireless networks [4]. Recommendations in [4] are valuable for preparing the target test environment and can be followed in conjunction with benchmarking methodology presented herein.

Figure 4.1 outlines the set of activities envisioned for benchmarking in wireless networks. The first step is to prepare Terms of Reference (TOR) or Plan. Other steps involve research on state of the art, design and development of benchmarking tools, setting up network management and experiment control tools. Then there is a cycle of activities as shown by circulating arrows in Figure 4.1. The cycle should be repeated for initial smoke runs in order to establish the precision and logic of results prior to the running of actual benchmarks. The cycle may also be repeated in order to achieve a certain level of confidence in the results. The activities include configuration of target test environment, performing the experiment, and undertaking measurements and data collection, analyzing and producing results, managing data-centric activities (such as storage, security, sharing, etc.) and preparing and publishing benchmark reports.

4.3.1 Plan: Terms of Reference

This step forms the basis of benchmarking. It sets goals and provides motivation for the undertaking. Type of the network (e.g., WiFi, WiMAX, Bluetooth, GPRS, IrDA etc), area of focus (e.g., wireless application such as peer-to-peer video streaming, content sharing), scope of measurements (i.e., set of metrics), and target deployment environment (indoor, outdoor, etc.) are key considerations. Priority should be given to the area that has greater value to the users or area that consumes larger resources. It has to be decided what should be the key indicators or metrics based on their importance and major aspects of cost. Terminology in the context of area of focus has to be defined so as to avoid confusing connotations. Planning for key benchmarking tasks such as network setup, cluster setup, benchmarking tools (e.g., traffic generators, sniffers, etc.), trace and meta-data collection, preprocessing, statistical analysis, reporting etc., has to be done.

A set of deliverables which conform to the requirements, scope and constraints set out in planning has to be listed and elaborated. A documentation or data management system has to be developed. Terms of reference are subject to change during the process of benchmarking as a consequence of change in high level requirements or system artifacts that may become clear later on.
4.3.2 Investigate: Existing best practices

Research on the current state of benchmarking and evaluation paradigms across domains relevant to benchmarking is constructive so that benchmarkers can bring them up to speed with the advances, avoid re-inventing the wheel, be able to make use of the existing knowledge-base of best practices and software tools, and start off from where peer benchmarkers have left. One needs to develop a comfortable understanding of the underlying wireless standards. It is imperative to investigate selection of metrics, run rules, baseline and peak configurations (if any), limitations, risks etc.

For instance if one were interested in improving handoffs in wireless networks, she would try to identify other domains that also have handoff challenges. These could include air traffic control, cell phone switching between towers etc. Typical aspects/metrics to consider would include (but not limited to) handoff costs, efficiency, delay, errors, and QoS. Benchmarking of handoffs is critical because depending on the application scenario, failed or wrong handoffs can result in enormous cost or have disastrous consequences.
4.3 Major Steps

4.3.3 Engineer: Benchmark tools

An ensemble of software tools (benchmarking toolbox) is at the core of any benchmarking endeavor. Before delving into the development of such tools, it is very useful to explore existing tools that serve a similar purpose. The golden rule here is to avoid re-inventing the wheel and re-use the existing code where possible in order to cut down the development cost. Benchmarking tools are desired to evolve as a result of bug-fixes, functional enhancements, re-factoring, re-engineering, etc. An agile development approach would be suitable wherein some of the benchmark tools would be implemented or adopted based on their priority. Sometimes adjustments are required to account for new understanding gained through mistakes during the course of benchmarking.

For example, consider wireless mesh networks. Wireless meshes normally facilitate broadband applications with various QoS requirements. Suppose in order to test the mesh network’s ability to carry large amounts of traffic, say in video surveillance of a metropolis, capacity planning might take precedence over security planning. In this case, we can put benchmarking of security or other qualitative aspects on hold and concentrate on what is more important: throughput.

It is more productive to embed the functional testing or unit testing (in vitro in most cases) within the development process. Indeed, this allows rapid enhancements and re-factoring while ensuring that the core functionality remains intact. A system documentation is necessary to describe the functionality, limitations, direction for future enhancements, dependencies and installation guidelines for the deliverable tools.

4.3.4 Deploy: Resource and Experiment control

The pre-requisite for benchmark tool suite deployment is setting up a computer network in the target test environment such that it meets all the mandatory software and hardware requirements laid out in the test specification. Typical test environments include calibrated over the air test (COAT) environment, conducted test environment, over the air (OTA) outdoor Line of sight (LOS) environment, OTA indoor LOS environment, OTA indoor non-line of sight (NLOS) environment and OTA shielded enclosure environment. Deployment involves setting in place the network equipment and installing the required software. Setting up of a computing cluster is also desirable in order to manage the execution of experimental tasks on the set of nodes participating in the experiment. It also empowers the benchmarker to perform multiple runs faster and efficiently. It is very imperative to have the network equipment calibrated and all the benchmark software tested. It is good practice to use latest versions of firmware and drivers for all the wireless products. Products are normally shipped with default optimal settings. The decision whether to use baseline configurations
or peak configurations or any other custom settings must be carefully considered but security settings might have to be adjusted anyway. Whatever settings are used must be carefully documented along with hardware models.

All of the required protocols will be configured and enabled during the setup. Parameters and settings associated with the devices and applications running thereon that affect the performance will have to be listed. Then, within this list, those parameters and settings that will vary during the experimentation have to be identified so that they can be included in the sampling process of network measurement. For example CPU usage, memory usage, swap usage, interference, etc.

We need to document all relevant configurations regarding devices (OS kernel, CPU, memory, etc.), tools (sniffers, spectrum analyzers, etc.), network (security usage, (TX, RX) signal levels, RTS/CTS usage, etc.), number of senders/receivers, etc. Key to successful benchmarking is holding as many parameters as possible constant in order to isolate the contribution of specific elements being compared.

4.3.5 Configure: Wireless experimentation scenario

Configurations elaborated in section 4.3.4 are general and are concerned with the network resources and experiment cluster setup. All of the benchmark tests would have to be run without changing the general configuration/setup of the devices in anyway other than that required for the specific test scenario. In this step, all the tools necessary to carry out the tasks specified in the experimentation scenario have to be calibrated. Nodes participating in the experiment such as source(s), receiver(s), probes and spectrum analyzers have to be configured. This is usually repeated for each run of the experiment to ensure a clean start.

4.3.6 Experiment: Undertake experiment execution and data collection

Multiple independent applications, such as data and streaming media, should be run and behavior of the wireless network should be measured according to the test specifications using a suitable sampling rate [11]. Applications should be representative of the real world situation and capable of associating with the wired/wireless interfaces of the devices and generating traffic at the desired rate. Benchmarkee should be tested under different frame sizes especially max and min legitimate frame sizes and enough sizes in between to get a full characterization of its performance [3].

Workload tools of the benchmark toolbox are expected to produce normal workload as well as fault-load. Fault-load represents stressful conditions to emulate real faults that are experienced in the real systems. Appropriate level of detail about the workload is important in order to make meaningful analysis. Network load characteristics along with extraneous and
internal RF interference should be measured. Network variations such as link failures and congestions have to be reported. Meta data about the result elements (e.g., traffic samples, RF interference samples) and configuration elements (e.g., network settings and parameters) would aid in keeping track of the context in which experiment was performed. It is also important to structure the chain of steps between launch and termination of the experiment and maintain version control of the participating scripts. Employing visual tools to explore in-consistencies and changes in scenario definitions of the subsequent runs can result in big payoffs.

Performing network measurements is a complex process. Precision and accuracy of measurement devices and tools has to be documented. It must be clear to the benchmarkers as to whether they are measuring what they actually wish to measure. A general strategy would be to gather more than one type of data set - either from a different location in the network or from a different time [14]. Measurement data needs to be collected using open industry-standard formats. Collection of meta-data, even if its immediate benefit is not obvious, may become useful in future benchmarking practice. It can be extremely helpful to seek out early peer review of proposed measurement effort.

4.3.7 Preprocess: Data cleansing, archiving and transformation

The data collected in the experiment execution stage needs to be cleansed. This could be achieved by employing self-consistency checks and investigating outliers and spikes. The first question to ask would be if the experiment was performed all right. Validity and integrity of measured data has to be assessed. Traces collected using a sniffer may contain significant amount of exogenous traffic. In order to reduce transformation and processing time, it may be desirable to filter out irrelevant data before transformations and analysis. We need tools to verify that measurements indeed provide a true picture of wireless network. One approach would be to create 802.11 finite state machines, look for inconsistencies and come up with an executive summary on the quality of measurements. If the measurements lack the desired level of integrity and validity, it would be required to repeat the experiment with better experience and improvements in the tools gained in previous measurement cycles.

Conducting benchmarking is costly in terms of time and resources. This, therefore, necessitates persistent structured storage of measured data using standard data formats such as XML [55], XBRL [19] and database management systems (DBMS). One such example is CRAWDAD [21]. Meta data (interference, variable resources, etc.) should also be associated with the traces.
4.3.8 Analyze: System performance

Finally, data would be processed to produce the results which represent the values of metrics as specified in the test specifications. For some metrics, data has to be transformed (normalized or converted to different scale) to fit the analysis needs. Effort should be made to minimize the generation of intermediate data between raw measurements and final results. Instead caches can be used for transient intermediate results. This would aid in reproducing the same analysis [14]. Calculation of mean (arithmetic or geometric) behavior over same measurements performed in different ways can provide a good insight of the network performance. Confidence intervals and distributions can also be used to depict the network behavior.

The whole chain of analysis must be documented. Versioning and storage of analysis scripts along with the measured data that underpins the results should be stored. We need to archive both measurement traces and benchmark results. Benchmark results are either obtained through internal benchmarking effort or from partner research groups or organizations. Versioning mechanism has to be employed to facilitate reproducible analysis.

As described before, real-world wireless experiments are not repeatable. Despite this obstacle, however, it is possible to enable peer verifiable comparison of networking protocols. To this end, we propose to conduct large number of experiment runs and group them into clusters based on the similarity of experiment conditions at the end. Comparing the performance of a protocol against matching clusters in subsequent evaluations will provide apples to apples and hence trustable performance comparison.

4.3.9 Report: Benchmarking score

Reports are the windows through which benchmarkers [12] can gain a visual access to the results. They provide detailed insight into the strengths and weaknesses of benchmarkee. All the benchmark-related information which is complimentary to the results must be made available. Meta-data (e.g., precision of tools, accuracy of measurements, etc.) which could be useful for trouble-shooting, decision-making, and management action, should also be reported. Reports should include an executive summary consisting of comprehensive graphs, configured parameters and drill-down details (if any). In fact, reports have to be designed and presented in accordance with the full disclosure report (FDR). Full disclosure report, for each benchmark, is prepared in accordance with reporting rules. Producing and interpreting benchmark results crosses into the realm between art and science. Web services are a great way to provide access to the database of benchmark results. Web services, then, can be used by interested organizations and groups to gain access to the results. Web services will also enable distributed access and sharing in the form of web reports.
4.4 Benchmarking Methodology in a Nutshell

Figure 4.2 illustrates the flow of events in a typical benchmarking process. Each step of the process is annotated with tasks that should be accomplished before proceeding to the next step.
Chapter 4: Benchmarking Methodology

Figure 4.2: An instance of wireless benchmarking process
BENCHMARKING FRAMEWORK

As mentioned before, experimentation is increasingly being employed for the evaluation of wireless networking protocols. However, rigorous wireless experimentation is a not a trivial undertaking. Apart from the complexities introduced by the wireless channel, setting up testbeds, managing configurations, conducting large number of experiments and orchestrating workflow of experiment tasks is very complex. It requires a diverse set of programming and trouble-shooting expertise on the part of experimenters. Time needed to accomplish an experimentation campaign also discourages many from venturing into the arena. That is why majority of the networking papers still rely on simulation-based approach.

After analyzing existing experimentation-based protocol evaluation approaches, we have implemented a toolbox, called WEX Toolbox, to ease the burden on the researcher and make it feasible to carry out large experimentation campaigns. In the context of this thesis, henceforth the toolbox will also be referred to as WEX Benchmarking Framework.

Note that WEX benchmarking framework [32] is a brand new toolbox developed and put together during the course of this dissertation. It has been designed to accomplish the tasks required at each step of the benchmarking methodology presented in Chapter 4.

At the core of WEX Toolbox design is reusability of third-party open-source tools. In that respect, along with some other tools, as explained later, CrunchXml [42] (a utility for loading packet traces into MySQL database) was also reused from legacy WexTool [123] with slight modifications.

In the rest of this chapter, we discuss various aspects of WEX Benchmarking framework such
as key features, design, deployment, usage, performance, etc.

5.1 Objective

WEX toolbox is designed to serve two main objectives as follows:

1. **Facilitate rigorous wireless experimentation**: Experiments are considered to be rigorous if consistent results are obtained from multiple runs of an experiment in time and space.

2. **Implement a benchmarking methodology**: This is to provide tools that make it easier to follow the benchmarking methodology explained in Chapter 4. This will allow wireless experiments which are scientifically rigorous and comparable.

   Benchmarking methodology elaborated in Chapter 4 has two dimensions. Firstly, it addresses resource control (i.e., testbed setup and management). Secondly, it elaborates experiment management (i.e., experiment description, measurements, analysis and reporting). In this chapter, we will mainly focus on experiment management tools. Experiment management functionality of the framework corresponds to the steps represented by circular arrows in Figure 4.1.
5.2 Key Features

In order to support the above objectives, WEX Toolbox offers several key features as follows:

- **Ease and speed**: Wireless experimentation is made easier by greatly reducing the manual effort. This enables the experimenter to conduct large number of experiments in a short period of time. It increases her ability to investigate a research issue rigorously.

