Understanding 802.11 Performance in Heterogeneous Environments

Kaushik Lakshminarayanan, Srinivasan Seshan, Peter Steenkiste
Carnegie Mellon University
{kaushik, srini, prs}@cs.cmu.edu

Abstract

The availability of unlicensed spectrum coupled with the increasing popularity of wireless communication has given rise to a diverse range of wireless technologies that compete for spectrum. In particular, 802.11 devices face a host of problems such as interference with other 802.11 devices (hidden terminals) as well as with technologies like Bluetooth and ZigBee. Understanding how the medium is utilized and inferring the cause of interference, based on observations from a single wireless node, is hard. Past work has used monitoring infrastructures to detect interference between 802.11 nodes in enterprise networks. In this paper, we try to answer the question: "how can we enable users to reason about wireless performance variations without requiring elaborate instrumentation and infrastructure support?". We propose *WiMed*, a tool that uses only local measurements from commodity 802.11 NICs (at the node being diagnosed) to construct a time map of how the medium is utilized. We have implemented a WiMed prototype using the MadWifi driver for Atheros NICs. Early results show that WiMed is useful and can characterize non-802.11 interference better than existing systems.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: [Measurement Techniques]; C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network monitoring*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Communication*

General Terms

Measurement, Experimentation, Performance

Keywords

wireless networks, heterogeneous environments, monitoring, diagnosis, performance, 802.11, Bluetooth, interference

1. INTRODUCTION

The dependence on wireless technologies has been increasing continuously over the last decade. Unfortunately, the behavior and performance of wireless links is often highly dependent on the surrounding environmental conditions. For example, interference from nearby transmitters sharing the same spectrum impacts performance. Chaotic environments (e.g., home) are especially challenging as they lack experts that can manage wireless networks. Despite the widespread use of wireless networks, an 802.11 user has few tools to help explain unpredictable wireless performance let alone address any problems.

Research on tools for wireless performance diagnosis have produced two types of solutions: infrastructure designs for monitoring wireless networks, and tools for addressing configuration failures. Our work is complementary to efforts that diagnose configuration-based problems [1, 8, 2]. Instead, we aim to build a standalone tool for the user to understand wireless performance variations, which is more related to the former direction of research. Existing wireless monitoring infrastructures have been designed in the context of diagnosing performance problems in enterprise wireless networks [10, 9, 7, 18]. These architectures use monitors throughout the enterprise network and try to get a unified view of the environment by combining the different views provided by these monitors. Another direction of research [1, 8, 19] uses wireless clients around the client being diagnosed to identify causes of wireless problems. There has also been work on identifying hidden and exposed terminals in 802.11 using conflict graphs [12, 3].

Most of the above research has been based on the premise that nodes experiencing performance fluctuations cannot do meaningful diagnosis in isolation. While using a group of nodes (peers, special purpose monitor nodes) simplifies this process, this is not always practical (e.g., in residential WLANs) due to the overhead in setting up and managing the distributed monitoring infrastructure. Moreover, previous work has largely focused on interference in WiFi-only networks and it does not provide a way for users to understand what is happening in the wireless ether. Recent work [6] attempts to perform cross-protocol interference diagnosis, but the preliminary work presented does not identify interference from different technologies.

We make the following contributions in this paper:

1. We have built a tool that provides a time-domain view of how the medium is used in a given 802.11 channel based on local observations alone.

2. We present a study of how physical layer properties such as bit error patterns and medium busy times measured using commodity 802.11 NICs can be used to identify the cause of interference from non-802.11 devices.

We first provide a taxonomy for classifying the broad range of problems in the space (Section 2). We then explain the WiMed

HomeNets'11, August 15, 2011, Toronto, Ontario, Canada.

