# WiMed: An Infrastructure-less Approach to Wireless Diagnosis

**Kaushik Lakshminarayanan** **Srinivasan Seshan**
**Peter Steenkiste**

June 2011
CMU-CS-11-118

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research is funded in part by NSF under award numbers CNS-0434824 and CNS-0520192.
An earlier version of this paper appeared at the ACM SIGCOMM Workshop on Home Networks, HomeNets'11 [16]

**Keywords:** wireless networks, heterogeneous environments, monitoring, diagnosis, performance, 802.11, Bluetooth, interference

## 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. Inferring the cause of wireless performance problems based on observations from a single node is hard. Past work on wireless diagnosis has made use of monitoring infrastructures with co-ordinated observations from multiple monitors to tackle this problem.
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 physical layer measurements from commodity 802.11 NICs (at the node being diagnosed) to diagnose a variety of performance-related problems. We have implemented a WiMed prototype using the MadWifi driver for Atheros NICs. Our results show that WiMed can diagnose non-802.11 interference better than existing systems. WiMed's time allocation map shows how the medium is utilized, which is useful in identifying performance bottlenecks.

## 1 Introduction

The dependence on wireless technologies has been increasing over the last decade, and wireless links have been replacing wired networks in many scenarios. Unfortunately, the behavior and performance of wireless links is often highly dependent on the surrounding environmental conditions. For example, walls and distance between a transmitter and a receiver result in poor signal strength, which, in turn, results in low bandwidth and high loss rates. Similarly, interference from nearby transmitters sharing the same spectrum can also impact performance. Despite the widespread use of wireless networks, an 802.11 user experiencing performance problems has few tools to help identify the cause of these problems, let alone address them.

This lack of tools is due to three key reasons. First, numerous transmitters using a wide range of link technologies such as 802.11, Bluetooth and ZigBee, share the unlicensed portions of the spectrum like the 2.4GHz ISM band. This leads to unpredictable interference and interactions between devices. Second, many of the performance problems that users encounter are due to problems at the physical layer and are hidden by layers of abstraction in both the system hardware, drivers and operating system. Third, existing wired networking tools such as *traceroute* and *ping* are rarely appropriate since they focus on node-level or routing-level failures and do not deal with link-layer problems.

We argue that an architecture for diagnosing wireless problems (in particular, for 802.11) should delegate as much work as possible to the end-device. If realized, this would eliminate the need for a separate monitoring infrastructure and make wireless diagnosis as easy as diagnosis in wired networks. The diagnosis architecture could also use the access point to help diagnose, as the link that is been diagnosed is between the end-device and the access point. In some cases, the diagnosing ability could be further improved by using the knowledge of other nodes in the network. However, the ability to diagnose should not entirely depend on other wireless nodes or a customized infrastructure. Towards this goal, this paper presents techniques to identify causes of performance problems, in particular, those faced by 802.11 links in heterogeneous environments.

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 to diagnose configuration-based problems ([1, 7, 2], which deal with problems establishing a connection between the client and the AP such as WEP/WPA and DHCP problems. This paper focuses on diagnosing performance problems created by a variety of causes, and has more in common with the efforts to create infrastructures to monitor wireless network operation.

Existing wireless monitoring infrastructures have been designed in the context of diagnosing performance problems in enterprise wireless networks [9, 8, 6, 17]. These architectures use monitors throughout the enterprise network to get different views of the environment and they try to diagnose by combining these views. Another direction of research [1, 7, 19] uses wireless clients around the client being diagnosed to troubleshoot problems. There has also been work on identifying hidden and exposed terminals in 802.11 using conflict graphs [12, 3]. Most of the research in wireless diagnosis has been under the premise that the nodes experiencing performance problems cannot do meaningful diagnosis in isolation. While using a group of nodes (peers, special purpose monitor nodes) simplifies diagnosis, this is not always practical (e.g. in residential WLANs).

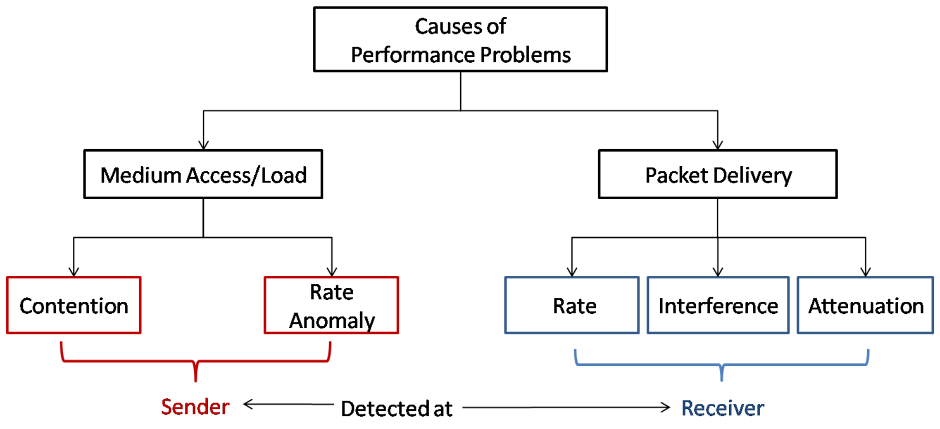


Figure 1: Taxonomy of causes of 802.11 performance problems

Moreover, there is considerable overhead in setting up the monitoring infrastructure for performing coordinated diagnosis. Finally, previous work has largely focused on WiFi-only networks and provides no way for users to understand what is happening in the wireless ether. Recent work [5] attempts to perform cross-protocol interference diagnosis, but the preliminary work presented does not identify interference from specific technologies.

This paper makes the following contributions:

1. We systematically classify causes of wireless problems using a top-down approach in a manner that is amenable to understanding 802.11 performance.

2. We present a study 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.

3. 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.

We first provide a taxonomy for classifying the broad range of problems in the space (Section 2). We then explain the diagnostic framework of WiMed in Section 3 before detailing on the implementation details (Section 4). Section 5 evaluates our diagnosis modules as well as our entire system. We compare and contrast WiMed with past approaches in Section 6 and summarize our work in Section 7.

## 2 Taxonomy of Causes

We account for the performance degradation that 802.11 clients can experience by breaking down the causes of such problems into two classes – medium access and packet delivery (see Figure 1). 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 study this taxonomy in more detail before presenting our diagnosis procedure.

## 2.1 Medium Access and Load-based Problems

802.11 nodes must gain access to the medium using CSMA/CA in order to send a packet. Because CSMA/CA guarantees equal long-term channel access probability, frequency of access to the medium depends on mainly two factors – contention (number of nodes competing with a given node) and the utilization of the medium by other nodes. In the former case, the probability of gaining access to the medium decreases as the number of contenders increases. In the latter case, the frequency of access to the medium goes down if other nodes occupy the channel for a longer time when they gain access.