- **Control**: WEX Toolbox makes it easy to specify the topology, applications and card configurations, and to control the execution of the experiment.

- **Manageability**: Without appropriate tools, manageability of even a small number of nodes is cumbersome. WEX toolbox empowers the researcher with improved manageability of the wireless testbed.

- **Scheduling**: This entails being able to launch experiments with arbitrary number of runs at anytime. Coupled with remote access, this is a powerful feature to investigate network issues.

Moreover, WEX toolbox has some other cool functionality to offer to the experimenter. Experiment description (ED) is based on XML. XML is universally accepted standard for storing and transporting data. We selected XML for four reasons. First, it is simple. Second, it is easy to read and write. Thanks to large number of cool XML editors. Third, it is easy to process. Every notable language comes with XML parser(s) of its own. Fourth, it is straightforward to store in the database. All major database management systems offer rich support for XML storage and processing.

In WEX toolbox ED is simply a hierarchy of tasks. Each task is associated with a group of nodes. Nodes are assigned to groups according to their desired role in the scenario. A group can assume the role of probes, traffic senders, traffic receivers, spectrum analyzers, etc. An ED file has experiment as the root element. Then there is groups element which in turn contains many group elements. The tasks specified for a group will be executed on all group members. For details, please see section 5.4.1.

At the end of an experimentation campaign, traces can be reorganized per run, indexed and loaded into the database in one go. Creation of database schemas for each run, export of traces to XML and invocation of CrunchXML for each trace is automatic.

More than 80% of the analysis is carried out using SQL queries. Some advanced calculations such as confidence intervals, K factor estimation, throughput per probe, etc are performed using SQL.

We have proposed full disclosure reports (FDR) for key network performance metrics. We developed programs to make plots using mainly octave, matlab and GNU plot. In a nutshell,
WEx Toolbox \cite{32} facilitates automatic scheduling and workflow management of large experimentation campaigns spanning over weeks or even months. At the end of an experimentation campaign (involving possibly hundreds/thousands of experiments), an indexing mechanism facilitates parallel processing of all the traces. Furthermore, it allows a deep analysis of network performance by facilitating the calculation of bit errors, packet errors, packet loss as well as channel characterization (i.e., multipath fading, channel analysis and path loss).

5.3 Design and Architecture

As wireless experimentation encompasses a large number of tasks \cite{4}, the corresponding experimentation framework is usually a considerably complex distributed software. This warrants special attention and care to the design of the system otherwise management and maintenance of the platform itself would become burdensome. To make the life of an experimenter easier, we have strived to employ a highly modular approach in the design and architecture of WEX Toolbox. Resuability of third-party tools has been at core of the design. Notable third-party tools and technologies are Sun Grid Engine (SGE) \cite{49}, TCPDump \cite{44}, TShark \cite{45}, XML \cite{55}, wireless tools for linux \cite{36}, kismet spectrum tools \cite{86}, MyPLC \cite{82}, NFS, NTP, etc.

5.3.1 Architecture diagram

Figure 5.1 demonstrates high level architecture of WEX Toolbox. This architectural breakdown is intended to provide a better perspective of the working vis-a-vis design of the framework. Also for the sake of simplicity, testbed setup and cluster management modules are not shown. For the curious, details are available at \cite{82} \cite{49} \cite{32}.

Three distinct sets of modules are shown in the figure. First set of modules consist of experiment description language (EDL), EDL parser, experiment scheduler and task scheduler. EDL is in fact an XML file with reinforced structure over the format and layout. The purpose is to enable experimenters define an experiment using the all-familiar syntax of XML. Only a predefined set of element names and attributes is allowed. This provisions simple yet powerful way of defining all the elements of wireless scenario in sufficient detail \cite{5.4.1}.

An experiment is executed by launching the experiment scheduler module with experiment description (ED) as input. Experiment scheduler generates the specified number of runs. A run is simply a shell script annotated with the information such as execution host (i.e., server) and
the start time (or scheduled time) of the run. The run is submitted to the grid engine as soon as it is generated [5.4.1].

The run is kept waiting at the server till its scheduled execution time arrives. It is at this stage that the run invokes EDL parser on the input experiment description. EDL parser loads ED into a parse tree. The parse tree is technically an hierarchical representation of the entire experiment and is based on a combination of list and dictionary data structures. Once constructed, parse tree is traversed to generate the final set of tasks which is essentially a bunch of shell scripts. The tasks are, then, archived and an index called task index is produced. Generated tasks are shown in the tasks module. Task scheduler submits all these tasks (as identified by task index).

Second set of modules in WEX toolbox is concerned with management of experiment data. It includes management of traces, meta data and analysis scripts. It is assumed that all the traces reside at one location. Currently WEX toolbox collects all the traces at the experiment server and we copy them manually to our data server at the end of an experimentation campaign. Data collected during a wireless experimentation campaign can be huge. Identifying runs and data belonging to each of them is necessary for a meaningful analysis. Management modules are aimed at making the experiment data easily manageable and tractable. This is achieved by organizing data into data bins and loading traces in a MySQL database. An index file serves as a liaison between what resides in the data bins and what resides in the database. One data bin is maintained per run and there is one database schema per run. Database schemas are managed using a python program and a set of SQL scripts.

Once the data is organized into bins and schemas are created, the next step is to clean the traces of exogenous data using TShark [45] filters. TShark is also used to export packet traces to XML/PDML files. These XML files contain information about the decoded packets. Packets are loaded into the database using CrunchXML utility [42]. CrunchXML automatically performs merging and synchronization of the packets collected at different probes.

Third set of modules consists of analysis and reporting modules. Analysis is based on the selected metrics [4]. It includes calculation of average score as confidence intervals for each metric. Reporting is module is concerned with plots and Full Disclosure Reports (FDR) for each metric.

### 5.3.2 Control flow

Figure [5.2] is control flow diagram for experiment execution and corresponds to the activities performed by first set of modules shown in figure [5.1]. Experiment Scheduler is invoked with ED, experiment start time and desired no. of runs as inputs. If experiment description and start time are valid, experiment scheduler creates and schedules specified number of runs. When a run is launched, it invokes task scheduler to generate and schedule the tasks. This continues until all
Figure 5.1: WEX toolbox design
the runs have been scheduled.

Figure 5.3 demonstrates flow of control through the data management steps. The process begins when data management programs are launched. Path to the data repository serves as input to these programs. Exogenous packets are filtered out from the packet traces. This saves disk space and reduces the processing time. Data is organized into data bins. Each data bin corresponds to exactly one experiment run. An index is created over the runs which serves as a reference for all the runs. In addition to data bins, one database schema per run is also created. Index also serves as liaison between data bins and database schemas. After this, decoded packet from each packet trace are saved in a temporary XML file which is parsed and packet information is extracted and stored in the corresponding database. Traces collected from different probes are merged and synchronized using an efficient merging mechanism [42].

5.4 Detailed Description

We categorize the modules in WEX toolbox into server side and client side modules. Server side deals with provisioning control over the testbed resources. It is responsible for activities such as testbed management, experiment scheduling, experiment workflow management and data aggregation. Moreover, we have server side tools for analysis and reporting. Client side consists of stations which actually execute an experiment. Stations on the client side assigned different roles depending on the wireless scenario. Currently supported roles include receiver, sender, probe and spectrum analyzer.

In the following sections, we discuss both client and server side in more detail.
5.4.1 Server side modules

We further divide server side modules into two categories namely Indigenous modules and third-party modules. Indigenous modules are software components that were developed by us. Third-party modules are those open-source software components that were developed by other independent developers. We have in most cases integrated third-party modules into WEX toolbox without modifications. This merit of this component-oriented approach is that we do not need to worry about each and every component of the modules. One disadvantage of this approach is that not every third-party goes through the same level of quality assurance. However, the advantages far outweigh the disadvantages.

Indigenous modules include Experiment Description Language (EDL), EDL Parser, Experiment scheduler, Direct Packet Sender (DPS), data cataloguer, preprocessor scripts, analysis modules and report generators. Third-party modules include task scheduler (borrowed from sun grid engine [49]), network time synchronization protocol (NTP), network file system (nfs) protocol, TShark packet analyzer. In this section we shed light mostly on indigenous server side modules. Information about the third party modules can be found on their respective web pages.

**Experiment Description Language (EDL)**

XML is employed as the experiment description language (EDL). Experimenter can specify sanity checks, meta-data, experimental and control networks, node configurations and tasks.
5.4 Detailed Description

Figure 5.4: Component-oriented architecture of WEX toolbox
Tasks are orchestrated based on finite state machine (FSM) and each task is meant for a group of nodes where a group consists of one or more nodes. Currently supported groups are monitor, sender, receiver and spectrum. A sample XML file describing an experiment is provided in Appendix A.1.

EDL Parser

EDL parser is a python program which parses the experiment description (ED). It looks for predefined tags corresponding to network (both wired and wireless) configurations and groups of nodes representing senders, receivers, probes and spectrum analyzers. It also keeps track of sanity checks.

The goal of the EDL parser is to generate tasks from experiment description. Each task specifies a single well-defined ACTION to be executed at a particular station at the specified time for a specified duration. Tasks are described using Task Description Language (TDL) which is based on shell scripting [124].

Mapping from EDL to TDL consists of two steps. First, EDL parser constructs an in-memory parse tree for ED. Second, the parser traverses the parse tree, identifies experiment actions and generates corresponding tasks. The parser also annotates each task with grid (grid engine in our case) specific parameters.

Task scheduler

Tasks generated by EDL are annotated with information such as action, target station, start time, end time, etc. These annotated tasks are treated as jobs by the grid engine. When a job is submitted, it is kept in an holding area until it can be run. When the job is ready to be run, it is sent to the execution station. Grid engine manages the execution of jobs and logs the status of job execution. Holding of jobs is managed by a master queue at the server. All the execution stations have their own queue instances called execution queues. Master and execution queues can be configured to hold multiple jobs. Number of jobs that can be executed by an execution station is limited by the number of slots int the execution queue.

It is also imperative for a researcher to configure the scheduling settings of the grid engine. By default it checks the master queue for pending tasks every 10 seconds but this can be modified according to the requirements. This information is also useful for deciding the workflow of tasks when describing an experiment using EDL.

Experiment scheduler

Experiment scheduler provides a command-line interface for the user to submit experiments. It can schedule an experiment for execution at anytime. She can also tell the scheduler to repeat
the experiment any number of times.

Python code snippet in Appendix A.3 shows how different runs of an experiment are launched. The gap between successive runs is around 2 minutes which can be changed by the experimenter according to her requirements. It also specifies the ACTION to be performed which is invoking EDL parser on the scenario or experiment description (ed.xml) file. The code creates experiment tasks, one for each run, and schedules them at regular intervals. When an experiment task is executed, it spans several step tasks. One or more step tasks generally scheduled for execution at each step of the experiment execution. Execution time of experiment tasks is decided according to the description provided in ED file.

**Direct Packet Sender (DPS)**

The traffic generator also referred to as DPS (Direct Packet Sender) is based on packet injection mechanism implemented using BSD raw sockets. DPS is used to generate data packets with a payload of 100 bytes each. Packet are transmitted at the max rate possible. However the link bandwidth is set to 1 Mbps. This results in DPS transmitting at an effective rate of less than 1 Mbps. It is configured as follows:

- It takes three command line parameters namely network interface, time duration and seed for random bit pattern generation.
- First 8 bytes of the payload are reserved.
  - First 4 octets are set to all 1’s. This is to make MAC treat the packet as a normal packet.
  - The next 4 octets are used for sequence number.

All the bits in the payload (after first 8 octets) are set to 1 in order to calculate bit errors on the receivers by counting the flipped bits.

**Data cataloguer**

As it is clear from the title, the purpose of this module is to build and maintain a catalogue of all that matters for an experiment. Currently we use both database management system (DBMS) and Linux directories to archive traces, meta-data and scripts. There are several compelling reasons to employ data repositories to archive wireless experimentation data.

- Wireless experiments are expensive comparative to simulations in terms of time, effort and resources
Data repositories make it possible to revisit the findings of an experiment and facilitate peer verification of the results.

They will be crucial in making networking papers interactive. This will give the reviewers freedom over manipulating the data sets and enable them to look into results from such angles as possibly ignored by the researcher.

They facilitate rigorous wireless experiments.

They are useful in enabling apples to apples comparison and repeatability of wireless experiments.

At the end of an experimentation campaign, traces are collected at the data server. Initially everything, corresponding to potentially large number (say hundreds) of experiments and several probes, resides in one directory (one large bin). This module has a number of sub-modules which are as follows:

data organizers: It places data in separate data bins with one bin for each experiment. These bins are also called directory based archives. Only experiment specific scripts are stored in the bins. Generic catalogue, ETL and analysis scripts are stored in a directory separate from the data directory [Appendix A.4.1].

Indexer: It creates an index of all the experiments and maintains one index entry for each experiment [Appendix A.4.2].

DB schema manager: Database repository of traces consists of one separate database per experiment in a MySQL database management system. DB schema manager allows automatic creation of all the databases and corresponding DB schemas for each experiment. Furthermore, it facilitates modification of databases and their schemas. Note that index works as a liaison between directory based archives and database schemas [Appendix A.4.3].