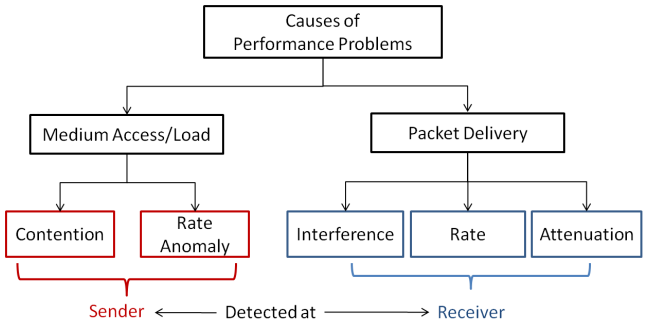

Figure 1: Causes of 802.11 performance problems

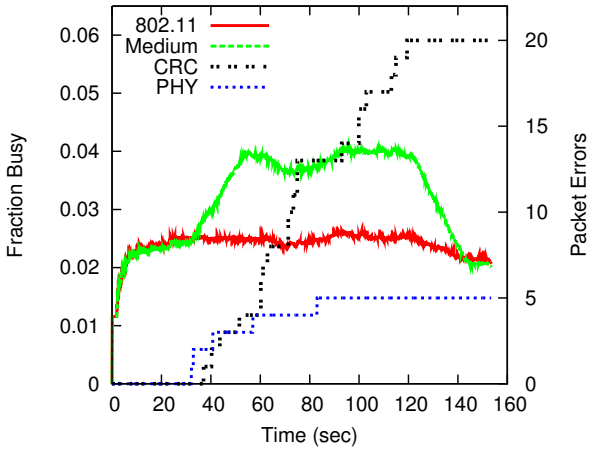

Figure 2: Bluetooth interference (30 to 120 sec) with 802.11 causing CRC and PHY errors

architecture in Section 3. Section 4 evaluates parts of our system and Section 5 summarizes our work.

2. TAXONOMY OF CAUSES

We account for the performance degradation that 802.11 clients can experience by breaking down time utilization into two classes – medium access and packet delivery. This is based on the observation that 802.11 goodput depends on, first, the opportunity to send a packet (CSMA/CA) and once an opportunity is there, the efficiency of packet delivery. This classification is critical to understanding the wireless performance experienced by clients in an 802.11 LAN. We use a time allocation map for the ether (in the channel used by the 802.11 device) based on this classification to understand 802.11 performance. Figure 1 shows some of the problems that could be faced by 802.11 nodes. In this paper, we focus mainly on detecting interferers (packet delivery).

2.1 Medium Access

802.11 nodes need to gain access to the medium using CSMA/CA in order to send a packet. Thus, interframe spacing (contention slots) and time utilized by other nodes competing for spectrum contribute towards medium access time. Other transmissions (802.11 and non-802.11) that do not compete with transmissions of the link in question do not contribute adversely towards performance of the link, although they add to the time utilization of the medium. In terms of causes for medium access problems, 802.11 nodes face contention from other 802.11 nodes as well as from non-802.11 nodes depending on the CCA (Clear Channel Access) mode used. Other nodes can also delay access to the medium for a sender by using unnecessarily low transmit rates, which is referred to as *rate anomaly* problem. Finally, broadcast traffic and misbehaving 802.11 nodes can cause unfairness in channel access.

2.2 Packet Delivery

Once an 802.11 sender gets access to the medium, it has to use the opportunity efficiently. This depends on two parameters – *the probability* that the packet is delivered successfully to the receiver, and the *data rate* used (for now, we do not consider the acknowledgment). Assuming the SINR model and constant thermal noise, probability of packet delivery for a given data rate depends on signal strength at the receiver (after attenuation or fading) and the presence of interferers (802.11 and non-802.11). In this paper, we focus on interference and break up time into successful receptions (useful time) and unsuccessful receptions (wasted time). Further, unsuccessful receptions due to interference is subdivided into 802.11 and non-802.11 interference. We also devise mechanisms to detect specific interference sources using local measurements from commodity 802.11 NICs. However, this paper does not address other causes of inefficiency such as problems with rate adaptation, overprotective APs [10] and attenuation. We believe some of the techniques that have been already developed for detecting these problems can be used by WiMed.

Prior research [11, 14] has shown that commodity 802.11 cards are vulnerable to both wide-band interference (e.g. microwave ovens) and narrow-band interference (e.g. Bluetooth). There has also been a lot of research on studying 802.11 hidden terminal problem. While hidden terminals are not common in 802.11 infrastructure deployments, heterogeneous environments that include devices with asymmetric transmit power levels are more likely to face such hidden terminal scenarios (which cannot be avoided by carrier sense) [17]. We performed a controlled experiment on a wireless emulator [13] to find the impact of Bluetooth interference in a setup where we had a Bluetooth link causing interference at an 802.11 receiver. Figure 2 shows that 802.11 performance is affected considerably in the presence of Bluetooth interference, causing both CRC errors and PHY errors (as reported by the NIC). It is also easy to see that the medium utilization tracking mechanism used in WiMed (top curve) shows non-802.11 activity when compared to the MAC busy time calculated using the length of 802.11 frames (bottom curve) between 30 and 120 sec.