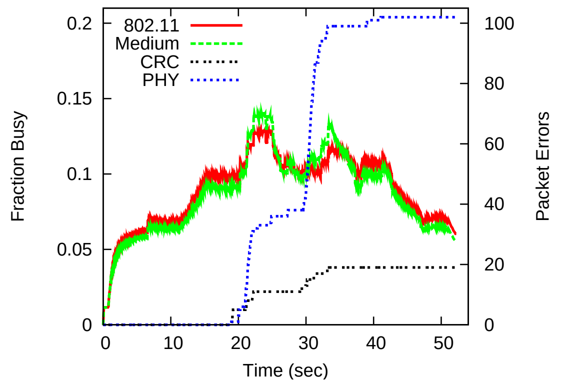
### 2.1.1 Contention

There are several CCA (Clear Channel Assessment) modes defined in the IEEE standard. For instance, mode 1 corresponds to energy detection above a threshold, mode 2 corresponds to detecting any valid 802.11-modulated signal and mode 3 is a combination of modes 1 and 2, i.e. any valid 802.11 modulated signal above a threshold. Considering only 802.11 nodes, 802.11 MAC (with exponential backoff) is packet-fair, i.e. the number of packets a node can send over a period of time is inversely related to the number of contenders in the vicinity of the node. This means that a high density of nodes adversely affects the probability of getting access to the medium for each of these nodes. This may or may not be the desired outcome depending on whether a subset of these nodes are exposed terminals or not. Exposed terminals are those nodes that are within carrier sense range of the sender but not the receiver. This clearly leads to sub-optimal use of the medium. If mode 1 is used, 802.11 nodes will contend for the medium with other RF devices, which increases the time required to gain access to the medium. In scenarios where the non-802.11 devices do not defer or backoff in an 802.11-like fashion, mode 1 could lead to 802.11 always deferring to other RF devices.

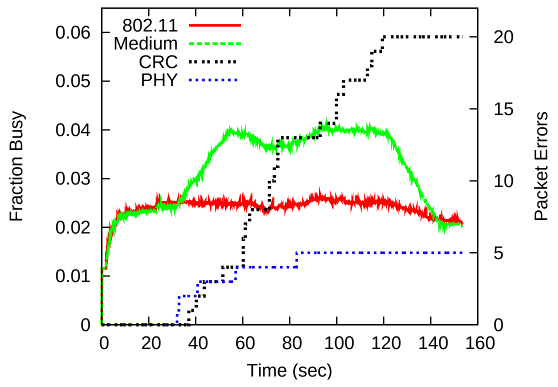
### 2.1.2 Other Causes

While the number of contenders reduces the probability of getting an opportunity to send, other contenders can delay access to the medium for a sender by using unnecessarily low rates. For example, if a node using 54Mbps competes with another node that uses 1Mbps, then the node using 54Mbps will get access to the medium less frequently compared to the case where both nodes are using 54Mbps. This is despite the fact that the nodes get “fair” access to the medium in either situation. This is referred to as the *rate anomaly* problem. While this might not have a concrete remedy (i.e. the node using 1Mbps may be doing so because of low SNR), this still contributes towards inefficient use of the medium and helps in reasoning about the observed performance [10].

The IEEE standard does not require broadcast traffic to perform random backoff. As a result, a high fraction of broadcast traffic could decrease the probability of access to the medium for unicast



(a) 802.11 interference (20 to 40 sec) causing CRC and PHY errors. MAC busy time and medium busy time are almost equal.



(b) Bluetooth interference (30 to 120 sec) causing CRC and PHY errors. Medium-busy time is higher than MAC-busy time indicating non-802.11 signal.

Figure 2: Interference with an 802.11 link due to different sources causing performance drop. 802.11 and non-802.11 interferers can be detected by tracking MAC-busy and medium-busy times.

traffic, leading to *unfairness*. Unfairness could also arise if there are misbehaving nodes that are not following the 802.11 standard for channel access.

## 2.2 Packet Delivery-related Problems

Once an 802.11 sender gets access to the medium, it has to make efficient use of the opportunity. Efficient use of the medium depends on two parameters – the *probability* that the packet is delivered successfully to the intended receiver and the *rate* at which the packet is sent (for now, we do not consider the acknowledgment for the delivered packet). 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).

### 2.2.1 802.11 Interference

Packet delivery probability is considerably affected by interference. Hidden 802.11 terminals can cause interference at the receiver as they are not in the sensing range of the sender. While the hidden terminal problem does not occur frequently, when it does occur, it can cause the throughput of the nodes involved to collapse. We performed controlled experiments on a wireless network emulator [13] using our monitoring infrastructure, where we track the medium-busy time and the MAC-busy time (details in Section 3). Figure 2 (a) shows how performance is affected in the presence of a hidden terminal problem. It is also easy to see that the curves corresponding to medium-busy and MAC-busy times are close to each other due to the absence of non-802.11 signal. These kinds of hidden terminal scenarios can affect even planned deployments as the clients can still be mobile.