ETL (Extract, Transform and Load) modules

These are a set of python programs and shell scripts. Exogenous data (i.e., traffic coming from exogenous access points) is filtered out and discarded using TShark filters. This reduces the storage size requirements and makes faster later operations on the data. We first export data from each packet trace file to a temporary XML file. This XML file is fed to CrunchXML which parses each packet, extracts a subset of fields from each protocol header and stores them in the database. There exists a separate database table for each protocol. Fields from each protocol header in the received packet are filtered out according to the fields specified in
the corresponding database table. This guarantees that what goes into the database is really needed.

It was observed that packet traces from different probes sometimes contain inconsistent MAC timestamps. Therefore, the inconsistent timestamps are also corrected by the ETL modules.

Piece of code in Appendix A.5 accomplishes the above task. CrunchXML is called behind the scenes to load data from XML tagged packet trace files in the database.

CrunchXML implements an efficient synchronization and merging algorithm, which takes XML (or PDML) input trace files generated by multiple probes, and stores only the packets fields that have been marked as relevant by the user in a MySQL database. Original pcap traces should be first formatted in XML using TShark/wireshark. These operations are done in a smart way to balance the CPU resources between the central server (where the database is created) and the different probes (PC stations where the capture traces are located).

Analysis modules

Analysis modules are an ensemble of python, C++, matlab programs and shell scripts. Our analysis philosophy revolves around calculating average results over large number of runs. We usually refer to the average result values as score. The score is calculated for the selected metrics. Currently there exists a rich support for metrics such as packet loss, bit error rate, packet error rate, goodput, ricean K factor, SNR/RSSI, received power and spectrum analysis. Sample code for calculating all these metrics is given in Appendix A.6.

The above scripts operate on individual experiments. There are other scripts which aggregate results over a set of experiments. They make it simple and easy to analyze the network behavior over a certain period of time (days, weeks, months). There are yet other scripts which are designed to calculate confidence intervals over aggregate average results.

Report generators

We aim at automatic generation of Full disclosure Reports (FDR) for each metric. FDR should be dynamically created through a web interface whenever user wants to see results for a particular report. The functionality is only partially supported at the moment. A typical FDR is available at [35].

5.4.2 Client side modules

We employ more third-party tools than indigenous tools on client side. These include network sniffer TCPDump [44], NFS (network file system) client, grid engine, spectrum analyzers, wireless tools from hp, SGE execution daemon, etc.
Chapter 5: Benchmarking Framework

**Spectrum Analyzer**

Kismet spectrum tools is a set of utilities for using various spectrum analyzer hardware. It supports the suite of Wi-Spy devices (original, 24x, 24x2, DBX, DBX2, 900, 24i) by Metageek LLC and the Ubertooth. We use Wi-Spy 24x portable USB spectrum analyzer which enables us to capture wireless landscape in 2.4 GHz band in the range [2400 MHz, 2483 MHz]. The range is in most cases mapped to the corresponding 14 WiFi channels. We modified kismet spectrum tools to associate timestamps with frequency-amplitude tuples and to change the format of measurement samples.

**MADWiFi wireless driver**

MADWiFi is open-source wireless driver which is widely used in wireless experimentation although its successor ath5k is also increasingly becoming commonplace. We instrumented the driver to fix a problem related to packet retransmission. The problem manifested itself when using packet injection.

**Wireless tools**

Wireless tools (WT) version 29 is used in conjunction with MADWiFi driver to configure senders, receivers and probes. WT provides textual user interface to facilitate configuration of wireless devices using Linux Wireless Extensions (WE). A sample script for configuring a probe using MADWiFi and WT is provide in Appendix A.2.

**5.4.3 Running an experiment: User’s perspective**

Wireless experimentation steps as supported by WEX toolbox are demonstrated in Figure 5.5. The first step is Experiment Description (ED). We use XML as the Experiment Description Language (EDL). However, experimenter is required to follow designated hierarchy and tags. For example, there are tags to specify sanity checks, define control and experimental networks. The nodes are treated as groups. For example, group of senders, group of receivers, group of probes, group of spectrum analyzers, etc.

In the second step, ED is fed to EDL Parser which is a set of python programs. Parser generates individual tasks to be executed on the specified nodes. A task is a shell script that specifies the action to be performed, start time of the action, duration of the action, execution node and log information.

The third step is the submission of tasks. Scheduler is responsible for scheduling tasks at the specified time and repeating an experiment desired number of times. Scheduler takes as input ED, start time and number of runs. In fact, user can just invoke scheduler on ED script
5.5 Deployment

WEX Toolbox is currently deployed on the top floor of three-story building. Most of the nodes are placed in a medium sized room. The room is not shielded from outside interference. A few nodes are also placed in neighboring rooms to provide for non-line of sight (NLOS) obstructions. The venue offers a typical office environment where many wireless production networks are operational 24/7 however level of interference from exogenous wireless networks varies with time. For example, We expect that 2.4 GHz ISM band would be more congested during office hours. Even during office hours level of congestion is different at different parts of the day. In order to have confidence in our measurements, we run large number experiments at different parts of the day. To keep track of interference, we employ spectrum analyzer.

5.6 Performance

We evaluate performance of WEX toolbox against number of experiments and level of automation of various experimentation tasks. The primary metric is the time (in hours) required to carry out the desired number of experiments. Duration for each experiment is 100 seconds. It is assumed that all the experiment tasks are carried out by a single human resource (i.e., the experimenter). The result is demonstrated in Figure 5.6.

Figure 5.5: User's perspective: Stages that go in between experiment description (ED) and task submission and scheduler will call EDL parser internally to generate and schedule corresponding tasks and will submit them to the Grid Engine.
Figure 5.6: Performance of an experimenter in terms of hours needed to run a given number of experiments

5.7 Conclusion

WEX Toolbox is built around a benchmarking methodology [4]. We used the toolbox extensively for experiments in indoor wireless environment. First, we conducted large number of exploratory experiments to uncover pitfalls and test calibration settings of software tools and devices. Second, we conducted hundreds of experiments to characterize wireless channel and investigate its impact on network performance. In Chapter [4] we present two benchmarking case studies carried out using the framework presented in this chapter.
Benchmarking provides a whole new perspective to the analysis by facilitating the identification of performance or benchmarking gaps. It makes easier the diagnosis of performance issues and provisions a better understanding on how to close the benchmarking gap. However, implementing a benchmarking methodology can be tricky. It took us more than one year to investigate experimentation issues and pitfalls, and to establish the benchmarking methodology proposed in Chapter 4. In this chapter, we present two case studies to demonstrate the methodology.

First case study focuses on wireless channel characterization and is aimed at providing step-by-step walk through the implementation of the methodology. Second case study focuses on multicast video streaming over WiFi and is aimed at investigating the impact of channel conditions on network performance through rigorous indoor experiments.
6.1 Case Study I: Wireless Channel Characterization

Channel characterization is a key step in wireless benchmarking. It enables researchers to record channel conditions and investigate their influence on wireless network performance. It allows them to understand how well a protocol or an application performs in different wireless environments.

The purpose of the case study is to study variations in the channel conditions at different points in time as well influence of receiver orientation. The impact of small changes in the receiver location is also investigated.

The following material is a step by step account of the benchmarking process. The material required to reproduce the results in the case study is available at [33] [34].

6.1.1 Plan: Terms of Reference

In this step, we lay out the foundations for the undertaking by giving a clear direction, setting a stage and provisioning the means to carry out the required benchmarking activities with efficiency. We select the wireless technology to be benchmarked, a small set of representative metrics (benchmark score), target deployment environment, resource requirements, set of tasks, output (final deliverable) and risks involved as detailed in Table 6.1.

6.1.2 Investigate: Existing best practices

We conducted an extensive analysis of the state of the art in wireless/wired networks as well as other computing and non-computing fields. The purpose was, in part, to understand the notion and utility of benchmarking in various fields. We also contemplated benchmarking jargon, concepts, practices, application scenarios and obtained interesting insight into this important paradigm which has been often undervalued in networking research. Computational benchmarks such as NPB (NAS Parallel benchmarks) [6], Standard Performance Evaluation Corporation (SPEC) [8] benchmarks, Transaction processing performance council (TPC) [7] benchmarks were looked into. Non-computational benchmarks especially those employed in the evaluation of business processes were also investigated. In fact, benchmarking forms a key component of business excellence models in today’s businesses as a means to measure competitiveness. Examples are Global Benchmarking Network (GBN) [62], European Foundation for Quality Management (EFQM) [61], etc. This investigation helped us polish our benchmarking terminology vis-a-vis fine-tune benchmarking methodology for wireless networks. We also investigated practices for wireless performance evaluation [4], development of metrics [11] [12],
Table 6.1: Planning for wireless benchmarking

<table>
<thead>
<tr>
<th>Activity</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of network</td>
<td>IEEE 802.11</td>
</tr>
<tr>
<td>Area of focus</td>
<td>Channel characterization</td>
</tr>
<tr>
<td>Benchmarkee</td>
<td>WiFi channel</td>
</tr>
<tr>
<td>Metrics</td>
<td>Co-channel and adjacent channel interference, K-Factor (Ricean fading model), received power, RSSI, packet error rate (PER), bit error rate (BER), packet loss</td>
</tr>
<tr>
<td>Deployment enviroment</td>
<td>Over-the-air (OTA) line-of-sight (LOS) or OTA non line-of-sight (NLOS) non-shielded indoor environment</td>
</tr>
<tr>
<td>Tasks</td>
<td>Network setup, cluster setup, experiment description (ED), scheduling, data collection, analysis, reporting, etc.</td>
</tr>
<tr>
<td>Resources</td>
<td>Hardware (Computers with PC slot, Atheros Wireless cards, Spectrum analyzers, High speed Ethernet switches, Database Server), Software (Madwifi driver, Traffic generators, sniffer, MySql DBMS, SUN Grid Engine (clustering software), Human (Benchmark facilitators, Network Administrator, Software developer, Benchmarker)</td>
</tr>
<tr>
<td>Cost</td>
<td>Cost of resources</td>
</tr>
<tr>
<td>Deliverables</td>
<td>Metrics, Benchmark score, full disclosure reports</td>
</tr>
<tr>
<td>Risks</td>
<td>Bugs in drivers/sniffers, limitation of spectrum analyzers, acquisition of resources</td>
</tr>
</tbody>
</table>

experimentation platforms and tools as listed in Table 6.2 which could facilitate wireless experimentation and hence aid in benchmarking. In the end, we selected some of the existing tools to incorporate them in our experimentation platform [32].

6.1.3 Engineer: Benchmark tools

After stepping into the wireless experimentation arena, we have explored and put to test a number of tools in order to gauge their functional suitability, precision and correctness. It became clear to us that we need to develop some new tools, instrument/enhance existing ones and harness them all together to achieve sound wireless experimentation and the larger objective of benchmarking in the end. Some of the existing tools such as TCPDump [44], tshark/wire-shark [45], sun grid engine (SGE) [49], served the purpose well apart from calibrations and fine tunings. TShark/WireShark suffer from performance degradation, in case of large trace
Table 6.2: State of the art: Literature and Tools

<table>
<thead>
<tr>
<th>Research</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understanding the benchmarkee</td>
<td>Everything You need to know about benchmarking [1], Draft Recommended Practice for the Evaluation of 802.11 Wireless Performance [4], Framework for Performance Metrics Development [11], Strategies for sound internet measurement [14], etc.</td>
</tr>
<tr>
<td>Tools</td>
<td>CrunchXML [42], Wireshark/Tshark [45], WiPal [57], Netdude [20], TCPCDump [44], Iperf, Netperf [10], tcp/nttcp/nuttcp [25, 26, 27], Nettest [24], DDoS benchmarks [28], MGEN, Kismet Spectrum tools [86], GNU Radio toolkit [43], etc.</td>
</tr>
<tr>
<td>Platforms</td>
<td>OMF [31], Emulab [59]</td>
</tr>
</tbody>
</table>

files, which could be overcome by employing more efficient tools such as WiPal [57]. In addition, we needed functionality to perform basic sanity checks, large scale scheduling, trace management, manipulation of wireless headers, (meta-)data management and post-processing, reporting, etc. Therefore, we indulged in the development of our own platform. Also, we developed a packet injector for traffic generation called Direct Packet Sender (DPS), based on BSD raw sockets [50], in order to inject packets directly at the link layer. Given in Table 6.3 is a list of tools that we brought together to build our wireless experimentation platform, henceforth, known as Wireless Experimentation (WEX) Toolbox [32].

Amongst the tools listed in Table 6.3 we modified MGEN, Madwifi and kismet spectrum tools as follows:

MGEN [51] was modified to customize the packet format. We stripped off unwanted fields from the payload other than the sequence number and the timestamp. Madwifi was instrumented to disable transmission of duplicate packets in case of packet injection. We customized the output format from kismet spectrum tools, associated timestamps with the frequency samples and inserted a code snippet to archive spectrum information.

The configurations of tools in our deployment of the platform are discussed below.

