The Shaman Automatic 802.11 Wireless Diagnosis System

Yu-Chung Cheng, Mikhail Afanasyev, Patrick Verkaik, Peter Benko, Jennifer Chiang,
Alex C. Snoeren, Stefan Savage, and Geoffrey M. Voelker

UC San Diego Tech Report CS2010-0959

Abstract

In this paper, we describe the design and implementation of an automated 802.11 wireless network diagnostic system called Shaman. Since the end-to-end performance of user traffic is some combination of factors across all network layers, Shaman incorporates comprehensive, cross-layer models of 802.11 network behavior and performance. These models include broadband interference at the physical layer, per-packet link layer media access delays and losses, network layer device mobility and association management, and transport layer congestion and flow control. No one anomaly, failure or interaction is singularly responsible for all network problems, and that a holistic analysis is necessary to cover the range of problems experienced in real networks.

When users experience unsatisfactory performance at a particular time, they can query Shaman for a diagnosis. Shaman will then profile a user's traffic at that time, determine the network events that shape the performance profile, infer the causal sources of those events, and report the results to the user. We demonstrate the use of Shaman on an enterprise wireless network deployed in a university campus building, and illustrate the underlying analysis Shaman performs on real network trouble reports submitted by users of the enterprise network.

1 Introduction

Internet access based on 802.11 wireless networking has become ubiquitous both in its use and its failures. Any reader of this paper is likely to have experienced both the pleasures of untethered high-bandwidth networking and the frustration posed by unexpectedly degraded service, unknown problems in connectivity and sudden disconnections. Moreover, such problems are frequently both transient and selective. Indeed, even more frustrating than having to ask, "Is the wireless working?", is the refrain, "Works fine for me." Such issues are only likely to grow in importance with the adoption of 802.11-based VOIP phones (a market projected to reach $70B by 2012 [12]) and 802.11-based telemetry in medical devices (increasingly ubiquitous in applications ranging from vital signs monitors to infusion pumps).

However, today's IT staff are poorly suited to solving such problems. In addition to the inherent complexity of diagnosing transient conditions, the complexity of the wireless environment places two unique burdens on the diagnostician.

First, the radio frequency environment is itself complex. Unlike point-to-point wired networks, an 802.11 transmission must contend with signal attenuation, interference (from both narrowband and broadband sources) and contention from competing 802.11 devices (who themselves may be mutually unaware of each other). As well, the 802.11 Media Access Control (MAC) protocol adds significant dynamism to the transmission schedule including link-layer congestion control, ARQ-based error control, dynamic rate selection and variable transmission power. Even further complicating this affair, individual vendors demonstrate significant heterogeneity in their implementation of the 802.11 "standard" and many of these impacts are not directly observable, but must be inferred.

Second, 802.11-based devices are generally designed to be mobile. Consequently, the 802.11 standard provides mechanisms for discovering, associating with and authenticating to *access points* – again with varying vendor algorithms on how best to do this. However, to manage this mobility at the network layer, most enterprise 802.11 networks employ some additional non-standard network access control, involving user authentication, dynamic network address management (typically via a combination of DHCP and ARP) and VLAN-based address mobility. All of these disparate parts are fragile – if one fails it may be sufficient to terminate a user's session and if one is overloaded it may indirectly cause transient delays across the network. Finally, these symptoms are further masked by the inherent variability of the Internet itself, which introduces its own packet losses and delays independent from those of any 802.11 access network. Consequently, it is rare that a network administrator can explain why the wireless network was slow and it is common that persistent performance problems go unnoticed and unresolved unless they are so severe to prevent use.

It is our contention that this state of affairs is unlikely to change. Network administrators simply can't know enough – both in the quantitative sense of examining large amounts of diagnostic data and in the qualitative sense of being an expert in complex interactions between a wide range of network protocol layers – to solve such problems. In this paper, we introduce a system, called Shaman, for automating 802.11 wireless diagnoses [1]. Based on the public Jigsaw engine of

[1] The English word Shaman is derived through Russian, from the Tungus word saman meaning "one who knows"

Cheng et al. [9], Shaman processes distributed packet traces from both wireless and wired monitors to construct a comprehensive viewpoint of both wireless activity and dependent network services (e.g., DHCP). Combining this viewpoint with a causal model of protocol interactions, Shaman identifies problem causes by process of elimination – ranging from high-level issues (inability to transmit due to DHCP lease timeout combined with DHCP server failure) to low-level interactions (temporary queuing at the access point caused by broadband interference from a microwave oven).