3. WiMed FRAMEWORK

The goal of WiMed is to allow 802.11 users to identify performance problems in heterogeneous wireless environments (e.g., interference) with minimal overhead. To this end, WiMed has been built to use local measurements alone to provide a meaningful view of the usage of the wireless medium to the user. WiMed also detects sources of interference (including non-802.11 sources) to try to help the user solve performance problems more easily. The WiMed framework consists of a *passive monitoring and trace merging* stage, an *analysis* stage and a *time allocation* stage as shown in Figure 3.

3.1 Passive Monitoring

The passive monitoring stage records two traces – *packet traces* with all overheard 802.11 packets and *kernel log traces* with timestamped values of relevant registers on the NIC. The traces are collected using a secondary 802.11 NIC connected to the node being diagnosed. This card is used in *monitor* mode tuned to the channel used by the primary 802.11 card. Thus, the secondary card over-

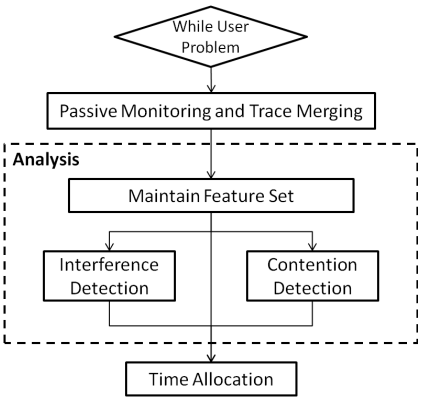

Figure 3: Illustration of WiMed framework

hears all 802.11 packets in that channel that can be decoded fully or partially (i.e., with CRC and PHY errors). We require the second NIC only to overcome limitations in current driver architectures that do not enable us to perform monitoring with high fidelity on a NIC that is already associated with an access point. This setup allows us to track the following properties:

- Signal strength, transmission rate, MAC sequence number and length of all 802.11 frames received
- Type of packet errors (e.g., CRC, PHY errors)
- Medium-busy periods as sensed by the Atheros NIC. This is the time during which there are active transmissions on the channel using any technology.
- MAC-busy time. This is the time during which 802.11 transmissions are active. This is computed using packets decoded in the packet trace.
- 802.11 clients that can be heard

Medium-busy times are recorded as values of a NIC register [5] that keeps track of the cycles (of the NIC clock) when the medium was sensed as busy (energy detection). In order to synchronize the kernel log trace with the packet trace, we record the corresponding driver timestamp also. We track this register at the granularity of two microseconds to get an accurate picture of the channel usage, and record only start and end of busy regions.

3.2 Trace Merging

The tcpdump and the kernel log traces are synchronized and merged to get a unified view of the wireless activity in the medium. Here, we compare busy regions identified by the register (from kernel log) with the packet reception regions (from tcpdump) and produce a trace with 802.11 packets and medium busy regions (excluding the time periods with decoded packets) interspersed. We also annotate the medium busy regions based on whether they overlap with the 802.11 packets or not. This annotation is used in the analysis to determine the presence of non-802.11 transmitters. We can also use this annotated information to find the start or end times of the interferer's transmission whenever there is a collision, though this is not currently implemented in our prototype.