6.1.4 Deploy: Resource and experiment control

The experimental LAN and cluster was setup in indoor environment. The deployment details are demonstrated in [33].
Table 6.3: WEX Toolbox

<table>
<thead>
<tr>
<th>Function</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workload/Traffic generators</td>
<td>Direct Packet Sender (DPS) (New), Multi-Generator or MGEN (Instrumented), Iperf</td>
</tr>
<tr>
<td>WLAN device driver</td>
<td>Madwifi (Instrumented)</td>
</tr>
<tr>
<td>Spectrum analyzer</td>
<td>Kismet Spectrum Tools (Instrumented)</td>
</tr>
<tr>
<td>Sniffer</td>
<td>TCPDump</td>
</tr>
<tr>
<td>Packet Analyzer</td>
<td>Tshark, Wireshark</td>
</tr>
<tr>
<td>Sanity checks</td>
<td>Unit test suite (New)</td>
</tr>
<tr>
<td>Scheduler</td>
<td>SGE Scheduler, Scheduler support tools (New)</td>
</tr>
<tr>
<td>Content / data Management (CM)</td>
<td>Data Cataloguer (New)</td>
</tr>
<tr>
<td>Database schemas</td>
<td>DB schema manager (New)</td>
</tr>
<tr>
<td>Database Management System (DBMS)</td>
<td>MySQL</td>
</tr>
<tr>
<td>Merge / Synchronization</td>
<td>CrunchXML(New)</td>
</tr>
<tr>
<td>Extract, Transform and Load(ETL)</td>
<td>ETL Scripts (New)</td>
</tr>
<tr>
<td>WEX Cluster</td>
<td>SGE 6.2u2.5</td>
</tr>
</tbody>
</table>

**Resource Control Setup**

We setup a wired local area network (LAN) in order to manage experimentation cluster, experiment workflow and data collection. All the stations are connected to each other through gigabit switches. MyPLC [82] is used for setting up and managing the LAN computers. Fedora 10 images are prepared using vserver [48]. All the tools required on each node are bundled into this image. The image is customized for each node to allow for network configurations. The specifications of the network equipment and tools are shown in Tables 6.12 and 6.13.

**Experiment Control Setup**

The WEX Toolbox [32] employs SGE (Sun Grid Engine) [49] in order to manage scheduling and execution of tasks on the experimental cluster. The cluster consists of master and execution nodes. The functionality of the cluster is divided into two parts, namely Control Network and Experimental Network. The entire cluster, in this scenario, consists of 7 Dell Latitude E6500 laptops, but it can be extended to a large number of computers (a few hundred) quite
Table 6.4: LAN setup (Hardware requirements)

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computers</td>
<td>Dell Latitude E6500 laptops</td>
</tr>
<tr>
<td>Switches</td>
<td>Linksys SRW2016 16-Port 10/100/1000 Gigabit Switch</td>
</tr>
<tr>
<td>Ethernet Card</td>
<td>Intel® 82567LM Gigabit LAN card</td>
</tr>
<tr>
<td>Wireless Card (built-in)</td>
<td>Intel® WiFi Link 5300 AGN</td>
</tr>
<tr>
<td>Wireless Card (External)</td>
<td>Atheros 5212 PCI card (For experimental wireless network)</td>
</tr>
<tr>
<td>Spectrum Analyzer</td>
<td>Wi-Spy 2.4x</td>
</tr>
<tr>
<td>Processor</td>
<td>Two x86-based Intel core duo processors (@ 2.4 GHz)</td>
</tr>
<tr>
<td>Physical memory</td>
<td>4 GB</td>
</tr>
</tbody>
</table>

easily because we employ MyPLC [82] to manage the network resources. MyPLC employs virtual server [48] based Fedora OS images which are installed with all the required software. Because of the centralized management, it saves the experimenter from the hassle of catering to the setup issues on individual machines. Tools employed by the cluster are described in [32]. They are grouped under control network or experimental network. In our current deployment, control network consists of a master node, a database server and an analysis server, whereas experimental network consists of one access point, one source node, one receiver, two probes and one Wi-Spy based spectrum analyzer. The cluster can easily support groups of senders, receivers, probes, spectrum analyzers, access points, etc.

Control/Management Network

It provisions command and control interface for experimental network and enables remote configurations. Also it provisions a reliable mechanism to schedule tasks and collect data (traces and meta-data) according to the run rules in distributed computing environment. Master or server node is the brain of control network and is used to configure and manage rest of the nodes.

Experimental Network

All the nodes in experimental network are designated as execution nodes mainly because they run experimental tasks and applications as instructed by the scheduler daemon running on master node.
Table 6.5: LAN setup (software requirements)

<table>
<thead>
<tr>
<th>Software</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Fedora 10 (Kernel 2.6.27.14)</td>
</tr>
<tr>
<td>Wireless driver</td>
<td>MadWifi 0.94 revision 4928</td>
</tr>
<tr>
<td>Sniffers</td>
<td>Tshark, tcpdump, wireshark</td>
</tr>
<tr>
<td>Network file sharing</td>
<td>NFS 1.1.4-1</td>
</tr>
<tr>
<td>Time synchronization</td>
<td>NTP 4.2.4p7 (ntp-sop.inria.fr)</td>
</tr>
<tr>
<td>Network Management</td>
<td>MyPLC [82]</td>
</tr>
<tr>
<td>Spectrum Analyzer</td>
<td>Kismet Spectrum Tools [86]</td>
</tr>
<tr>
<td>Wireless Tools</td>
<td>Wireless Tools for Linux version 29 [36], Compat Wireless 2.6.32 [46]</td>
</tr>
</tbody>
</table>

6.1.5 Configuration: The Wireless experiment scenario

Tasks in this activity may vary greatly from scenario to scenario. Therefore, the configurations laid out hereunder are specific to the scenario chosen for this case study. The focus is on capturing the characteristics of wireless medium in order to enable in depth analysis of wireless network performance under varying channel conditions. To that end, we use the packet injection technique to generate traffic at the source node, capture traffic over the selected channel using probes and monitor RF activity in the 2.4GHz band using Wi-Spy spectrum analyzer [85]. Often, multiple runs of a wireless experiment for the same scenario are necessary. At the end of each run, data is transferred to the content/collection server. At the end of an experimentation session, data is preprocessed, analyzed and full disclosure reports are generated. Scenario configurations consist of relative placement of nodes, software/hardware configurations, wireless interface configurations, experimentation workflow configurations, etc., as described below.

Placement of nodes

Around 20 nodes are positioned in $8 \times 5$ m room in a regular fashion as shown in Figure 6.1. The nodes used in the case study are Source (labeled in red), Server, Probe 16, Probe 21, Probe 44, Probe 49 and Wi-Spy. The relative distances between the nodes can be estimated from the room dimensions. Source, Probe 16, Probe 21, Probe 44, Probe 49 and Wi-Spy participate in the experiment.

All the nodes are placed on top of wooden tables with metal structures underneath. All of the stations are at 0.75 m height from the floor. The room is located at the top floor of a 3-storey building and is not RF isolated from the outside world. Actually, many APs are present at
Table 6.6: WEX cluster (server side)

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduler</td>
<td>Configured to run every 8 seconds to schedule the execution of pending tasks</td>
</tr>
<tr>
<td>NTP</td>
<td>Network Time Protocol to ensure time synchronization</td>
</tr>
<tr>
<td>NFS server</td>
<td>Directories containing SGE binaries and experimentation scripts are shared on the cluster server</td>
</tr>
<tr>
<td>MySQL database server</td>
<td>Time sequenced unified repository for traces</td>
</tr>
<tr>
<td>Crunch XML</td>
<td>Export traces from intermediate XML format to database relations.</td>
</tr>
<tr>
<td>Logs</td>
<td>Errors, warnings, information during the course of operation of cluster.</td>
</tr>
<tr>
<td>Jobs</td>
<td>Experimental tasks are translated to jobs which are scheduled for execution on EN.</td>
</tr>
<tr>
<td>Java</td>
<td>Java version 1.6.0.17, Java™ SE Runtime Environment (build 1.6.0.17−b04)</td>
</tr>
</tbody>
</table>

Table 6.7: WEX cluster (client side)

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGE Execution daemon</td>
<td>Responsible for managing the execution of jobs on client nodes</td>
</tr>
<tr>
<td>NTP</td>
<td>Network Time Protocol to ensure time synchronization</td>
</tr>
<tr>
<td>NFS client</td>
<td>Shared directories are mounted</td>
</tr>
</tbody>
</table>

the different floors of the building, which makes possible to run experiments in a real working environment. As interferences are not controlled, it is crucial to be able to monitor the RF spectrum during the various experimentations.

Test Cases

We divide the scenario into four test cases as shown in Table 6.8. Each test case consists of four sessions per workday. Each session in turn consists of five runs. The sessions are desired to capture spatial and temporal variations in channel conditions. During each session, runs are launched one after the other every 7 minutes. Duration of each run is 300 sec. The role of the nodes employed is described in Table 6.9.
### 6.1 Case Study I: Wireless Channel Characterization

**Figure 6.1:** Indoor experimentation setup and placement of nodes

**Table 6.8:** Four test cases of the wireless scenario

<table>
<thead>
<tr>
<th>Test case</th>
<th>LOS / NLOS</th>
<th>Period (days)</th>
<th>Sessions per day</th>
<th>Runs per Session</th>
<th>Duration per run (sec)</th>
<th>Run offset (min) in each session</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Both</td>
<td>15</td>
<td>4</td>
<td>5</td>
<td>300</td>
<td>0, 7, 14, 21, 28</td>
</tr>
<tr>
<td>II</td>
<td>LOS</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>300</td>
<td>0, 7, 14, 21, 28</td>
</tr>
<tr>
<td>III</td>
<td>LOS</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>300</td>
<td>0, 7, 14, 21, 28</td>
</tr>
<tr>
<td>IV</td>
<td>LOS</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>300</td>
<td>0, 7, 14, 21, 28</td>
</tr>
</tbody>
</table>

**Test Case I**  
The purpose of first test case is to monitor *temporal variations in channel conditions* over a period of 3 weeks during workdays.
Table 6.9: Wireless scenario (Stations and their role)

<table>
<thead>
<tr>
<th>Node</th>
<th>Qty</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>1</td>
<td>Mode=Monitor, PHY rate=1 Mbps</td>
</tr>
<tr>
<td>Probes</td>
<td>4</td>
<td>Two LOS and two NLOS probes. Capture radiotap packet trace.</td>
</tr>
<tr>
<td>Spectrum analyzer</td>
<td>1</td>
<td>Employs Wi-Spy 2.4x spectrum analyzer and kismet spectrum tools</td>
</tr>
</tbody>
</table>

Four sessions (each consisting of 5 runs) are carried out each day according to the timeline shown in Table 6.10.

Table 6.10: Timeline of experiment sessions on each workday

<table>
<thead>
<tr>
<th>Session</th>
<th>Part of the day</th>
<th>Start time</th>
<th>Office hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Midnight</td>
<td>00:00</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Morning</td>
<td>09:00</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Noon</td>
<td>13:00</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Evening</td>
<td>18:00</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Test Case II

The purpose of this test case is to study the impact of orientation of receiver station with respect to the transmitter and receiver sensitivity on multipath fading, path loss, packet errors, packet loss, etc.

Starting from an initial reference orientation, receivers (or probes) are rotated by 90° for each subsequent session. The procedure is carried out for LOS probes only, i.e., Probes 16 and 44. The sender station labeled Source is kept fixed throughout the experimentation. Four sessions per day are carried out for a total of two days. In order to account for receiver sensitivity, wireless cards of the two probes are swapped for the set of last four sessions.

Test Case III

The purpose of this test case is to study the impact of orientation and machine itself on multipath fading, path loss, packet errors, packet loss, etc.

The orientation of the two stations is also changed with respect to the transmitter as de-
6.1 Case Study I: Wireless Channel Characterization

Figure 6.2: Displacement of probes

scribed. We did four sessions during one day. This test case is the same as test case II except that positions of stations Probes 16 and Probe 44 are interchanged.

**Test Case IV**

The purpose of this test case is to study the impact of small displacements of receivers on K factor and other metrics.

Experiments are performed against successive displacements of probes 16 and 44. Each displacement moves the station roughly 1 foot farther from its original position as shown in Figure 6.2. We carried out 4 sessions (3 sessions on one day and another session on another day) against 4 displacements Probe 44 and Probe 16. Each plot shows measurements against 4 sessions which indicates 4 displacements of a node.

**Software Parameters**

Sun Grid Engine version 6.2u2.5 [49] is used for scheduling experiment tasks. The scheduler is configured to periodically check the execution queues for pending experimental tasks. Wireless tools for Linux [36] version 29 is used for interface configurations. Packet injection [50] is used for traffic generation. In order to harness MetaGeek’s Wi-Spy 2.4x portable USB spectrum analyzer [85].

**Hardware Parameters**

All the nodes have x86 based architecture with 2 dual core CPUs. Each node has a total physical memory of 3.5GB and total swap size of 1024.0MB. Wi-Spy 2.4x is configured to scan radio activity in the entire 2.4 GHz band. We use Atheros wireless card (GWL G650) with
Madwifi (Multimode Atheros driver for Wi-Fi on Linux) version 0.9.4 revision 4128 from the trunk.

**Wireless Parameters**

MAC and PHY revisions used by the driver are 2414 and 2413 respectively. Channel type is 11g (operates in 2.4 GHz frequency range). Channel 11 is selected (this tells nodes to lock to the channel frequency 2.452). Fragmentation, RTS and retries are turned off. Transmission (TX) power is fixed at 18 dBm which is the maximum value for our Atheros wireless cards.

**Reference Time duration**

The total run time for an experiment is 345 seconds, but reference time duration for which results are calculated is 300 seconds. Difference between the total run time and reference time accounts for time required for scenario configuration.