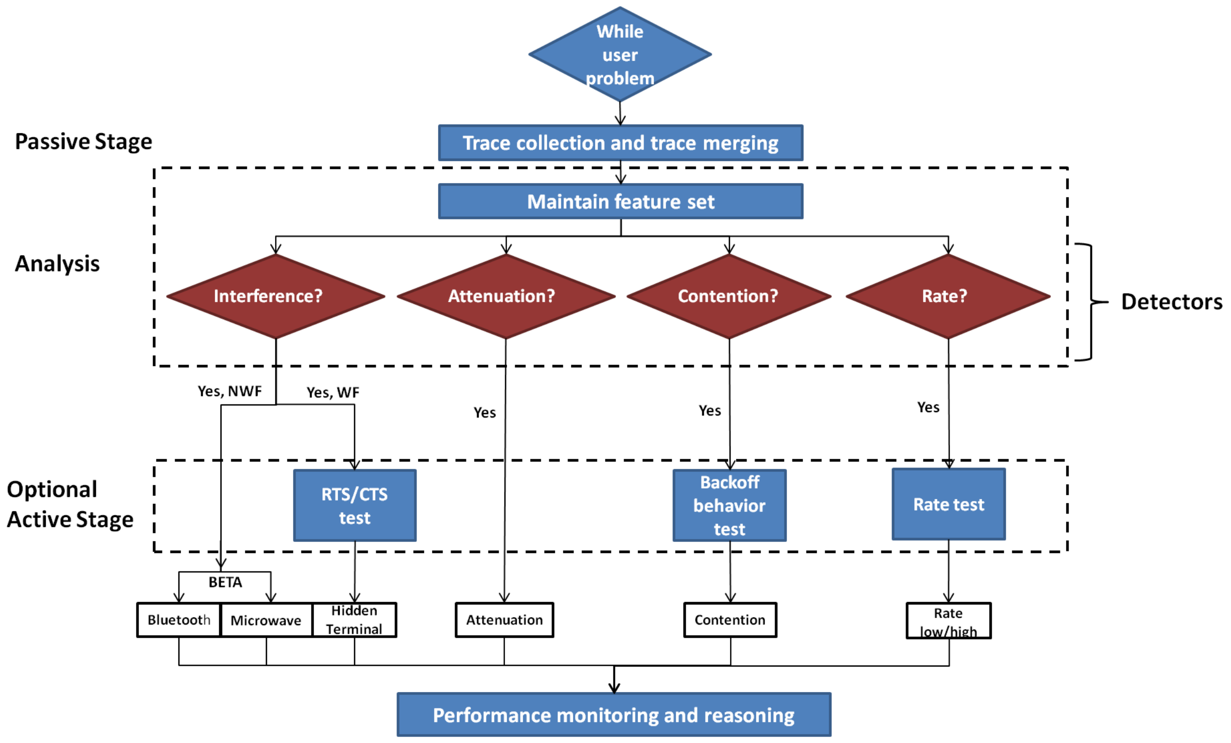


Figure 3: Illustration of WiMed framework

### 2.2.2 Non-802.11 Interference

Prior research [11, 14] has shown that commodity 802.11 cards are vulnerable to both wideband interference (e.g. microwave ovens) and narrow-band interference (e.g. Bluetooth). Unfortunately, these devices share the 2.4GHz ISM band with 802.11 devices and with their increasing popularity over the last decade, tend to be one of the primary causes of interference with 802.11 devices. Again, we performed controlled experiments on the wireless emulator 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 (b) 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.

### 2.2.3 Other Causes

An 802.11 node sending at a rate that is lower than optimal does not utilize the medium efficiently leading to performance drop. In addition, this could lead to the *rate anomaly* problem as viewed by other nodes in the network. On the other hand, when an 802.11 node sends at a rate that is higher than the optimal rate for the SNR of the node at the node's receiver, the bit error rate (BER) goes up, thus wasting the transmission opportunity.

When an 802.11 network has both 802.11b and 802.11g nodes, APs use what is called *802.11g protection mode* [9]. In this mode, the 802.11g nodes in the network send a CTS-to-self frame before sending a packet to ensure backward-compatibility with the 802.11b nodes. Unfortunately, this leads to lower throughput and higher latency for 802.11g clients (similar to using lower rate). Sometimes, APs have conservative timeouts for switching to the normal mode during which all the nodes will be unnecessarily using the protection mode.

Measurements in enterprise WLANs have shown that clients try to associate with APs that are more than 20 meters away even in deployments where the average distance between a client and the nearest AP is only around 10 meters [6]. Since RF signals attenuate with distance, this can lead to links with low SNRs. At very low SNRs, bit error rates increase even if the lowest available rates are used. This leads to inefficient use of the medium for the weak links and other stronger links (rate anomaly). The received signal strength could also be low for other reasons, such as fading due to multipath or obstacles in the environment.

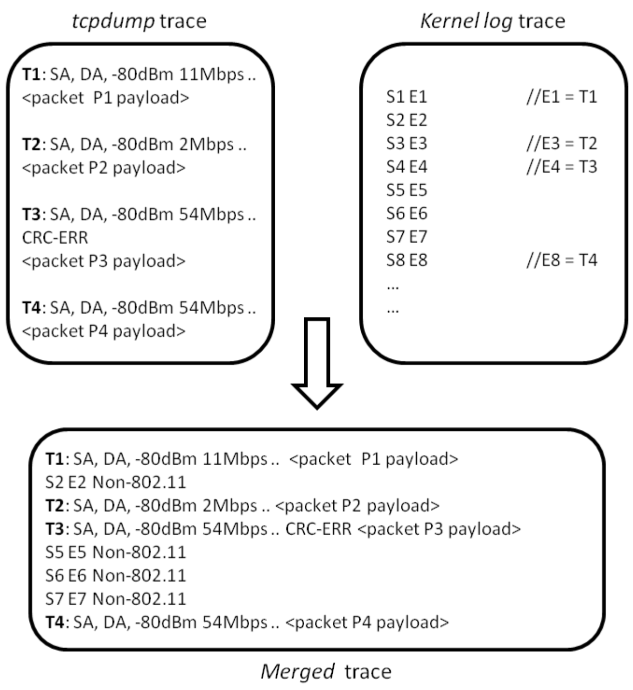
## 3 Diagnosis Framework

The WiMed framework consists of a *passive monitoring* stage and an optional *active probing* stage as shown in Figure 3. The passive monitoring stage attempts to detect the cause of the performance problem, if one exists, using either lightweight coarse-grained or more heavyweight fine-grained techniques. WiMed can then use the output of the passive monitoring phase and choose to do active tests to further improve detection confidence. In this paper, we present mostly results for fine-grained passive monitoring as it gives us the highest fidelity. Active probing allows us to obtain more detailed information, and it does not require additional measurement infrastructure. While active probing can be useful, it has the potential to change the network behavior leading to other problems.

### 3.1 Passive Monitoring

The passive monitoring stage record two traces. *Packet traces* identify overheard 802.11 packets with relevant meta data (e.g. RSSI, etc.). *Kernel log traces* include timestamped values of relevant registers on the NIC. The traces are collected using a second 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 second card overhears all 802.11 packets in that channel that can be decoded partially or fully. We require the second NIC only to overcome limitations in current driver architectures that do not allow us to monitor with high fidelity on a NIC that is already associated with an access point. This setup allows us to track many physical layer features including:

- Signal strength, transmission rate, MAC sequence number and length of frames received
- Type of packet errors (e.g., CRC, PHY errors)
- Medium-busy times as sensed by the Atheros NIC. This is the time during which there are active transmissions on the channel using any technology.



*tcpdump* trace
**T1**: SA, DA, -80dBm 11Mbps ..
<packet P1 payload>
**T2**: SA, DA, -80dBm 2Mbps ..
<packet P2 payload>
**T3**: SA, DA, -80dBm 54Mbps ..
CRC-ERR
<packet P3 payload>
**T4**: SA, DA, -80dBm 54Mbps ..
<packet P4 payload>
*Kernel log* trace
S1 E1 //E1 = T1
S2 E2
S3 E3 //E3 = T2
S4 E4 //E4 = T3
S5 E5
S6 E6
S7 E7
S8 E8 //E8 = T4
...
...
**T1**: SA, DA, -80dBm 11Mbps .. <packet P1 payload>
S2 E2 Non-802.11
**T2**: SA, DA, -80dBm 2Mbps .. <packet P2 payload>
**T3**: SA, DA, -80dBm 54Mbps .. CRC-ERR <packet P3 payload>
S5 E5 Non-802.11
S6 E6 Non-802.11
S7 E7 Non-802.11
**T4**: SA, DA, -80dBm 54Mbps .. <packet P4 payload>
*Merged* trace

Figure 4: A simple case of trace merging. Tn represents packet timestamp, Sn and En denote the start and end timestamp of a medium-busy region.