3.3 Analysis

This stage uses the merged trace as input to extract the features mentioned in Section 3.1. It then outputs the confidence level for the existence of performance problems and their causes based on specific detectors. In the trace merging stage, we identify regions where medium was sensed as busy but no 802.11 packets were present in the packet trace. In the analysis stage, this is used as

Procedure 1 Interference detection (Passive)

fineErrHigh: isFEH($hashCrcErr, hashPhyErr$)
myFineErrHigh: isFEH($hashMyCrcErr, hashMyPhyErr$)
  if myfineErrHigh **then**
    if nonWiFiTimeHigh **then**
      report("interference", "non-WiFi")
    else
      report("interference", "WiFi")
    end if
  else
    if fineErrHigh **then**
      report("potential interference", NULL)
    end if
  end if

Procedure 2 Contention detection

currStart: Start time of the current frame
currEnd: End time of the current frame
prevEnd: End time of the previous frame
nextStart: Start time of the next frame
  if isDIFSpluskTS(currStart - prevEnd) **then**
    conTentionTime + = (currPktDuration + IFS)
  else
    if isDIFSpluskTS(nextStart - currEnd) **then**
      contentionTime + = (currPktDuration + IFS)
    end if
  end if
  if contentionTime / totalTime > threshold **then**
    report("contention", NULL)
  end if

an indicator of the level of non-decodable energy present in the medium. We calibrate the environment during a relatively quiet period when there are no non-802.11 devices active (e.g., nights) to quantify the default level. We perform the calibration to take into account the fact that there could be packets with low RSSI that are detected partially by the medium-busy register but not decoded by the NIC. During analysis, as this level increases, we increase the confidence in the presence of a non-802.11 source.

Procedures 1 and 2 give the methods for detecting interference and contention in passive mode. Confidence value is calculated in a similar manner (details in [16]), but instead of giving a Boolean output based on a threshold, we map it to the interval $[0, 1]$ based on deviation from a threshold. In Procedure 1, $hashCrcErr(rate)(ss)$ represents the number of errors found in all packets received at $rate$ Mbps (including packets that were not sent to the node being diagnosed) with signal strength between $ss - \delta$ and $ss + \delta$ dBm. $hashMyCrcErr(rate)(ss)$ represents the number of errors found in packets destined to the client in question, received at $rate$ Mbps with signal strength between $ss - \delta$ and $ss + \delta$ dBm. Also, isFEH() finds if any of the elements of the error table exceeds a threshold. Method isDIFSpluskTS() checks to see if the packet spacing is equal to DIFS + $k\times$Slot_Time in which case, the packet is marked as one that contended with some other node.

Further, we use Bit Error Timing Analysis (BETA) to identify the presence of interference due to specific non-802.11 sources. Whenever we see a packet (destined to the node being diagnosed) that has been corrupted, we try to find a copy of the packet that was retransmitted and received successfully. Once we do that, we identify the bit errors in the packet that was corrupted to form bit error peaks. These peaks are regions that correspond to transmissions of the interfering source. Using these *inferred* interfering transmissions, we do a timing analysis similar to what is done in RFDump [15]. The timing analysis uses known protocol timing parameters such as Bluetooth time slots (625 μsec) and microwave oven's AC

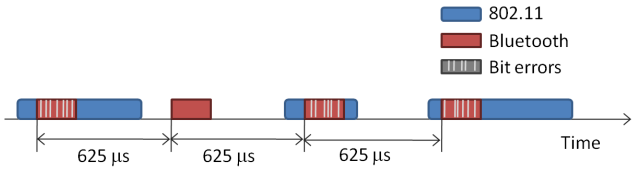

Figure 4: Bit error timing analysis (BETA) for Bluetooth

RXI-U-NI	Packets received by the node (RX) and destined to the node (I) that are unsuccessful/have errors (U) and the cause is identified as non-Wifi interference (NI) that is not Bluetooth
RXI-U-WI	Similarly for Wifi interference
RXI-S	Packets successfully (S) received by the node and destined to the node
TX	Packets transmitted by the node
IFS	Time attributed to interframe spacing
RXO-C	Packets received by the node (RX), but not destined to the node (O), identified as packets that had an IFS preceding/succeeding the packet (instead of idle time)
RXO-NC	Packets that do not have an IFS-based spacing preceding or succeeding it
NW	Time attributed to signal that could not be decoded by the NIC (identified as medium busy, but not decoded)