**Run Rules and Workflow Configurations**

An experiment is formulated as a set of tasks which are configured to be executed according to a finite state machine (FSM). The workflow is as follows:

Wireless interfaces on the Source, Probe 44, and Probe 16 are configured 10 seconds after the launch of an experiment run. After waiting for another 15 seconds, TCPDump is launched on Probe 44 and Probe 16, and spectrum analyzer is launched on the Wi-Spy machine. TCPDump and spectrum tools are scheduled for execution for total duration of 320 seconds each. After waiting for another 10 seconds, DPS is put into action for exactly 300 seconds. DPS is terminated 10 seconds before the termination of TCPDump and spectools. The timeline of the flow of events during an experiment is demonstrated in Figure 6.3. Traces obtained for the first 10 and the last 10 seconds are discarded. The delays at the start and the end serve as grace periods. Long delays at the beginning are intended to allow the driver to reach steady state in terms of noise floor calibration and internal state. Also, there is an inter-run gap (i.e., pause between successive runs) when the experimentation session consists of multiple runs. The gap is set to 75 seconds.

**Metrics**

For this case study, we will measure RF interference, RSSI, Ricean K factor, Bit error rate, packet error rate and packet loss ratio at the probes.
6.1.6 Experiment: Undertake experiment execution and data collection

Launch experiment

A bootstrap python program, called primary scheduler, generates an initial schedule for all runs of the experiment. Input parameters of the primary schedule are desired number of runs and session start time. Initial schedule is a set of startup tasks, one for each run. A startup task encapsulates information such as start time of a run and link to the scenario definition files. Startup tasks are submitted to the grid engine right away. When an startup task gets executed by the grid engine, it generates a secondary schedule based on the scenario definition. Secondary schedule formulates the state machine of the run and governs the flow of tasks. These tasks specify each and every action to be performed on the target cluster nodes. Typical actions include scenario configurations, BSS setup, workload generation, traffic sniffing, capturing spectrum information, etc. Each task is converted as a job and submitted to the grid engine. We employ a naming convention based on timestamp and node ID to identify each run.

Workload generation

The packet injector DPS enables us to generate custom payload with different sizes, formats and traffic distributions. This is important for the soundness of performance measurements. In this scenario, DPS is used to generate data packets with a payload of 100 bytes each. Packets are transmitted at the maximum rate possible. We set the link bandwidth to 1 Mbps by setting the physical bit rate of the wireless interface to 1 Mbps. This results in DPS transmitting at an effective rate of less than 1 Mbps. In order to be able calculate bit errors, we set bits in the payload to all 1’s. First 8 bytes of the payload are reserved. Rest of the bytes is used for calculating bit errors per packet.
Trace capture

We use TCPDump to capture packet trace and spectrum analyzer’s to capture RF trace.

6.1.7 Preprocessing: Data cleansing, archiving and transformation

We identify each trace by assigning it an identification tag based on the timestamp and node ID. At the end of each run, traces are collected at the server. However, preprocessing is deferred until the end of entire experimentation session. This makes it easier to manage the traces. We filter out unwanted extraneous packets to save space, speed up packet transfer to database and later on decrease analysis time. Extraneous packets are the ones originating from other wireless networks deployed in the vicinity of wireless experimental setup. Traces are exported to an intermediate XML format which is then used to filter out relevant packet fields to MySQL database on a database server using CrunchXML.

6.1.8 Analyze: System performance

We implemented various scripts and programs to analyze packet traces. Analysis code is an ensemble of C++ programs, python and SQL scripts. An effort was made to avoid maintaining intermediate states and data. This means that analysis is performed on the actual data store each and every time. This practice facilitates reproducible analysis. We explain selected metrics and the mechanism to calculate each of them in Section 6.3.

Ricean K Factor

Figure 6.4 shows K factor values as measured on two LOS (line of Sight) probes named Probe 44 and Probe 16 and two NLOS (Non LOS) probes named Probe 21 and Probe 49. Large value of K signifies less scattering and reflections caused by surrounding objects/walls and hence smaller level of multipath fading which is the case for Probes 21, 44 and 49. Small value of K means greater depth of fading which is the case for Probe 16. Contrary to RSSI which has strong dependence on distance, multipath fading depends on location, orientation with respect to the obstructions in the environment.

Figure 6.5 shows K factor as estimated during test cases II, III, and IV corresponding to experiments with swapped cards, swapped probe locations and small probe displacements respectively.

In order to demonstrate the relationship between K factor and received power samples, we plotted histograms of received power at each probe against 5 runs of an experiment as shown in Figure 6.6. The histograms are also superimposed with fitted normal distribution. A steeper normal distribution curve mean less variation in the received power and hence higher K factor.
6.1 Case Study I: Wireless Channel Characterization

Figure 6.4: Average K factor on each probe during each session of test case I. There are 4 sessions per day and each x-axis tick mark represents the first session in the sequence.

value and vice versa. The highly stretched distribution of Rx power at Probe 16 as shown in Figure 6.6 explains the lowest K factor values at the same probe in Figure 6.6.

Bit Error Rate (BER)

As shown in Figure 6.7, bit error rate is more stable on Probe 16 and Probe 44 compared to probes 21 and 49 which are NLOS. Even between Probes 21 and 49, Probe 49 experiences higher bit errors the reason being Probe 49 is farther plus there is greater human activity in the room.

Figure 6.8 shows BER as measured during test cases II, III, and IV corresponding to experiments with swapped cards, swapped probe locations and small probe displacements respectively.

Packet Error Rate (PER)

Figure 6.9 demonstrates average packet error rate as experienced by the probes. PER is more reliable metric than BER because it gives an accurate measurement of how many packets
Figure 6.5: Top plot shows average K factor values on the LOS probes during test case IV. In the bottom plot, first 4 orientations correspond to initial reference orientations, next 4 orientations correspond to test case II (i.e., with swapped NICs) and last 4 orientations correspond to test case III (i.e., with swapped probes) were corrupted during each experiment run. Same explanation as provided for Figure 6.9 in section 6.1.8 applies to PER in Figure 6.9.

Figure 6.10 shows PER as estimated during test cases II, III, and IV corresponding to experiments with swapped cards, swapped probe locations and probe displacements respectively.

Packet loss ratio

Packet loss incurred at each probe demonstrated in Figure 6.11. Probes 49 and 21 experienced greater packet loss and more variations in packet loss compared to Probes 16 and 44. It is interesting to note that packet loss ratio is roughly 10 times greater than PER. This means that, in the real-world wireless, corrupted packets make only a small part of the lost packets.

Figure 6.12 shows K factor as measured during test cases II, III, and IV corresponding to experiments with swapped cards, swapped probe locations and probe displacements respectively.
Figure 6.6: Histogram of received power at the probes during one session (i.e., 5 runs).
Figure 6.7: Average BER on each probe for each session. There are 4 sessions per day and each x-axis tick mark represents the first session in the sequence.

Figure 6.8: Top plot shows average BER values on the LOS probes during test case IV. In the bottom plot, first 4 orientations correspond to initial reference orientations, next 4 orientations correspond to test case II (i.e., with swapped NICs) and last 4 orientations correspond to test case III (i.e., with swapped probes)
6.1 Case Study I: Wireless Channel Characterization

Figure 6.9: Average PER on each probe for each session. There are 4 sessions per day and each x-axis tick mark represents the first session in the sequence.

Figure 6.10: Top plot shows average PER values on the LOS probes during test case IV. In the bottom plot, first 4 orientations correspond to initial reference orientations, next 4 orientations correspond to test case II (i.e., with swapped NICs) and last 4 orientations correspond to test case III (i.e., with swapped probes)
Figure 6.11: Average packet loss ratio for each session on each probe. There are 4 experiments (representing 4 sessions) each day. Each x-axis tick mark represents the first experiment in the sequence.
Figure 6.12: Top plot shows average packet loss ratio values on the LOS probes during test case IV. In the bottom plot, first 4 orientations correspond to initial reference orientations, next 4 orientations correspond to test case II (i.e., with swapped NICs) and last 4 orientations correspond to test case III (i.e., with swapped probes)
6.2 Case Study II: Multicast Video Streaming over WiFi Networks: Impact of Multipath Fading and Interference

In the previous case study, we characterized wireless channel in terms of multipath fading, channel interference and path loss in conjunction with Bit Error Rate (BER), Packet Error Rate (PER) and packet loss. In this case study, we go one step further and investigate the impact of conditions determined through channel characterization on the network performance. We select multicast video streaming as the area of focus. Testbed setup is the same as in the previous case study. Only the activities represented by the circular arrows in the methodology are carried out.

The objective of this case study is to analyze the impact of interference, multipath fading and path loss on multicast video streaming (i.e., goodput, packet loss and ultimately on the video quality) using off-the-shelf fixed WiFi equipment in a wireless (802.11 b/g) local area network (WLAN) environment.

We use ricean K-factor as a measure of multipath fading, spectrum analyzer to estimate channel interference and received signal strength indicator (RSSI) as indication of signal power and attenuation. Furthermore, we use VQM (Video Quality Metric) score to compare the quality of received video with that of original video. In order to realistically measure forementioned metrics, we conducted extensive wireless experiments against six test cases representing common real-world situations using off-the-shelf wireless equipment. Also, we study the relative impact of channel interference and multipath fading.
The remainder of the case study is organized as follows: Section 6.2.1 provides an insight on multicast issues in wireless networks. Section 6.2.2 provides details about selected metrics, evaluation methodology, and wireless scenario. Section 6.2.5 provides details about the results and analysis.

6.2.1 Background

Multicast streaming is very useful for broadcasting live events, conference meetings, IPTV, distance education, etc. Unlike the wired packet switched networks where packet loss and delays are caused by congestion, wireless networks have to cope with unpredictable (random) channel conditions such as multipath fading, interference, path loss, etc. Wireless channel conditions could vary over very short time scale (order of microseconds). In case of 2.4 GHz band, small changes in path lengths can alter the signal highly since the wavelength is only 12.5 cm. Mobility of physical objects and people also impact the signal highly and cause signal envelop to fluctuate at the receiver. Measurement studies of fading report signal variations as high as 15-20 dB [73] [74]. Furthermore, because of license free access to the 2.4 GHz ISM band, it is difficult to avoid (inter/intra)-radio interference from other wireless networks, and other radio devices.

Therefore, multicast video streaming over wireless networks is more challenging as compared to their wired counterparts. As the UDP/IP communication stack does not provide any error detection and error correction scheme for multicast delivery over wireless channels [75] [80], video quality can suffer from loss of information aggravated by multipath fading and radio interference. Furthermore, because the wireless network is broadcast in nature, a packet is transmitted only once and will reach all the recipients. If the sender transmits regardless of whether receivers are ready or not, serious loss of data may result.

Recently, several studies have been conducted to investigate multicast streaming in WLANs and solutions have been proposed to improve its reliability and efficiency, but almost all of these undertakings rely on simulation tools [76]. However, the results obtained from simulations can be drastically different from those possible in the realworld environment [77].

Very few experimental studies have been conducted to evaluate multicast video streaming over WLAN networks. In [78], authors have studied the impact of packet size and rate of background unicast traffic on unicast video streaming in WLAN environment. Both the background traffic and video are relayed through the same AP. However, the experimentation scenario ignores interference from co-existing wireless networks and adjacent channels. In [80], authors provide an improved picture of multicast video streaming issues and solutions through experimentation. They study the impact of leader-based multicasting comparative to the normal multicasting on packet loss. Both these studies did not consider the impact of interference on video quality. Angrisani et al. [81] study the impact of interference in greater detail and report...
Chapter 6: Benchmarking Case Studies

quality and performance metrics at different layers of the protocol stack.

However, all these studies have completely ignored the impact of multipath fading which can have considerable impact on network performance as shown later in our analysis. Moreover, aforementioned studies lack experiment rigor in the sense that very few experiments (only one run per test case) are performed to draw the conclusions. Furthermore, as not enough details about the scenarios are provided, it is almost impossible to reproduce the results [79].

6.2.2 Experiments for multicast video streaming: Configuration and Execution

In this section, we present key metrics, measurement methodology and our multicast video streaming scenario. Key metrics were carefully chosen to represent channel characteristics as well as user level performance indicators. An indoor wireless testbed was set up using normal laptop machines. Two wireless networks were configured one for video streaming and one for generating adjacent channel interference. Video streaming network consists of multicast streaming server, client stations and probes. One spectrum analyzer was employed to log the entire 2.4 GHz band during the course of each experiment run. The scenario was subdivided into 6 test cases to reflect various channel conditions in different situations. The performance was evaluated using a set of performance and quality metrics as explained in section 6.2.2.

Metrics

The metrics are categorized into primary and secondary metrics. Secondary metrics are concerned with the channel characteristics and include ricean $K$-factor, RSSI and channel interference. These metrics can undergo high variations depending on the channel conditions. Primary metrics indicate network performance and depend on secondary metrics. Also they make more sense to the end user. We selected goodput, packet loss and VQM (Quality Video Metric) score [89] as primary metrics. For metrics such as $K$-factor, RSSI, goodput and packet loss, results are averaged over 5 measurements to ensure accuracy and confidence intervals are computed to signify the level of fluctuations around the mean result. VQM is explained below. Remaining selected metrics and the mechanism to calculate each of them is explained in Section 6.3.