- 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 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 also record the corresponding driver timestamp. More details on the implementation of the logging procedure are explained in Section 4. In coarse-grained passive monitoring, we record the medium-busy register values at the granularity of a packet event (i.e. whenever we receive a packet), whereas in fine-grained passive monitoring, we track this register at the granularity of a few microseconds. We do this to get a better picture of the channel usage. We also get the entire payload of packets received in fine-grained monitoring instead of just the first 68 bytes in coarse-grained mode. While the wireless node can run coarse-grained monitoring continuously due to minimal overhead, fine-grained passive monitoring is performed only when the user initiates the diagnosis procedure as it has a high overhead.

## 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. The merging procedure is straightforward in the case of coarse-grained monitoring, but it is more involved in fine-grained monitoring. In the latter case, we must 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. An abstract version of the above procedure is illustrated in Figure 4. 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 make use of this annotated information to find the start or end times of the interferer's transmission for packets that get corrupted due to interference, though this is not currently implemented in our prototype. As detailed later in Section 4, the medium busy regions in the merged trace do not always indicate the presence of non-802.11 transmitters.

## 3.3 Analysis

The diagnostic procedure 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.

First, packet-level features such as signal strength and transmission rate are directly extracted from the merged trace. Then, features such as the cumulative medium busy time and loss rates are maintained using a sliding window to average out spikes in activity. Finally, at each packet event, this sliding window is updated and a set of confidence values is calculated for each of the high level problem indicators such as interference, presence of non-802.11 activity and contention.

We use the medium-busy register to track regions where the medium was sensed as busy. 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, as explained above. In the analysis stage, this is used as 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 very low RSSI that are detected partially by the medium-busy register but not decoded by the NIC. During diagnosis, 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 calculation is along the same lines, 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.

Interference detection in WiMed uses Bit Error Timing Analysis (BETA) to identify the presence of interference due to specific non-802.11 sources. Whenever it sees a packet (destined to the node being diagnosed) that has been corrupted, it tries to find a copy of the packet that was

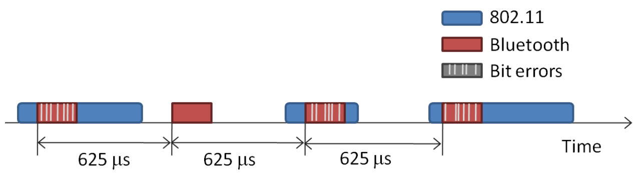


Figure 5: Bit error timing analysis (BETA) for Bluetooth

retransmitted and received successfully. Once this is done, it identifies the bit errors in the corrupted packet to form bit error peaks. These peaks are regions that correspond to transmissions of the interfering source. Using these *inferred* interfering transmissions, BETA is used to determine the protocol of the interfering transmission (similar to what is done in RFDump [15]). The timing analysis uses known protocol timing parameters such as Bluetooth time slots (625 $\mu$sec) and microwave oven's AC frequency timing. Figure 5 illustrates how BETA works in the case of a Bluetooth interferer. We believe that BETA can also be used to detect the presence of 802.11 interference when used in conjunction with traces from multiple nodes in the environment. However, in our current prototype, we detect 802.11 interference through elimination only (i.e., interference without the presence of non-802.11 interferers means it is 802.11 interference).