Table 1: Notation used for time allocation

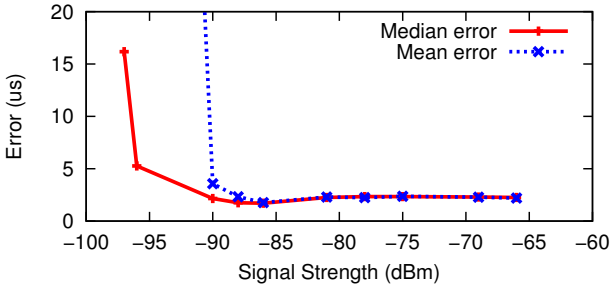

Figure 5: Accuracy of register tracking for 802.11

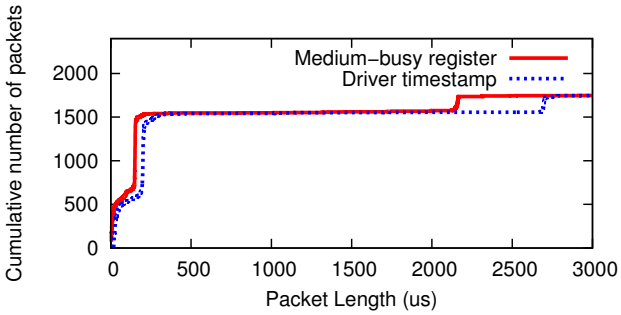

Figure 6: Accuracy of register tracking for Bluetooth

frequency timing. Figure 4 illustrates how BETA works in the case of a Bluetooth interferer.

While we have only implemented detectors for interference and contention, WiMed can use existing mechanisms to detect other causes for performance fluctuations. For example, since we record signal strength of packets, attenuation can be detected whenever received signal strength drops below a threshold. As we maintain statistics of packet loss rates for different received signal strengths, 802.11 rate-related problems can also be detected.

3.4 Time Allocation

WiMed's analysis stage outputs the confidence values for the presence of certain problems like interference (Bluetooth, Microwave oven, etc.) and contention. Though the above output provides information about what could be the cause for performance degradation, it does not tell the user the extent to which it affects the throughput. As a first step towards this goal, we have built a performance monitoring tool to generate a time break down for various events in the wireless medium. This *time allocation* tool uses the detection results to appropriately mark regions in time with events such as packet transmission, non-802.11 transmission and successful 802.11 packet reception (see Table 1). This helps the user understand the impact of problems, as we will show in Section 4.2.

4. EVALUATION

The monitoring component of WiMed was built into the *MadWifi* driver for Atheros chipset. We used an Atheros PCMCIA card based on the AR5212 chipset as the secondary NIC (in *monitor* mode). We used *tcpdump* to get the packet trace along with the entire payload. We have modified the driver to record the frame length (from the frame descriptor in the driver) through the *radiotap* header and correspondingly modified *tcpdump* to be able to read it. The register `AR5K_PROFCNT_RXCLR` was read every 2 μs to track the medium busy time. In this section, we first evaluate the accuracy of the medium-busy register in detecting the presence of 802.11 and non-802.11 sources. Then we show results for performance monitoring and interference detection.

4.1 Accuracy of Sensing

We evaluated the effectiveness of register tracking mechanism at detecting 802.11 and Bluetooth transmissions. The following experiments were done on a wireless emulator testbed [13].

First, we sampled the value of the register at successive 802.11 packet receptions and used the difference of these values to calculate the packet lengths as seen by the register. In the emulator, we set up experiments with a single link and varied the loss on the link to see the accuracy of the register for different received signal strength values. Figure 5 shows that the error in packet lengths reported by this register is negligible for signal strength greater than -90 dBm. In order to show that tracking the register works even at a finer granularity (smaller packets), we did an experiment where we tracked this register at the granularity of 20 μs from the start to the end of several packets which spanned a few milliseconds. We then calculated the difference between the actual utilization using timestamps ($t_2 - t_1$) and the utilization computed using register tracking ($reg(t_2) - reg(t_1)$)). We found that the mean error in medium utilization computation using register tracking was only 0.1 μs (0.5 %).