Video Quality Metric (VQM)  VQM [89] is an objective measure of video degradation (compared to the original video) which reflects the human visual system (HVS). Quality estimates are reported on a scale from zero to one. On this scale, zero means that no impairment is visible and one means that the video clip has reached the maximum impairment level.
6.2.3 Method

We use VideoLAN VLC [83] to stream an MPEG-4 video clip to multicast clients from the streaming server. On each client, the video stream is captured into a file. At the same time, probes are used to capture packet trace of the video stream. We did not use the same node to capture both packet trace and video stream for performance reasons. Instead, each probe and each multicast client were placed on top of each other to ensure similar reception conditions. Captured video files are analyzed using BVQM (Batch Video Quality Metric) tool. Packet traces are loaded into a MySQL database and desired performance metrics are computed using SQL based analysis scripts.

6.2.4 Video Streaming Scenario

We employ one multicast streaming server, four multicast clients, one interference generator and four probes as shown in Table 6.11. Packet trace with radiotap wireless headers is captured using TCPDump. RF activity in the 2.4 GHz band is recorded using Wi-Spy spectrum analyzer [85].

<table>
<thead>
<tr>
<th>Node</th>
<th>Qty.</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multicast Server (MS)</td>
<td>1</td>
<td>Serves as both AP and Video Streaming Server. Channel=11, multicast PHY rate=24 Mbps</td>
</tr>
<tr>
<td>Multicast Clients (MCs)</td>
<td>4</td>
<td>Receive and capture the video stream. Channel=11</td>
</tr>
<tr>
<td>Probes</td>
<td>4</td>
<td>Capture radiotap packet trace. Channel=11</td>
</tr>
<tr>
<td>Interference Generator</td>
<td>1</td>
<td>Serves as both AP and traffic generator. Channel=10, PHY rate=24Mbps, data rate=11Mbps</td>
</tr>
<tr>
<td>Spectrum analyzer</td>
<td>1</td>
<td>Employs Wi-Spy 2.4x spectrum analyzer and kismet spectrum tools</td>
</tr>
</tbody>
</table>

Table 6.11: Multicast video streaming scenario

Configuration

We setup a wired LAN using MyPLC [82] in order to manage wireless testbed resources. For scenario configurations, experiment workflow and data collection, we employ WEX (Wireless EXperimentation) toolbox [88]. The specifications of the network equipment and tools are shown in Tables 6.12 and 6.13.
Chapter 6: Benchmarking Case Studies

Table 6.12: Wireless scenario (Hardware specifications)

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computers</td>
<td>Dell Latitude E6500 laptops</td>
</tr>
<tr>
<td>Wireless Card (built-in)</td>
<td>Intel® WiFi Link 5300 AGN</td>
</tr>
<tr>
<td>Wireless Card (External)</td>
<td>Atheros 5212 PCI card</td>
</tr>
<tr>
<td>Spectrum Analyzer</td>
<td>Wi-Spy 2.4x</td>
</tr>
<tr>
<td>Processor</td>
<td>Two x86 Intel core duo processors (2.4 GHz)</td>
</tr>
<tr>
<td>Physical Memory</td>
<td>4 GB</td>
</tr>
</tbody>
</table>

Table 6.13: Wireless scenario (software specification)

<table>
<thead>
<tr>
<th>Software</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Fedora 10 (Kernel 2.6.27.14)</td>
</tr>
<tr>
<td>Wireless driver</td>
<td>MadWifi 0.94 revision 4928</td>
</tr>
<tr>
<td>Sniffer</td>
<td>TCPDump</td>
</tr>
<tr>
<td>Packet analyzer</td>
<td>Tshark</td>
</tr>
<tr>
<td>Spectrum Analysis Tools</td>
<td>Kismet Spectrum Tools [86]</td>
</tr>
<tr>
<td>Wireless Tools</td>
<td>Wireless Tools for Linux version 29</td>
</tr>
<tr>
<td>Streaming Server and Clients</td>
<td>VLC 1.1.7, communication protocol = MPEG-TS (Transport Stream)</td>
</tr>
<tr>
<td>Video</td>
<td>Format = MPEG, bit rate = 800 kbps, fps = 25, resolution = 640 x 360</td>
</tr>
</tbody>
</table>

Placement of nodes

Around 20 nodes are installed in 8 × 5 m room in a regular fashion as shown in Figure 6.13. The nodes used in wireless video streaming scenario are multicast streaming server (MS) (labeled in red), experiment control server, probes, multicast clients, interference generator and Wi-Spy. The nodes are placed on top of wooden tables with metal structures underneath. All of the stations are at 0.75 m height from the floor. The room is located at the top floor of a 3-floor building and is not RF isolated from the outside world. Actually, many APs are present at the different floors of the building, which makes possible to run experiments in a real working environment.
6.2 Case Study II: Multicast Video Streaming over WiFi Networks

Software Parameters

Wireless tools for Linux version 29 is used for interface configurations. VLC is used generate multicast video stream. In order to harness MetaGeek’s Wi-Spy 2.4x portable USB spectrum analyzer [85], we use open-source tools from kismet known as Kismet spectrum tools [86] with custom modifications.

Hardware Parameters

Wi-Spy 2.4x is configured to scan radio activity in the entire 2.4 GHz band. We use Atheros wireless card (GWL G650) with Madwifi (Multimode Atheros driver for Wi-Fi on Linux) version 0.9.4 revision 4128 from the trunk. Antenna diversity is disabled on all the machines in order to get consistent K-factor values.

Wireless Parameters

MAC and PHY revisions used by the driver are 2414 and 2413 respectively. Channel type is IEEE 802.11g (operates in 2.4 GHz frequency range). Fragmentation, RTS and retries are turned off. Transmission (TX) power is fixed at 6 dBm. The maximum transmission power for our Atheros wireless cards is 6 dBm.

Time duration

The total time duration, for which traffic is generated and results are calculated, is 200 seconds.
Workload generation

We use an MPEG-4 video clip to stream over the wireless network using VLC as streaming server. Video clip is transcoded using MPEG-4 codec and transmitted using RTP protocol. The video is played for a duration of 200 seconds.

Scenario Test Cases

The entire video streaming experimentation campaign consists of six test cases as shown in Table 6.14. The first 4 test cases were carried out in the afternoon during office hours. Two more cases were tested in non-office hours, when the spectrum is usually quieter, to focus on the impact of only multipath fading on packet loss.

<table>
<thead>
<tr>
<th>Test case</th>
<th>Office hours</th>
<th>Description</th>
<th>Runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Video streaming + controlled interference − human movements</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>Video streaming − controlled interference − human movements</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Yes</td>
<td>Video streaming + controlled interference + human movements</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Video streaming − controlled interference + human movements</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Video streaming − controlled interference + human movements</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Video streaming − controlled interference − human movements</td>
<td>5</td>
</tr>
</tbody>
</table>

6.2.5 Analysis: Multicast streaming performance

In this section, we demonstrate plots for the metrics described in section 6.2.2 and explain the results in the light of various factors such as channel interference, multipath fading, signal attenuation, etc. The results in section 6.2.6 correspond to the measurement campaign conducted during office hours when interference from production networks is high. However, the level of interference from external networks was comparatively lower during test cases 3 and 4. So despite higher multipath fading during test cases 3 and 4, network performance slightly improved. This is because performance improvement brought by reduced interference overshadowed the packet loss induced by increased multipath fading. In order to demonstrate this, we added test cases 5 and 6 [Table 6.14] to understand the impact of multipath fading in greater isolation from interference. The results for experiments conducted during the two cases are reported in section 6.2.7.
6.2.6 Measurements during office hours

6.2.6.1 Channel interference and radio activity in 2.4 GHz band

The RF landscape in 2.4 GHz wireless band during the course of one wireless experiment is shown in the Figure 6.14. Spectrum information captured by Wi-Spy spectrum analyzer is in the form of frequency vs. amplitude. For graphical demonstration, we map entire band 2400 - 2483 MHz each frequency to the corresponding 14 WiFi channels.

Figure 6.15 shows the evolution of channel interference during 6 test cases. Overall interference (per channel) for channels \([8, 13]\) is estimated by averaging all the frequency amplitudes falling in the frequency range of each channel. Channel interference results in an increase in average signal amplitude in that channel. It is clear that during first 3 test cases (especially in test cases 1 and 2) interference is higher than rest of the cases. The impact of interference on goodput and packet loss is demonstrated in subsections 6.2.6.4 and 6.2.6.5.

![Figure 6.14: Spectrum analysis in test case 6](image)

6.2.6.2 Ricean K-factor

Figure 6.16 shows the impact of each test case on multipath fading. Large value of \(K\) signifies lower multipath fading which is the case in test cases 1 and 2 when there are no movements in the environment. Small value of \(K\) means greater depth of fading which is the case for test cases 3 and 4 when there are human movements. This fact is further explained
by sub figures in the Figure 6.17 which demonstrate received power at Probe 2 in 4 test cases. The band representing received power in case 1 and 2 is thinner than cases 3 and 4, therefore K-factor is greater in cases 1 and 2 than in cases 3 and 4. Both location of receivers and movement of objects in the environment have a clear impact on the fading. In the first 4 test cases, the impact of fading on packet loss is not obvious as shown in Figure 6.20 and goodput as shown in Figure 6.19 because of substantial interference from external networks. The behavior is investigated more in section 6.2.7.

6.2.6.3 RSSI

As it is obvious from the Figure 6.18, average RSSI remains pretty much same despite fluctuations caused by the movements in the wireless environment. Because all the receivers are placed within the same room, the signal is strong enough and the transmission power at AP, does not have any noticeable impact on packet loss and K-factor. Furthermore K-factor depends more on signal fluctuations rather than the strength of the signal as shown in Figure 6.16.
6.2.6.4 Goodput

Figure 6.19 shows average goodput as well as goodput variations (confidence intervals) as measured on each probe. There is a drop in goodput at all the probes in cases 1 and 3 when there is more interference comparative to the cases 2 and 4 [Table 6.14].

6.2.6.5 Packet loss ratio

Packet loss incurred at each probe during the course of experiments, corresponding to the first 4 test cases [Table 6.14], is demonstrated in Figure 6.20. There is higher packet loss when there is more interference in the wireless environment.

6.2.6.6 VQM (Video Quality Metric)

We use BVQM tool [89] in order to assess the quality of captured videos on multicast clients. Video is calibrated using full reference calibration model and VQM is computed using video conferencing model. Video scanning standard is set to progressive. One video (out of 5) on each multicast client is selected against each test case. Figure 6.21 shows the results for 24 such videos that belong to test cases 1 to 6 [Table 6.14]. Videos clips 1 to 4 (corresponding to clients
Chapter 6: Benchmarking Case Studies

6.2.7 Measurements during non-office hours

Figure 6.22 demonstrates the impact of ricean fading on packet loss. Each packet loss value has been averaged over 5 runs. In the face of movements in the wireless environment [test case 5], ricean K-factor on average fell below 10. This caused almost 50% increase in packet loss. However, the packet loss reported in Figure 6.22 against cases 5 and 6 is much lower than the packet loss reported in Figure 6.20 against cases 1, 2, 3 and 4. This is because there was significant interference from production networks during office hours which caused more...
packet loss compared to multipath fading. The impact of fading became prominent when the 2.4 GHz spectrum was less congested during non-office hours.

### 6.2.8 Conclusion

We conducted six sets of multicast video streaming experiments over 802.11 b/g WLAN against six test cases corresponding to different realworld situations with varying levels of exogenous interference and signal fading. It is shown that interference has more impact on performance than multipath fading. Multipath fading can result in considerable performance degradation in environments where moving objects cause perturbation. On the contrary, channel interference is more frequent and more prominent cause of performance degradation in wireless networks because ISM 2.4 GHz band is increasingly being utilized in homes and work places. Being able to quantify the impact of multipath fading and interference is crucial in planning, troubleshooting, managing as well as benchmarking and optimizing wireless networks.
Chapter 6: Benchmarking Case Studies

Figure 6.19: Goodput averaged over 5 runs

Figure 6.20: Packet loss ratio averaged over 5
Figure 6.21: Quality of video received at each client in cases [1,6]

Figure 6.22: K-factor and Packet loss ratio averaged over 5 runs
6.3 Metrics

6.3.1 Channel interference and RF activity in 2.4 GHz ISM band

Because the radio spectrum used by wireless LAN (WLAN) is freely available for public and research use, it is usually highly congested. Interference occurs when communication from one node impedes communication from another node. Interference can be caused by not only wireless networks but also by devices such as wireless game controllers, Bluetooth, microwave, WiMAX, etc. The purpose is to capture frequency fluctuations in the entire wireless spectrum of either 2.4 GHz band (or at least adjacent channels) and study the impact of the level of interference on performance metrics such as BER/PER, packet loss, goodput, etc. The level of interference is quantified by computing average signal power over all the collected RF samples. Average signal power is relatively higher in the presence of channel interference.

Spectools \[86\] is configured to log frequency fluctuations for 2.4 GHz band. It collects information consisting of frequency range 2.400 to 2.483 at 419 points with a step size of 119 kHz. The rate at which it can capture samples depends on the processing time, called sweep time which in our case is around 600 ms. The RF trace file is a sequence of tuples of the form time, frequency, amplitude. It can be used to plot a variety of graphs, e.g., frequency vs. amplitude, amplitude vs. frequency vs. time, frequency vs. running, average and peak amplitudes, etc.