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

**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
```

## 3.4 Confidence Metrics

We use different metrics for expressing the confidence in the presence of performance problems as well as the cause of the problem. For example, there is a confidence level associated with both the presence of non-802.11 interference and the cause being a Bluetooth interferer (which can be combined to express the confidence of the problem being Bluetooth interference). These values reflect on a scale of 0 to 1, how confident WiMed is, in the particular cause for an existing problem. Suppose $c_{\beta-bt}, c_{\beta-\mu w}$ denote the confidence metric value for the existence of timing patterns in bit errors corresponding to Bluetooth and microwave respectively, and $c_{if}, c_{nwf}$ denote the confidence metric values for the presence of some interference (based on one of the three cases in Procedure 1) and the presence of a non-WiFi transmitter. If $C_{BT}, C_{\mu W}, C_{WF}, C_{cont}$ represent the instantaneous confidence metric values for Bluetooth interference, microwave interference, 802.11 interference and 802.11 contention respectively, then we define:

$$C_{BT} = c_{if}.c_{nwf}.c_{\beta-bt}, \tag{1}$$

$$C_{\mu W} = c_{if}.c_{nwf}.c_{\beta-\mu w}, \tag{2}$$

$$C_{WF} = c_{if}.(1 - c_{nwf}), \tag{3}$$

$$C_{cont} = (1 + \frac{numCl}{maxCl}).\frac{contTrafTime}{nonContTrafTime}.w^2 \tag{4}$$

where,

$$w* = \left(5.\frac{contTrafTime + nonContTrafTime}{totalTime}\right) - 0.5, \tag{5}$$

$$w = \begin{cases} 0 & \text{if } w* < 0, \\ w* & \text{if } w* \in [0,1], \\ 1 & \text{if } w* > 1. \end{cases}$$

The above equations 1, 2, 3 are similar to probability estimation based on independence assumption of the relevant variables in the expression. Equation 4 weights the current fraction of non-contention to contention time utilization (calculated based on IFS and corresponding packets involved) by a quadratic term which equals a scaled value of the current MAC utilization (mapping $[0, 0.1)$ to 0, $[0.1, 0.3]$ to $[0, 1]$ scaled linearly and $(0.3, 1]$ to 1).

## 3.5 Active Probing

We do not evaluate active diagnosis in this paper, but here we outline how it could be used to make the detection slightly more robust. After getting clues from the passive monitoring phase, we could use active probing to find the cause at a finer granularity in order to:

- Determine more rigorously the source of interference using dense probes. Dense probes can induce more packet errors, which in turn can increase our confidence in the detection outcome. Also, the user can move the device (spatially) to change the RSSI and then, observe the effect on performance. On the other hand, we can also reduce our RSSI and determine more accurately the start of interfering transmission as the bit error rate is expected to be higher.

- Find at a finer granularity the contending nodes in 802.11 by making sure we have at least one packet in the transmission queue at any point of time and study our backoff behavior.

- Determine the optimal rate to use by trying different rates and calculating the throughput directly (instead of predicting using metrics)

- Determine hidden terminals by repeating experiments with RTS/CTS both enabled and disabled.

## 3.6 Performance Monitoring and Reasoning

WiMed outputs the confidence values for the presence of certain specific problems based on the metrics described in Section 3.4. In the process of communicating these confidence levels, WiMed uses some *history* (based on sliding window) to weight the confidence values accordingly. For instance, if on most occasions, presence of a Bluetooth interferer is detected, then a smaller weight is given to the confidence value for the detection of a Microwave interferer as it is most likely a false positive (though this is theoretically possible when both interferers are present).

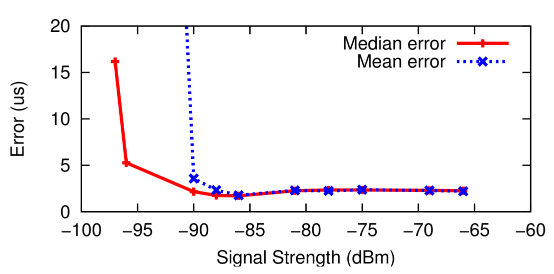
Though the above output gives the user hints on what could be the cause for performance problems, it does not tell the user the extent to which a problem degrades the throughput. As a first step towards this goal, we have built a performance monitoring tool that prescribes a time break down for various 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 detected, as we will show in Section 5.3.

## 4 Implementation

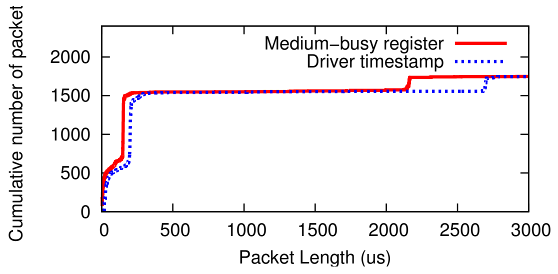
WiMed's monitoring component is built on the *MadWifi* driver for the Atheros chipset. We use an Atheros PCMCIA card based on the AR5212 chipset as the secondary NIC (in *monitor* mode). We use *tcpdump* to get the packet trace along with the entire payload. We have modified the driver to record the frame length captured (from the frame descriptor in the driver) through the *radiotap* header and correspondingly modified *tcpdump* to be able to read it. The `AR5K_PROFCNT_RXCLR` register is read every 2 $\mu$s to track the time periods when the medium is busy. In this section, we look at challenges in implementing fine-grained monitoring while keeping limitations of buffering and the CPU load on the diagnosis node in mind. We then briefly look at some of the implementation details of the diagnosis procedures which use the monitoring data to detect performance problems.

The medium-busy register is a counter that maintains the number of cycles (of the NIC clock) the medium was sensed as busy. In coarse-grained monitoring, this register is read whenever a packet is received along with the timestamp. In the case of fine-grained monitoring, this register is read every 2 microseconds in a tight loop. If we were to run this continuously, the diagnostic node would become unresponsive. As a trade-off between measurement fidelity and machine load, we run this loop for a time period just less than one *jiffy* (which is one tick of the system timer interrupt), and then schedule the same function for the next jiffy and so on. Note that we cannot schedule this function at a granularity of less than a jiffy. Although we can perform this at a scale of multiple jiffies, there is also a trade-off between measurement fidelity and the amount of data we write to the kernel log buffer. Here, we do not record the timestamps each time we read the register, as it takes a couple of microseconds to read and record the timestamp. The amount of data that is being written to the kernel log is reduced by doing this. So, we record timestamps at the start and end of each loop (jiffy) and then interpolate for each observation. Moreover, we record just the start and end of a region where we sense the medium busy instead of recording the entire trace of the register at each iteration. The value of jiffy in the platform used for our experiments is 4 ms. Since the above functions are implemented in the driver, this portion of WiMed runs in kernel space.