We also conducted a similar experiment for Bluetooth (but used only the lowest loss that can be set on the emulator). We used `l2ping` utility to send 800 pings of length 2600 μs (and an equal number of MAC-level acknowledgments). Figure 6 shows the cumulative distribution of the packet lengths as reported by an 802.11 monitor node using register tracking. The two curves correspond to packet lengths calculated using medium-busy register values ($reg(t_2) - reg(t_1)$) and timestamps ($t_2 - t_1$), where t_1 and t_2 are the starting and ending timestamps of a frame found using energy detection (based on the medium-busy register values). We see clear *steps* at 150 μs/190 μs and 2150 μs/2700 μs corresponding to the L2CAP

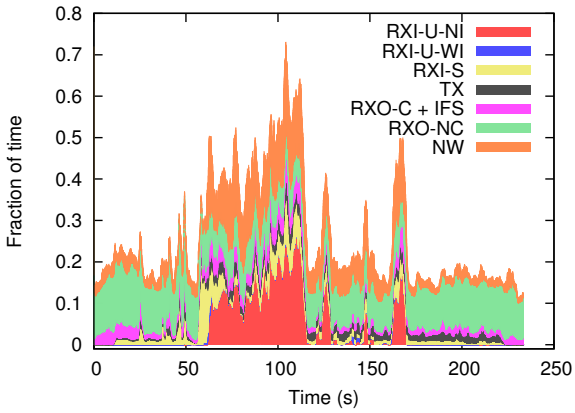

Figure 7: Time utilization map of the wireless environment for the Bluetooth interference experiment

pings and the MAC-level acknowledgments. The number of ping-sized frames (and ACK-sized frames) reported is roughly as expected (i.e., 200 out of 800 as an 802.11 channel occupies 20 MHz out of the 79 Bluetooth channels). However, the reported length of both these frames are shorter than the expected length (by about 20 %) when using the former method. This indicates that the energy detection as performed by the 802.11 receiver of a Bluetooth signal (narrow-band) is not accurate, but at the granularity of a packet, using the latter method works well.

4.2 Performance Monitoring

We have incorporated a performance monitoring tool in WiMed that provides a time break-up of how the medium is utilized. Table 1 gives the notation used in Figure 7. For preliminary evaluation, we used a laptop with two 802.11 NICs (Intel and Atheros chipsets), one in *managed* mode and the other in *monitor* mode (for WiMed). In the experiment, the NIC in managed mode pinged a machine in the LAN at the rate of 10 pings a second using 1464 byte frames. To induce Bluetooth interference, we made the Bluetooth radio on the laptop transmit (using *l2ping* utility) to a mobile phone.

Figure 7 shows the time allocation graph for the Bluetooth interference experiment. The curves corresponding to `RXI-U-NWI` and `NW` occupy about 20% of the medium each during the interference period. This shows that the interference was strong enough to affect the throughput considerably. Figure 8 shows how the confidence values of the interference detectors change over time. We see that the Bluetooth detector has fairly high confidence about the presence of a Bluetooth interferer between 60 sec and 160 sec, during which the Bluetooth radio on the laptop was active. Although the microwave detector also has some confidence about the presence of microwave, it is significantly lower than that of the Bluetooth detector. On the other hand, WiMed has very low confidence for the presence of an 802.11 interferer. This is more important as it is able to differentiate between an 802.11 interferer and a non-802.11 (Bluetooth) interferer. The graph has intermediate spikes instead of a continuous line because the confidence value peaks only when there are packet reception errors. Though the confidence value is exponentially averaged, the decay factor is set high to ensure the detection confidence does not become stale and to show when exactly performance problems were faced by the system.

In another experiment, we induced an 802.11 hidden terminal scenario. This was done by using a modified driver [4] that disables clear channel assessment, backoff behavior etc., so that the 802.11 node will behave like a hidden node. The diagnostic laptop was placed a few feet from the *hidden* laptop. The hidden laptop sent broadcast pings at the rate of 20 per second. Figure 9 shows that interference from the hidden terminal (RXI-U-WI) did not affect performance as much as contention (RXO-C + IFS). This is consistent with the fact that there were only 184 corrupted packets (destined to the node) in the 802.11 experiment compared to 1607 in the Bluetooth experiment.