6.3.2 Ricean K-factor

Ricean distribution is similar to rayleigh distribution except that a deterministic strong component is present. It is completely defined by ricean K-factor. K-factor is defined as the ratio of the signal power in dominant component to the (local-mean) signal power in multipath components. When the dominant component between the transmitter and the receiver disappears, K approaches 0 and ricean distribution degenerates to Rayleigh distribution. Therefore, the higher K is, the less multipath fading is. We estimate K-factor from empirical power samples using a moment based method as explained in [87]. Received power measurements are extracted from the received packets and K is obtained using the following equation

\[
K = \frac{\sqrt{1 - \gamma}}{1 - \sqrt{1 - \gamma}} \quad (6.1)
\]

where \(\gamma = \frac{V[R^2]}{E[R^2]} \), with \(V[.]\) denoting the variance, \(E[.]\) denoting the expectation and \(R\) denoting the received signal envelope. According to the literature, equation (6.1) gives fairly accurate estimation of K for a sample size of at least 500. In our estimation, each K value is calculated using 120,000 samples in Case Study I [6.1] and 20,000 samples in Case Study II [6.2] on average.
6.3 Metrics

We developed both Matlab and SQL based scripts for estimating the K factor. Received power measurements are extracted from the received packets. Wireless interface measures the power in dBm which is a logarithmic scale. We convert the power measurements into Watts, normalize and then apply the formula (6.1).

Received Signal Strength Indicator (RSSI)

RSSI is a measure of power present in the RF signal. RSSI implementation varies from vendor to vendor. In madwifi, RSSI is equivalent to signal-to-noise ratio and essentially is a measure of signal power above the noise floor. It is calculated for each packet by subtracting noise power from the received signal power.

6.3.3 Packet Loss ratio

Packet loss ratio is ratio of the number of packets lost to the total number of packets transmitted. In Case Study I [6.1], it is determined from the sequence numbers (embedded in the payload) of correctly received packets. In Case Study II [6.2], it is estimated by examining RTP sequence numbers of the received packets.

6.3.4 Bit Error Rate (BER)

BER is a very important metric which shows the efficiency and resilience of the entire communication path including transceivers and the communication channel. BER is calculated as the number of bits flipped in each packet with bad CRC flag. We consider only the payload bits and any bits flipped in the packet headers are not accounted for.

6.3.5 Packet Error Rate (PER)

PER is the ratio of number of packets received with CRC errors to the total number of packets sent.

6.3.6 Goodput

Goodput is computed using a time window of 100 ms. In our wireless scenarios, as the traffic is generated at low rate (at most 1 Mbps), we are able to signify goodput fluctuations better using the said time window.
6.4 Full Disclosure Reports

Figure 6.23 is a typical FDR for reporting benchmarking score. In this case it shows K factor results reported in Figure 6.4 in Case Study I [6.1]. It is desired to be one page long and yet contain sufficient information for others to be able to correctly interpret and compare the results. As an example, we have provided metric description, computation formula, orchestration of the experiment campaign, scenario configuration, workload parameters and metadata. We believe that for K factor this much information is sufficient. Note that the report does not mention channel interference and path loss. This is because multipath fading does not depend on them.

6.5 Fair Comparison

To enable comparability, we propose to cluster runs based on experiment conditions. Exploratory tests can be performed to investigate anomalies and perform calibrations. Basic sanity checks can be performed to verify accuracy of software/hardware tools. For example, performance of packet generator/sniffer, antenna diversity, time synchronization, etc. Empirical estimates of the metrics such as Ricean K factor are verified against the theoretical models.

Figure 6.22 demonstrates the variations in Ricean K factor caused by human traffic and its impact on the percentage packet loss. Test case 5 corresponds to the case when there is human traffic in the vicinity. On the contrary, there is no human traffic during Test case 6 [6.2.4]. Ricean K factor defines one the uncontrollable characteristics of the wireless channel. By clustering, we will be able to group experiments into clusters according to criteria such as high, medium and low human traffic. This will allow us to perform apples to apples comparison with future experiment campaigns. This is especially useful for a wireless experimentation campaign conducted over an extended period of time.
Figure 6.23: Full Disclosure Report for K factor
CONCLUSION

In this thesis, we have introduced and clarified the concept of benchmarking in the context of wireless networks. Benchmarking is a very powerful tool for in-depth objective performance evaluation of wireless networking protocols. It can facilitate the selection, adaptation and deployment of best applications and protocols. However, the potential of benchmarking in wireless networks hasn’t yet been realized to its fullest, the reason being inherent complexities in the wireless network and the test environments, lack of best practices and software tools. This thesis was aimed at providing an answer to these issues.

This chapter concludes our thesis. In Section 7.1, we summarize our contributions to the benchmarking of wireless networks. In Section 7.2, we provide direction for the future work and enhancements.
Chapter 7: Conclusion

7.1 Overall Closure Comments

Wireless Benchmarking is the scientific method of protocol evaluation which requires *scientifically rigorous experimentation, repeatable results and fair comparison*. These requirements can only be met by following a *benchmarking methodology* that provides a complete roadmap for scenario configuration, experiment execution, packet/RF trace and metadata collection, analysis and reporting of results. It is also required to maintain provenance (chronological record of measurement and analysis steps) of results along with analysis scripts and record of *controllable and uncontrollable experiment parameters*.

The methodology proposed in this thesis is designed to achieve these benchmarking requirements. Asides from the methodology, a benchmarking framework which allows smooth execution of the benchmarking methodology was also developed. The framework was used to conduct extensive wireless experiments and to refine the methodology. Several pitfalls were identified during the process which if ignored can lead to misleading interpretation of the results. Some of them are as follows:

- **Antennas do not pick every transmission and sniffer can drop packets without any trace.** We propose to use multiple probes with good processing power and light-weight sniffers.

- **Improper sniffer configuration can cause random packet drops.** Calibration is advised to ensure sane working of the sniffer.

- **Small changes in placement and orientation of antennas result in big changes in multipath fading.** We recommend the use of K factor to compare fading in two wireless environments.

- **In real-world, interference from collocated networks is almost impossible to avoid.** We recommend the use of spectrum analyzer to record co-channel and adjacent channel interference.

- **Antenna diversity, if enabled, can result in unexpected changes in the received power.** One needs to be aware of the impact of antenna diversity on measurements.

To avoid pitfalls, software/hardware tools need to be calibrated against *sanity checks*, e.g., power control, time synchronization, antenna diversity, etc.

According to the proposed benchmarking methodology, *scientific rigor* in wireless experiments can be accomplished by scheduling large number of experiments (possibly hundred
of them) and computing averages for the selected metrics and confidence intervals around them. Uncontrollable parameters in the wireless environments do not permit repeatability. We therefore, propose statistical repeatability where an experiment is considered repeatable if it yields similar results under similar networking conditions. Similarity of networking conditions is determined by comparing recorded controllable and uncontrollable parameters and channel conditions. Fair comparison is accomplished by clustering experiments with similar conditions and then doing apples to apples comparison.

Benchmarking requires rigorous ‘in field’ experimentation and tools which can facilitate the above benchmarking activities. Given that management of even a small number of nodes can be cumbersome, benchmarking poses following challenges:

- Scheduling and managing large number of experiments with multiple runs
- Recording controllable and uncontrollable parameters which matter for correct interpretation of results and fair comparison.
- Management and indexing of large amount of packet traces and metadata.
- Batch analysis over large experimentation campaigns.
- A convenient mechanism to correctly interpret the reported results.

We addressed these challenges by developing a Benchmarking Framework. The framework is designed to facilitate the execution of benchmarking methodology. It is also a complete wireless experimentation toolkit. It facilitates all the stages of a wireless experiment right from testbed setup to result reports. It has been used actively in indoor wireless environment. Using the framework, we conducted extensive wireless experiments. Two rigorous case studies were carried out. In the first case study, we characterized the wireless channel using ricean K factor, path loss, channel interference as the metrics. BER/PER and packet loss were also measured. We used packet injection technique to generate traffic directly at the MAC layer with custom payload. Four hundred experiment runs were carried to record spatial and temporal channel variations in 802.11 b/g wireless networks. In the second case study, we conducted six sets of multicast video streaming experiments over 802.11 b/g WLAN against six test cases corresponding to different real-world situations with varying levels of exogenous interference and signal fading. We showed that by grouping experiments according to the similarity of experiment conditions, we can enable repeatability and fair comparison. In the end, we have presented a Full Disclosure Report (FDR) for ricean K factor. The report is intended to provide sufficient information for others to be able easily interpret benchmarking score and make meaningful comparison.
7.2 Perspectives of Future Work

We argue that benchmarking is key to realistic and fair comparison of wireless networking protocols. There is considerable interest in wireless benchmarking from both academia and industry. There is a lot of space to explore and huge potential to contribute. Some rewarding contributions can be made along following paths:

- One such interesting contribution can be the ability to benchmark protocols using simulation, say ns-3, and then using experimentation by translating the same simulation script to experiment description language (EDL).

- Another dimension is to validate simulation models by translating wireless channel conditions to simulation scripts.

- It would be desirable to work on clustering of wireless experiment data (packet traces, RF traces and metadata) to group experiments into clusters for meaningful comparisons.

- It is desirable to have a unified data repository (in the form of a database) to store all controllable/uncontrollable experiment parameters, analysis steps and scripts in the database. Tools are needed to dynamically generate full disclosure reports (FDR) from the data repository or through a web portal.

- It is also highly desirable to share benchmarking score, FDRs and experiments through a web portal.
7.2 Perspectives of Future Work
Appendix A


The material provided in this appendix is intended to supplement Chapter 5 and also to server as a basic guide for the usage of WEX Toolbox. The tools invoked in the code snippets are available online at [32].

A.1 Experiment Description Language (EDL)

```xml
<?xml version="1.0"?>
<experiment>
  <shell> /bin/bash </shell>
  <startup>
    <sanity enable="yes">
      <check> connectivity </check>
    </sanity>
    <metadata enable="yes">
      <info> memsize, mem in use </info>
    </metadata>
  </startup>
  <net_config>
    <eth_domain> s.pl.sophia.inria.fr </eth_domain>
    <wi_domain> 192.168.11.x </wi_domain>
  </net_config>
  <groups>
    <group mode="monitor">
      <hosts> <host no="16"> wlab16 </host> </hosts>
```


A.2 Example Probe Configuration

Listing A.1: Configure a wireless interface as probe

```
#!/bin/bash

# Mark the probe interface ath0 down
```
A.3 Scheduling Multiple Experiments and Runs

Listing A.2: Scheduling multiple runs of an experiment. -t:star time and -r:runs

```
./schedule.py -t "2011-03-21 16:55:00" -r 5
```

Listing A.3: Scheduling experiments.
for i in range(1, int(self.runs) + 1):
    sgeTask = os.getcwd() + "/run%0%s.sh" % (i, next_launch.strftime("2011-03-21 16:55:00"));
ofile = open(sgeTask, 'wb')
ofile.write("#!/bin/bash
")
# Set SGE variables
ofile.write("#$ -N run_no%0s
" % (i))
ofile.write("#$ -S /bin/bash
")
ofile.write("#$ -cwd"
")
ofile.write("#$ -q all.q@wlabXXX.inria.fr"
")
ofile.write("#$ -a %
" % (next_launch.strftime("2011-03-21 16:55:00"))
")
ofile.close()
os.chmod(sgeTask, 0777);

# Set execution time for this task
next_launch = next_launch + relativedelta(seconds=+5);
# ACTION performed by this task
ofile.write("./parse.py -e ed.xml -t "%%s" % (next_launch.strftime("%%Y-%m-%d %H:%M:%S")))
ofile.close()

os.chmod(sgeTask, 0777);

# Submit the task
cmd = 'qsub %s' % (sgeTask)
p = subprocess.Popen(cmd, shell=True, executable="/bin/sh", stdin=subprocess.PIPE)
sout = p.communicate()[0];
print "%%s" % (sout);

# Gap of approximately 2 minutes between successive runs. Can be changed according to the requirements.
next_launch = next_launch + relativedelta(minutes=+6, seconds=+55);

A.4 Data Management

A.4.1 Organization

Listing A.4: Organize traces into data bins (or archives)
A.5 Loading Packet Traces in MySQL Database

Listing A.5: Create an index over all the experiment runs

```bash
#!/bin/bash
t tracedir=path_to_the_trace_directory;
# create index with one index entry for each run. -f is the path to index file
./OrganizeTraces.py -l tracedir -f tracedir/dblist
```

A.4.2 Indexing

Listing A.6: Create databases and schemas

```bash
#!/bin/bash
t tracedir=path_to_the_trace_directory;
# create databases and schemas (set of tables)
./firstmodule.py -d connection.wx -l tracedir/dblist -q shafqat_script_nstd3.sql --create
```

A.4.3 Schema Management

Listing A.7: Load packet traces in the database repository

```bash
#!/bin/bash
t tracedir=path_to_the_trace_directory;
# Convert traces to xml files, filter out relevant data from traces and load it in MySQL databases
./tcpdump2mysql.py -d connection.wx -t tracedir -l tracedir/dblist
```
A.6 Analysis

```bash
#!/bin/bash

tracedir=path_to_the_trace_directory;

# Calculate goodput, RSSI, received power and estimate ricean K Factor and ricean←
distributions
./analyze.py -d connection.wx -t tracedir -l tracedir/dblist -q power.sql -s "←
" -o power.dat

# Calculate and plot BER and PER
./analyze_ber.py -t tracedir -l tracedir/dblist

# calculate packet errors
./analyze_ploss.py -t tracedir -l tracedir/dblist

# plot WIFI spectrum activity
./analyze_spec.py -t tracedir -l tracedir/dblist
```
Appendix B

K Factor Estimation using SQL

The ricean K factor is one of several measures of characterization of wireless environment. K factor completely defines the *ricean distribution*. The higher K Factor is, the less signal fading is. *Rayleigh distribution* is a special case of ricean distribution. When the direct LOS or dominant component between the transmitter and the receiver disappears, K approaches zero and ricean distribution degenerates to Rayleigh distribution.