The trace merging and diagnosis modules are written in *perl* and run in user space. The procedures used in diagnosis are outlined in Section 3. We now explain the implementation details of non-802.11 source detection and interference detection in this section. We use thresholding for detecting the presence of non-WiFi signal in the medium. Since any non-decodable WiFi signal would be treated as a non-WiFi signal by WiMed, we calibrate the fraction of non-WiFi to WiFi signal as seen from the location where the wireless node is situated. The diagnosis module's confidence in the presence of non-WiFi signal is based on how much the current value of the parameter is above the calibrated value. With respect to interference detection, we use fixed values for expected packet error rates for packets received at different rates and at different SNRs (signal-to-noise ratio). Since it is hard to collect the entire matrix of values, we empirically calculated a few points in this matrix and interpolated (or extrapolated) as the case may be. Also, we only calculate packet error rates for 1500-byte packets and use it as a conservative estimate. However, the accuracy of the diagnosis modules is not entirely dependent on the exact values of error rates. This is because we use a continuous range of confidence values instead of binary thresholds and



(a) 802.11



(b) Bluetooth

Figure 6: Accuracy of register tracking for 802.11 and Bluetooth

further, only if the confidence values of multiple features are high, will the resulting confidence be high. We use a moving window of hundreds of packets to maintain various parameters such as packet error rates and medium busy time.

## 5 Evaluation

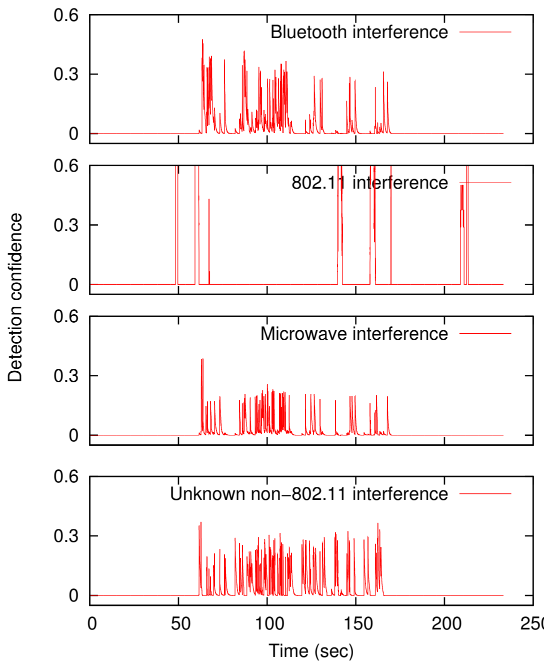
In this section, we first evaluate the accuracy of the medium-busy register in detecting the presence of 802.11 and non-802.11 sources. Then we show results for interference detection in the presence of different kinds of interferers and also, contention detection.

### 5.1 Accuracy of sensing

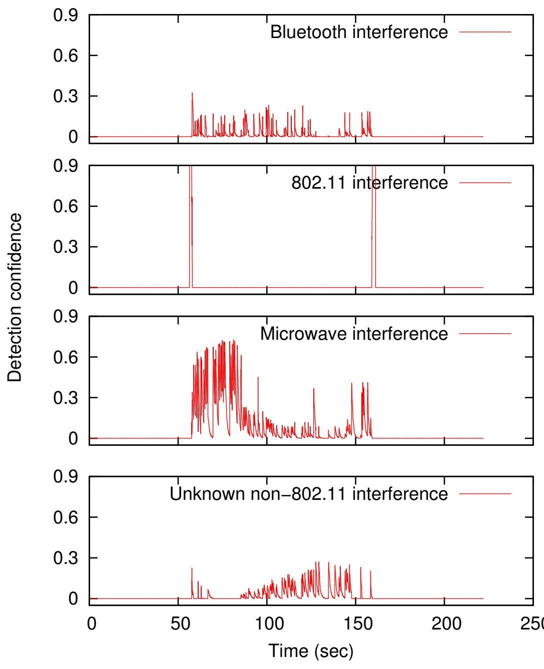
As described in Section 4, we use the medium busy register to track if there is any signal present in the environment. We compare this with the time occupied by demodulated 802.11 packets to infer the nature of the signal. First, we show that tracking this register gives us an idea of when the medium is being utilized and whether there is presence of 802.11 or non-802.11 signal. The following experiments were performed on a wireless emulator testbed [13].

#### 5.1.1 802.11

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. We see (Figure 6 (a)) 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 sampled this register periodically every 20 $\mu$s 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 $\mu$s (0.5 %).



(a) Bluetooth interference (60 to 160 sec)



(b) Microwave interference (60 to 160 sec)

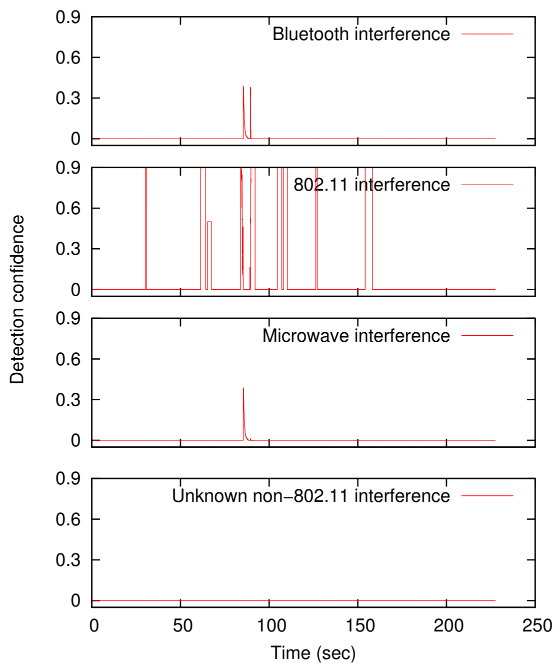
Figure 7: Bluetooth and Microwave interference as detected by WiMed

### 5.1.2 Bluetooth

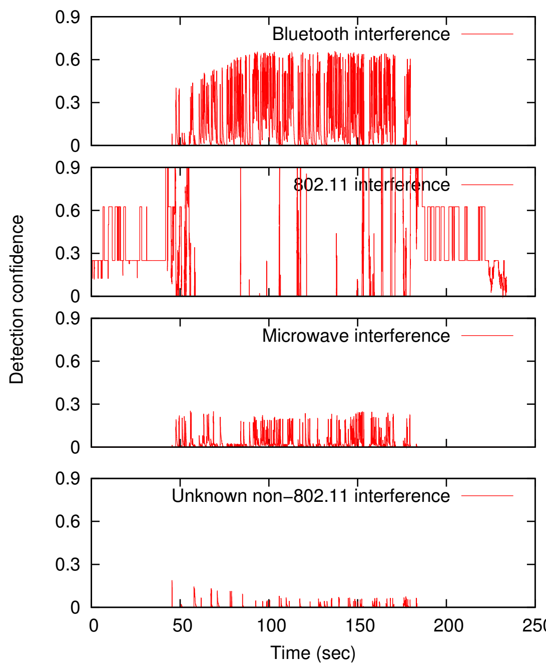
We also conducted a similar experiment for Bluetooth. We used `l2ping` utility to send 800 pings of length 2600 $\mu$s (and an equal number of MAC-level acknowledgments). Figure 6 (b) shows the cumulative distribution of the packet lengths as reported by an 802.11 monitor node using the latter tracking mechanism. 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 timestamp of a frame found using energy detection (based on the medium-busy register values). We see clear *steps* at 150 $\mu$s and 2150 $\mu$s corresponding to the L2CAP pings and the MAC-level acknowledgments. The number of ping-sized packets (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 slightly shorter than the expected length (by about 20 % of actual length) 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.

## 5.2 Accuracy of detection

We now evaluate some of the diagnosis components of WiMed in the real-world and show their accuracy. For this purpose, we used a laptop with two 802.11 NICs (Intel and Atheros chipsets),



(a) 802.11 hidden terminal interference (60 to 160 sec)



(b) Cordless phone interference (60 to 160 sec)

Figure 8: 802.11 and cordless phone interference as detected by WiMed

one in *managed* mode and the other in *monitor* mode. WiMed has been installed on this laptop and uses the Atheros-based NIC. We set up experiments in which we introduced one known problem at a time and studied how the WiMed prototype fairs in each of the scenarios. In every experiment, the NIC in managed mode pinged a machine in the LAN at the rate of 10 pings a second using 1464 byte frames.

### 5.2.1 Bluetooth Interference

Figure 7 (a) shows how the confidence values of the interference detectors change over time in the presence of Bluetooth interference (between 60 sec and 160 sec). For inducing Bluetooth interference, we use the Bluetooth radio on the laptop to send packets (using *l2ping* utility) to a mobile phone. 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, we see that it is significantly lower than that of the Bluetooth detector. However, 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 values is exponentially averaged, the decay factor is set fairly high to ensure that the detection confidence does not become stale and to show when exactly performance problems were faced by the system.

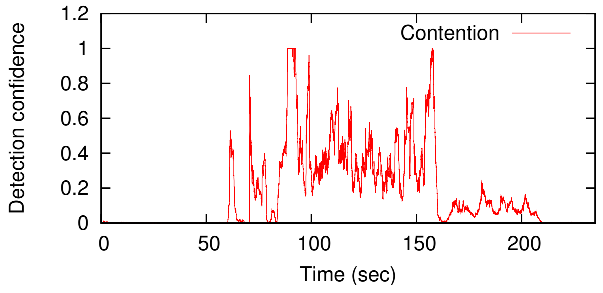


Figure 9: Contention as detected by WiMed using passive fine-grained analysis (60 sec to 160 sec) for the same trace as 802.11 interference

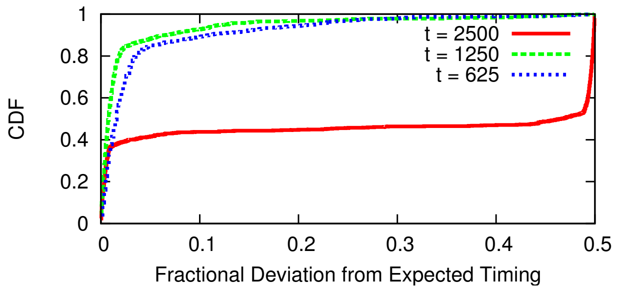


Figure 10: Timing behavior detection (TDMA) of a cordless phone based on BETA