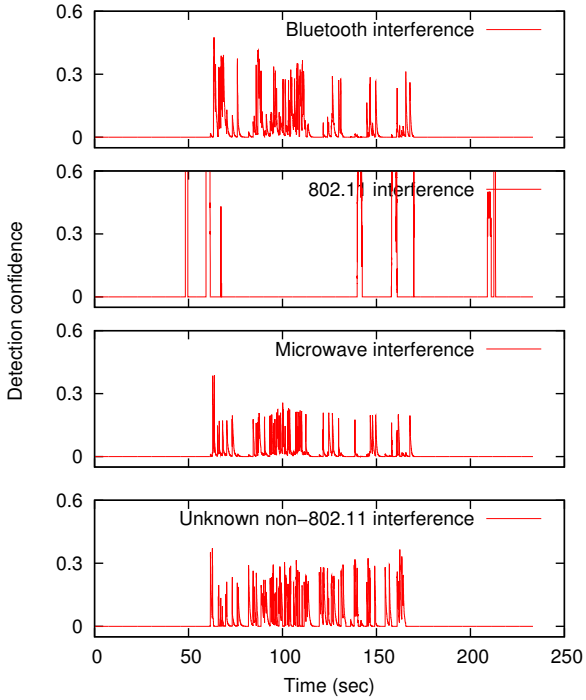

Figure 8: Bluetooth interference ($\sim$ 60 to 160 sec) as detected by WiMed

Figure 10 shows the confidence values of the interference detectors for a real-world experiment using a microwave oven as an interferer, placed at a distance of about 6 feet from the laptop. More results on unknown interferer detection and contention detection are presented in the longer version of the paper [16].

5. CONCLUSION

In this paper, we presented the design and implementation of WiMed, a tool for understanding 802.11 performance without the use of dedicated infrastructures in heterogeneous environments such as the home. WiMed uses commodity 802.11 NICs to produce a time allocation map showing how the medium is used. It is also able to detect non-802.11 sources of interference using NIC registers and bit error analysis. While this is only a first step towards infrastructure-less diagnosis, we believe that adding such monitoring capability to APs and clients could significantly alleviate the problems faced by wireless users, especially in chaotic environments such as home.

Acknowledgments

We thank the anonymous reviewers for their valuable suggestions and comments. This research was funded in part by NSF under award numbers CNS-0434824 and CNS-0520192.

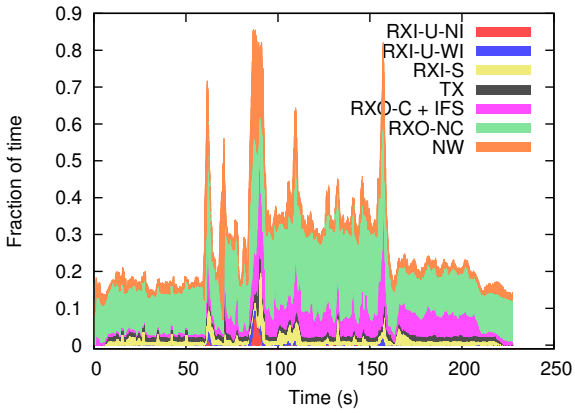

Figure 9: Time utilization map of the wireless environment for the 802.11 interference experiment

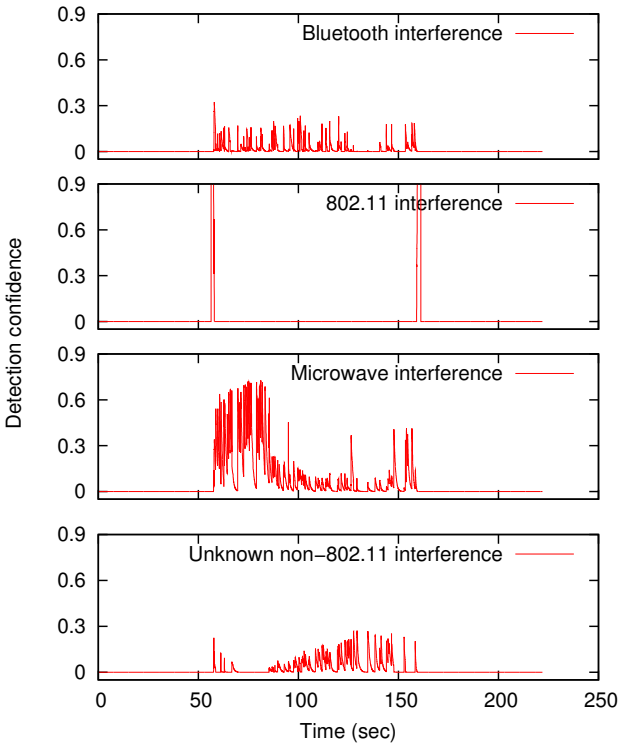

Figure 10: Microwave interference (∼ 60 to 160 sec) as detected by WiMed

6. REFERENCES

[1] A. Adya, P. Bahl, R. Chandra, and L. Qiu. Architecture and techniques for diagnosing faults in ieee 802.11 infrastructure networks. In *MobiCom '04*, Philadelphia, PA, USA, 2004.