Related studies on empirical estimation of K factor employ specialized hardware. On the contrary, we take advantage of off-the-shelf network equipment and tools to collect packet traces and estimate K factor from RX power of received data packets using following formula:

\[
K = \frac{\sqrt{1 - \gamma}}{1 - \sqrt{1 - \gamma}}
\]  

(B.1)

B.1 Estimation Algorithm

The steps followed to compute K in equation B.1 are outlined below. Corresponding SQL script is provided in Section B.2.

1. let R be the received power, then normalized received power is

   \[ R = \frac{R}{M} \]

   where \( M = \text{AVG}(R) \)

   R is converted to linear scale (Watts) from logarithmic scale (dBm) using

   \[ R = \text{POW}(10, R/10)/1000 \]

   The result is stored in a view called T1.

2. In order to calculate \( \gamma \), we need to calculate *mean* and *variance* for the resultset in T1.
Chapter B: K Factor Estimation using SQL

\[ E = \frac{1}{\text{count}()} \times \text{sum}(T1.R) \]

This implies that

\[ \text{gamma} = \frac{\text{VARIANCE}(T1.R)}{\text{POW}(E, 2)} \]

The result is stored in view T2. Finally, we calculate K factor from T2 using the formula

\[ K = \sqrt{1 - T2.\text{gamma}} / (1 - \sqrt{1 - T2.\text{gamma}}) \]

which corresponds to the K factor formula in equation B.1.

### B.2 SQL-based K Factor Calculation

Listing B.1: K factor calculation using SQL for Probe 16 and experiment u20100201170015

```sql
/* Calculate R */
SELECT T1.probe, T1.datas, SQRT(POW(10, T1.datas/10) / 1000) R
FROM (SELECT probe, dbm_antsignal
    FROM u20100201170015.radiotap
    WHERE probe = 16 AND frame_len = 602)
    datas
    FROM u20100201170015.radiotap
    WHERE probe = 16 AND frame_len = 602) T1;

/* Calculate E and gamma from R */
SELECT T2.probe, ( (1/count(*)) * sum( pow(T2.R, 2) ) ) E,
    (VARIANCE(pow(T2.R, 2))) / ( pow( (1/count(*)) * sum( pow(T2.R, 2) ), 2 ) ) gamma
FROM T1, T2;

/* Calculate K from gamma */
SELECT T3.probe, sqrt(1 - T3.gamma) / (1 - sqrt(1 - T3.gamma)) as K
FROM (SELECT T2.probe,
    ( (1 / count( )) * sum(pow(T2.R, 2)) ) E,
    (VARIANCE(pow(T2.R, 2))) / ( pow((1 / count( )) * sum(pow(T2.R, 2)), 2) )
FROM T1 ) T2;
```
gamma

FROM T2) T3;
Chapter B: K Factor Estimation using SQL
Appendix C

RSSI and SNR

We have used RSSI (Received Signal Strength Indicator) and SNR (Signal to Noise Ratio) interchangeably although these terms are generally considered different. SNR is considered a ratio based value that shows signal level based on the noise in the channel. RSSI is considered to simply indicate strength of the received signal. However, they are treated as the same in MADWiFi which is an open-source WiFi driver used for wireless experimentation during this thesis. To make the concept clear, a brief overview of RSSI and SNR is given below.

RSSI is one of measurements of the power present in the RF signal. Other RF signal strength measurements are mW, dBm and percentage. It is unitless integer value and is allowed to fall within the range 0...255 (a one-byte value). No vendor has, thus far, chosen to report 256 different signal levels. So each vendor’s wireless NIC will have a specific max RSSI value (RSSI\_max). For example, Atheros Wi-Fi chipset (the one used in our experiments) uses an RSSI\_max value of 127 which means it will report 128 different power levels. The 802.11 standard does not specify any relationship between RSSI and power levels in mW and dBm. It is up to the vendors to provide their own accuracy, granularity and range for RSSI in order to characterize actual signal strength.

In madwifi, the reported RSSI is actually equivalent to Signal-to-Noise Ratio (SNR) i.e., RSSI = Signal - Noise. The 'quality' parameter reported by some of the Wireless Tools such as iwconfig is used by madwifi to report the SNR (RSSI). Therefore, link quality, SNR and RSSI represent the same measure in the context of MadWiFi. This does not hold for other drivers though [125].

All signal strength measurements in 802.11 are based on RSSI, but 802.11 does not mandate how RSSI should be calculated, so different vendors will almost measure it differently. Because of unspecified and usually-low precision of 802.11 card’s RSSI, it should not be assumed that signal strength as reported by an 802.11 card is a reliable method of determining whether one
signal is stronger than another. If the signals are relatively close in power, the 802.11 card will probably say that they are same strength.

In general, an RSSI of 10 or less represents a weak signal although the chips can often decode low bit-rate signals down to -94 dBm. An RSSI of 20 or so is decent. An RSSI of 40 or more is very strong and will easily support both 54 Mbps and 108 Mbps operation.
Appendix D

Publications

Following research work was produced during the development of this thesis:

D.1 Journals

1. Shafqat-ur-Rehman, Thierry Turletti, Walid Dabbous, A Roadmap for Benchmarking in Wireless Networks. [In Submission]

D.2 Conferences

1. Shafqat-ur-Rehman, Thierry Turletti, Walid Dabbous, Multicast Video Streaming over WiFi Networks: Impact of Multipath Fading and Interference, IEEE Workshop on multiMedia Applications over Wireless Networks (MediaWiN), June 28 - July 1, 2011, Corfu, Greece.


D.3 White Papers and Technical Reports


129
D.4 Posters


All the work listed above is available online at [126]
BIBLIOGRAPHY


[33] Channel characterization using WEX Toolbox, [https://twikisop.inria.fr/twiki/bin/view/Projets/Planete/ChannelCharacterization](https://twikisop.inria.fr/twiki/bin/view/Projets/Planete/ChannelCharacterization).


[71] Cristian Tala, Luciano Ahumada, Diego Dujovne, Shafqat-Ur Rehman, Thierry Turletti, and Walid Dabbous, Guidelines for the accurate design of empirical studies in wireless networks, 7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities(TridentCom), April 2011, Shanghai, China.


[80] D. Dujovne, T. Turletti, *Multicast in 802.11 WLANs: An Experimental Study*, MSWiM’06, October 2-6, 2006, Torremolinos, Malaga, Spain.


[82] PlanetLab MyPLC, [https://svn.planet-lab.org/wiki/MyPLCUserGuide](https://svn.planet-lab.org/wiki/MyPLCUserGuide)


[88] WEX Toolbox, [http://planete.inria.fr/Software/](http://planete.inria.fr/Software/)


[90] PhySimWiFi for NS-3, [http://dsn.tm.uni-karlsruhe.de/english/misc_2945.php](http://dsn.tm.uni-karlsruhe.de/english/misc_2945.php)


[110] Iperf tool, http://xb.pert.geant.net/PERTKE/IperfTool


ABSTRACT

The objective of this thesis is to enable realistic yet fair comparison of the performance of protocols and applications in wireless networks. Simulation is the predominant approach for comparable evaluation of networking protocols however it lacks realism and can lead to misleading results. Real-world experiments guarantee realism but complicate fair comparison. Fair comparison depends on correct interpretation of the results and repeatability of the experiment. Correct interpretation of results is an issue in wireless experiments because it is not easy to record all the factors (e.g., channel conditions, calibration settings of tools and test scenario configurations) that can influence the network performance. Repeatability of experiments is almost impossible because of channel randomness. In wireless experiments, ‘realism’ can be taken for granted but ‘fair comparison’ requires a lot of hardwork and is impossible without a standard methodology. Therefore, we design a workable experimentation methodology which tackles the aforementioned issues as follows:

To ensure correct interpretation of the results, we need to accomplish the following: channel characterization to determine the exact channel conditions, calibration of tools to avoid pitfalls, a simple mechanism to specify scenario configurations. Channel conditions such as path loss, fading and interference are a direct result of radio propagation, movement of objects and co-existing WiFi networks/devices in the environment respectively. Pitfalls mainly result from imperfections/bugs or wrong configurations of a tool. Scenario description consists of a precise specification of the sequence of steps and tasks to be performed at each step. Tasks include traffic generation, packet trace capture (using a sniffer), RF trace capture (using spectrum analyzer) and system/network workload collection. Correct interpretation of results requires that all this information be organized and presented in an easily digestible way to the reviewer. We propose Full disclosure report (FDR) for this purpose.

Repeatable experimentation requires additional work. As repeatability is impractical in the wild wireless environment, we propose statistical repeatability of results where experiments are clustered based on the similarity of networking conditions (channel conditions, station workload, network traffic load) and scenario configurations. Then, it is possible to make a comparison based on the similarity of conditions.

Providing tools to allow a user-friendly mechanism to apply the methodology is also equally important. We need tools to easily describe scenarios, manage scheduling and large number of runs (possibly hundreds or thousands) of them. We also need tools to manage huge amount of packet trace data, metadata and provenance (chronological record of measurement and analysis steps) of results (figures, tables, graphs etc.). Therefore, in addition to the methodology, we developed a toolbox for wireless experimentation and carried out two case studies to validate the methodology. In short, we present a holistic view of benchmarking in wireless networks, formulate a methodology complemented by tools and case studies to help drive future efforts on benchmarking of protocols and applications in wireless networks.

RÉSUMÉ

L’objectif principal de cette thèse est d’obtenir une comparaison réaliste et équitable des performances des protocoles et des applications pour les réseaux sans fil. Dans la communauté réseau, la simulation est l’approche prédominante pour l’évaluation comparative des protocoles, cependant elle manque de véritable car elle utilise le plus souvent des modèles simplifiés des couches de communication. D’un autre côté, les expérimentations sans fil effectuées dans le monde réel sont réaliste mais elles compliquent fortement la comparaison équitable des protocoles. En effet, la comparaison équitable des protocoles dépend de l’interprétation correcte des résultats et de la répétabilité de l’expérimentation. L’interprétation correcte des résultats est un problème majeur pour les expérimentations sans fil car il n’est pas facile de tenir compte de l’ensemble des paramètres qui peuvent avoir un impact sur les performances des protocoles en particulier, les conditions du canal, les paramètres de configuration des outils de mesure). Avec les expérimentations sans fil, la répétabilité des résultats est quasiment impossible à obtenir en raison du caractère aléatoire du canal de transmission. Quant à la “comparaison équitable” des protocoles, elle est complexe à obtenir et nécessite une méthodologie standard.

Dans ce thèse, nous proposons une méthodologie d’expérimentation dont l’objectif est d’assurer une interprétation correcte des résultats expérimentaux. Cette méthodologie est composée des étapes suivantes: caractérisation de canal pour déterminer les conditions exactes de canal; calibrage des outils de mesure, spécification de la configuration du scénario à l’aide d’outils simples. Dans les expérimentations sans fil, les ondes radio sont affectées par des phénomènes multiples et complexes comme l’atténuation en fonction de la distance, les réflexions sur le sol ou les murs qui peuvent provoquer l’évanouissement du signal. De plus, le déplacement d’objets ou de personnes entre émetteur et récepteurs et la proximité d’autres réseaux sans fil peuvent introduire des interférences que l’on ne peut pas ignorer. D’autre part, les expérimentateurs peuvent facilement faire des erreurs si les outils et logiciels utilisés sont
mal configurés. Pour cela, il est important de spécifier de manière précise et détaillée le scénario d'expérimentation. Cette spécification doit inclure la séquence des étapes et la description des tâches à effectuer à chaque étape de l'expérimentation. Les tâches comprennent la génération de trafic, la capture des paquets (e.g. à l'aide de sondes), la capture de traces RF (e.g., avec un analyseur de spectre) et la collecte d'autres mesures, comme la charge CPU des machines utilisées. Pour interpréter correctement les résultats, comme ces informations peuvent être très volumineuses, elles doivent être organisées et présentées de manière efficace. Dans cet objectif, nous proposons l'établissement d'un rapport détaillé appelé FDR ("Full Disclosure Report").

Comme la répétabilité des expérimentations sans fil est impossible à obtenir dans un environnement non contrôlé, notre objectif est de pouvoir répéter des résultats statistiques. Ces derniers sont obtenus en regroupant les résultats d'expérimentations multiples qui ont eu lieu avec des conditions d'environnement similaire. Etant donné que les expérimentations sans fil sont fastidieuses à réaliser et sujet aux erreurs humaines, nous avons développé une boîte à outils permettant de faciliter le benchmark. Ces outils permettent de décrire les scénarios, gérer la planification d'un grand nombre d'expérimentations et traiter l'énorme quantité de traces qui en résultent, sauvegarder les métadonnées et leur provenance (enregistrement chronologique des étapes de mesure et d'analyse), ainsi que les résultats d'expérimentation. Enfin, nous illustrons et validons notre méthodologie de benchmark avec deux cas d'étude.