### 5.2.2 Microwave Interference

We performed a similar experiment with a microwave interferer. We used a residential microwave oven placed at a distance of about 6 feet from the laptop. A microwave oven, because of its wider frequency range of use and higher power, causes interference to 802.11 packets irrespective of the channel used. Figure 7 (b) shows the detection confidence of the three interference detectors. Unlike the previous experiment, there is a clear difference in the output of the non-802.11 interference detectors. While the microwave detector has a very high confidence, the Bluetooth detector has a very low confidence. The 802.11 detector has zero confidence in most of the region where there was interference except for the start and end of the interference region. This phenomenon is because the non-802.11 presence detected at the beginning and end of the region was less than the detection threshold.

### 5.2.3 802.11 Interference

A hidden terminal in 802.11 is one where there is an 802.11 sender that is hidden from another 802.11 sender, but the receiver of at least one of them can hear both of them. We induced this scenario by modifying the driver using the patch provided by [4] to disable clear channel assessment, backoff behavior etc., so that the 802.11 node behaved 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 to increase the probability of collision. From Figure 8 (a), we see that while the

802.11 detector reported high confidence during the interference period (60 sec to 160 sec), the other two detectors barely reported anything at all.

### 5.2.4 Contention

Now, we used the same trace as in the above experiment (802.11 interference) to plot the confidence level of the contention detector. Since the hidden laptop and the diagnosed laptop were within carrier sense range, the latter still contended for using the medium with the former (though the reverse is not true). Figure 9 shows the confidence of the contention detector. During the active period of the hidden node, we see that the contention detector reported high confidence. This along with the above observation, could be used to detect the presence of a hidden 802.11 terminal with a higher confidence.

### 5.2.5 Unknown Interference

In order to test WiMed's diagnosis capability, we introduced an unknown interferer in the form of a 2.4 GHz DSS cordless phone and conducted several experiments. Figure 8 (b) shows the output of one such experiment. We see that though it is detected as non-802.11 interference, it is classified as Bluetooth interference. However, there were no Bluetooth devices active close to the 802.11 receiver. In order to understand this misclassification, we fixed an initial interference start time as a reference point and plotted the absolute deviation of the start times of the remaining bit error peaks (detected as Bluetooth) from the expected timing for various values of TDMA slot times. Assuming that the detection of start times of interfering transmissions is accurate, a low deviation for a give slot time $T_0$ would mean that the TDMA slot time of the interferer is a multiple of $T_0$. Figure 10 shows the cumulative distribution of the fractional deviation from expected timing behavior. We can see clearly that the TDMA behavior of the cordless phone matches with a slot time of 1250 $\mu$s (while a slot time of 625 $\mu$s also works, a slot time of 2500 $\mu$s does not). This is very close to Bluetooth's timing behavior, which explains our results.

The above experiment shows that we can develop automated mechanisms for detecting unknown types of interferers if they have distinct timing properties. Having said this, there could also be scenarios (like above) where multiple technologies exhibit similar kind of timing behavior, which could make it harder to pin down on the exact cause. However, from our experiments, we see that interference is most often correctly classified as 802.11/non-802.11 interference. This itself goes a long way in aiding the user to diagnose such problems.

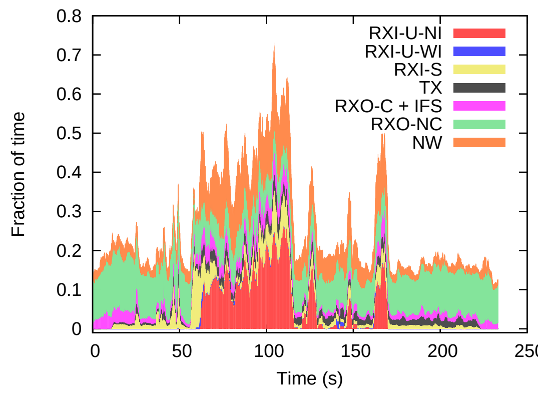
## 5.3 Performance monitoring

In the previous section, we looked at how well the detection schemes work under real-world scenarios. Detection provides an idea of what the cause of the performance problem could be, but it does not tell us how much each of these causes impacts the performance. We have incorporated a performance monitoring tool in WiMed that provides a time break-up of how the medium is utilized. It gives the user an idea of how the spectrum is utilized and can provide insight in how impact the identified problems may have on performance. This is a first step towards a mechanism

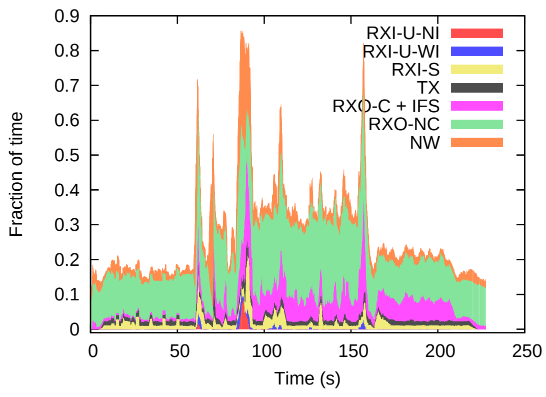
| | |
|---|---|
| RXI-U-NI | Packets received by the node (RX) and destined to the node (I) that are unsuccessful/have errors (U) and the cause is identified as non-Wifi interference (NI) that is not Bluetooth |
| RXI-U-WI | Similarly for Wifi-interference |
| RXI-S | Packets successfully received by 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 the graphs

that can automatically quantify the impact of problems on performance. Table 1 gives the notation used in Figures 11 (a) and (b).



(a) Bluetooth experiment



(b) 802.11 experiment

Figure 11: Time utilization map of the wireless environment in the presence of Bluetooth and 802.11 interference

As we see in Figure 11 (a), the graph corresponding to `RXI-U-NWI` and `NW` occupy about 20% of the medium each during the interference period (corresponding to the Bluetooth experiment in Section 5.2, Figure 7 (a)). This shows that the interference was strong enough to affect the throughput considerably. Though the 802.11 client was sending (and hence, receiving) pings at a constant frequency, there are regions in the graph corresponding to RXI-S where there are spikes. These are primarily due to the fact that the secondary NIC was able to decode a bunch of successive retransmitted packets, which were not decoded by the primary NIC. On the other hand, Figure 11 (b) (802.11 experiment in Section 5.2, Figure 8 (a)) shows that the interference by the hidden terminal (`RXI-U-WI`) has not affected the throughput as much by interference compared to 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.