[2] B. Aggarwal, R. Bhagwan, T. Das, S. Eswaran, V. N. Padmanabhan, and G. M. Voelker. Netprints: diagnosing home network misconfigurations using shared knowledge. In *NSDI'09*, Boston, MA, 2009.

[3] N. Ahmed, U. Ismail, S. Keshav, and K. Papagiannaki. Online estimation of rf interference. In *CoNEXT '08*, Madrid, Spain, 2008.

[4] E. Anderson, G. Yee, C. Phillips, D. Sicker, and D. Grunwald. Commodity ar52xx-based wireless adapters as a research platform. `http://systems.cs.colorado.edu/wiki/CARP`, April 2008.

[5] P. Aravinda, K. Acharya, A. Sharma, E. M. Belding, K. C. Almeroth, and K. Papagiannaki. Congestion-aware rate adaptation in wireless networks: A measurement-driven approach. In *SECON'08*, 2008.

[6] A. Baid, S. Mathur, I. Seskar, T. Singh, S. Jain, D. Raychaudhuri, S. Paul, and A. Das. Spectrum mri: Towards automated diagnosis of multi-radio interference in the unlicensed band. In *IEEE Wireless Communications and Networking Conference*, 2011.

[7] R. Chandra, J. Padhye, A. Wolman, and B. Zill. A location-based management system for enterprise wireless lans. In *Proc. NSDI'07*, Cambridge, MA, Nov. 2007.

[8] R. Chandra, V. N. Padmanabhan, and M. Zhang. Wifiprofiler: cooperative diagnosis in wireless lans. In *MobiSys '06*, Uppsala, Sweden, 2006.

[9] Y.-C. Cheng, M. Afanasyev, P. Verkaik, P. Benkö, J. Chiang, A. C. Snoeren, S. Savage, and G. M. Voelker. Automating cross-layer diagnosis of enterprise wireless networks. In *SIGCOMM '07*, Kyoto, Japan, 2007.

[10] Y.-C. Cheng, J. Bellardo, P. Benkö, A. C. Snoeren, G. M. Voelker, and S. Savage. Jigsaw: solving the puzzle of enterprise 802.11 analysis. In *SIGCOMM '06*, Pisa, Italy, 2006.

[11] R. Gummadi, D. Wetherall, B. Greenstein, and S. Seshan. Understanding and mitigating the impact of rf interference on 802.11 networks. In *SIGCOMM '07*, Kyoto, Japan, 2007.

[12] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. *Wirel. Netw.*, 11(4):471–487, 2005.

[13] G. Judd and P. Steenkiste. Using emulation to understand and improve wireless networks and applications. In *Proc. NSDI'05*, Berkeley, CA, USA, 2005.

[14] A. Kamerman and N. Erkocevic. Microwave oven interference on wireless lans operating in the 2.4 ghz ism band. *Proc. PIMRC '97*, 3, Sep 1997.

[15] K. Lakshminarayanan, S. Sapra, S. Seshan, and P. Steenkiste. Rfdump: an architecture for monitoring the wireless ether. In *CoNEXT '09*, Rome, Italy, 2009.

[16] K. Lakshminarayanan, S. Seshan, and P. Steenkiste. Wimed: An infrastructure-less approach to wireless diagnosis. Technical Report CMU-CS-11-118, Carnegie Mellon University, Department of Computer Science, June 2011.

[17] C.-J. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis. Surviving wi-fi interference in low power zigbee networks. In *SenSys '10*, Zürich, Switzerland, 2010.

[18] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Analyzing the mac-level behavior of wireless networks in the wild. In *SIGCOMM '06*, Pisa, Italy, 2006.

[19] A. Sheth, C. Doerr, D. Grunwald, R. Han, and D. Sicker. Mojo: a distributed physical layer anomaly detection system for 802.11 wlans. In *MobiSys '06*, Uppsala, Sweden, 2006.