## 6 Related Work

Client Conduit [1] was one of the first proposed architectures for diagnosis in 802.11 networks. WiFiProfiler [7] addresses the problem of diagnosis in 802.11 hot-spot scenario and like [1], largely looks at configuration/connectivity-related issues such as AP detection, association, DHCP and DNS problems.

Jigsaw [9, 8] and Wit [17] monitor 802.11 packets from different vantage points. By merging traces from these locations, they try to find the cause of the problem (throughput/latency). DAIR [6] monitors enterprise 802.11 networks using a dense deployment of monitor nodes. It attempts to address performance issues in 802.11 networks using location information. Jigsaw, Wit and DAIR represent the current state-of-the-art in wireless diagnosis. However, these systems require a lot of infrastructure to be functional.

MOJO [19] attempts to tackle the problem of fine-grained diagnosis with the help of a similar distributed setup. The range of problems addressed by MOJO is limited and like the above systems requires fairly complex infrastructure (i.e. requires wireless peers to combine observations from these peers). WiMed is probably closest to MOJO in spirit, but diagnoses problems in a top-down fashion and attempts to do delve deeper into the physical and MAC layer to get a better picture of the wireless environment. Moreover, WiMed tries to diagnose based on local measurements at the node and hence pushes the envelope on local diagnosis. Wherever applicable, this could be integrated with a distributed mechanism to perform more sophisticated global diagnosis. RFDump [15] uses software-defined radio to get a packet-level trace of all wireless technologies. This makes interference diagnosis easier and provides a local diagnosis mechanism, but requires relatively expensive SDR hardware as against cheap commodity NICs. Finally, there has been research on use of active approaches such as *bandwidth tests* [18] and *micro-probing* [3] to build conflict graphs for understanding 802.11 interference. These approaches try to solve 802.11-specific problems and deal with only a subset of performance problems in wireless systems.

Spectrum MRI [5] is a recent project that attempts to perform cross-protocol interference diagnosis. However, their preliminary efforts to diagnose these problems are based on coarse parameters like retransmission rate and link occupancy. We believe that these high-level parameters are unlikely to give insights into the technologies that are interferring with each other.

## 7 Conclusion

In this paper, we presented an architecture for diagnosing 802.11 problems without the use of a dedicated infrastructure. We built a prototype using commodity 802.11 NICs to identify the cause of wireless problems such as interference and contention. 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 wireless deployments such as the home. Unlike existing systems which largely focus on WiFi-only networks, WiMed extends its capability to diagnosing cross-protocol interference.

## References

[1] Atul Adya, Paramvir Bahl, Ranveer Chandra, and Lili Qiu. Architecture and techniques for diagnosing faults in ieee 802.11 infrastructure networks. In *MobiCom '04*, Philadelphia, PA, USA, 2004.

[2] Bhavish Aggarwal, Ranjita Bhagwan, Tathagata Das, Siddharth Eswaran, Venkata N. Padmanabhan, and Geoffrey M. Voelker. Netprints: diagnosing home network misconfigurations using shared knowledge. In *NSDI'09*, Boston, MA, 2009.

[3] Nabeel Ahmed, Usman Ismail, Srinivasan Keshav, and Konstantina Papagiannaki. Online estimation of rf interference. In *CoNEXT '08*, Madrid, Spain, 2008.

[4] Eric Anderson, Gary Yee, Caleb Phillips, Douglas Sicker, and Dirk Grunwald. Commodity ar52xx-based wireless adapters as a research platform. `http://systems.cs.colorado.edu/wiki/CARP`, April 2008.

[5] Akash Baid, Suhas Mathur, Ivan Seskar, Tripti Singh, Shweta Jain, Dipankar Raychaudhuri, Sanjoy Paul, and Amitabha Das. Spectrum mri: Towards automated diagnosis of multi-radio interference in the unlicensed band. In *IEEE Wireless Communications and Networking Conference*, 2011.

[6] Ranveer Chandra, Jitendra Padhye, Alec Wolman, and Brian Zill. A location-based management system for enterprise wireless lans. In *Proc. NSDI'07*, Cambridge, MA, November 2007.

[7] Ranveer Chandra, Venkata N. Padmanabhan, and Ming Zhang. Wifiprofiler: cooperative diagnosis in wireless lans. In *MobiSys '06*, Uppsala, Sweden, 2006.

[8] Yu-Chung Cheng, Mikhail Afanasyev, Patrick Verkaik, Péter Benkö, Jennifer Chiang, Alex C. Snoeren, Stefan Savage, and Geoffrey M. Voelker. Automating cross-layer diagnosis of enterprise wireless networks. In *SIGCOMM '07*, Kyoto, Japan, 2007.

[9] Yu-Chung Cheng, John Bellardo, Péter Benkö, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. Jigsaw: solving the puzzle of enterprise 802.11 analysis. In *SIGCOMM '06*, Pisa, Italy, 2006.

[10] Martin Heusse Franck, Franck Rousseau, Gilles Berger-sabbatel, and Andrzej Duda. Performance anomaly of 802.11b. In *IEEE Infocom '03*, San Francisco, CA, 2003.

[11] Ramakrishna Gummadi, David Wetherall, Ben Greenstein, and Srinivasan Seshan. Understanding and mitigating the impact of rf interference on 802.11 networks. In *SIGCOMM '07*, Kyoto, Japan, 2007.

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

[13] Glenn Judd and Peter 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] Kaushik Lakshminarayanan, Samir Sapra, Srinivasan Seshan, and Peter Steenkiste. Rfdump: an architecture for monitoring the wireless ether. In *CoNEXT '09*, Rome, Italy, 2009.

[16] Kaushik Lakshminarayanan, Srinivasan Seshan, and Peter Steenkiste. Understanding 802.11 performance in heterogeneous environments. In *HomeNets '11*, Toroto, Canada, 2009.

[17] Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Analyzing the mac-level behavior of wireless networks in the wild. In *SIGCOMM '06*, Pisa, Italy, 2006.

[18] Jitendra Padhye, Sharad Agarwal, Venkata N. Padmanabhan, Lili Qiu, Ananth Rao, and Brian Zill. Estimation of link interference in static multi-hop wireless networks. In *IMC '05*, Berkeley, CA, 2005.

[19] Anmol Sheth, Christian Doerr, Dirk Grunwald, Richard Han, and Douglas Sicker. Mojo: a distributed physical layer anomaly detection system for 802.11 wlans. In *MobiSys '06*, Uppsala, Sweden, 2006.