Shaman is designed to operate in two modes: reactive and proactive. In the reactive mode, individual trouble tickets identify the MAC address and approximate time of a service problem and then system attempts to identify the cause post-mortem. By contrast, the proactive mode is designed to alert automatically on significant outages or degradations and then apply its analysis to these. To evaluate our approach we have subsumed all "help desk" functions for a modest production wireless network (several hundred users) and have used our system to diagnose submitted trouble tickets. As well, we have constructed artificial problems and verified our ability to identify their underlying causes.

The remainder of this paper is structured as follows: in Section 2 we discuss the prior work that we build upon and our relation to other diagnostic approaches. We then outline the architecture of the Shaman system in Section 3 followed by detailed descriptions of each of our analysis modules in Section 4. We evaluate Shaman as applied to problems in our production wireless network in Section 5 followed by a summary of findings.

2 Related Work

Various commercial products [1, 2, 3] and research systems have been developed to monitor and diagnose 802.11 wireless networks. Research systems in particular have evolved from developing infrastructure for performing distributed wireless monitoring and demonstrating the analysis capabilities of such platforms [9, 10, 19, 15] to systems for diagnosing problems in wireless networks [5, 6, 7, 18, 14, 16].

In particular, Jigsaw [9] uses monitoring nodes distributed throughout a university campus building to capture every wireless event in the building across location, channels, and time. Jigsaw combines and synchronizes traces from all radios into a single, unified trace, but in its original form provides little analysis for diagnosing problems. We use the Jigsaw software in our environment, and have extended it with our models and analyses for diagnosis.

Shaman shares some goals with WiFiProfiler [5], which also helps users troubleshoot wireless connectivity problems. The two systems take different approaches, however. WiFiProfiler relies upon peer diagnosis among clients, while Shaman relies upon third-party monitoring and inference. WiFiProfiler installs custom software on the client to collect detailed network stack statistics, such as beacon losses and queue length, as well as OS and driver details. It then exchanges this information with peers to diagnose connectivity problems. The client can then determine if it has an association problem, DHCP problem, or TCP problem. Due to local knowledge, however, the diagnosis is restricted to determining relatively high-level causes. For example, the client TCP diagnosis can indicate high TCP loss rates, but not the cause of the losses. The advantage of WiFiProfiler is zero infrastructure requirement and is best suited for ad-hoc first-step diagnosis. On the other hand, the diagnosis is limited at high level and users may raise security/privacy concerns to install custom software to exchange detailed OS information with other users. Overall, Shaman and WiFiProfiler are complementary to each other because each poses extra knowledge about the wireless network that cannot be perfectly inferred.

DAIR [6, 7] helps system administrators diagnose WiFi problems. DAIR and Shaman use similar approaches, distributed wireless monitors, for monitoring detailed wireless events in an enterprise network — DAIR uses wireless USB dongles attached to standard desktop machines, while Shaman uses a monitoring infrastructure similar to Jigsaw [9]. DAIR applications install trackers on the desktop machines to trace information of interest and store it in a central database; applications (inference engines) then query this database to perform analyses. DAIR uses a sophisticated location algorithm to estimate client locations accurately, and narrow diagnoses relative to location; in contrast, Shaman uses a merged trace of global activity across the entire network. Unlike Shaman, though, DAIR does not perform detailed PHY or link layer loss or delay analyses due to lack of low-level traces. Our goals are similar in that we develop analyses to aid network management, but our approach is to base analysis on a global understanding of network behavior across all protocol layers.

3 Shaman system architecture

This section describes the architecture of the Shaman system. Shaman consists of four main components. Figure 1 illustrates the relationship of these components and how data flows among them. A combination of wireless and wired network monitors collect network traffic in real-time as input into the system. Then a series of synchronization and preprocessing steps synchronize and merge the traces, normalize frames, track station association status, and reconstruct frame exchanges from individual transmission attempts. Next the processed trace data feeds into a collection of modules that perform detailed analyses of network performance and behavior across all layers of the network stack. Finally, two diagnostic tools use the output of the analysis modules. A user diagnostic tool answers queries on demand from users about wireless network problems, and a real-time network diagnostic tool alerts network administrators to pervasive wireless problems.

We describe the first two components of the architecture in detail in the remainder of this section. Then in Section 4 we describe the network models and analyses incorporated into each of the analyzers, and in Section 5 we describe the operation of the user and network diagnostic tools.

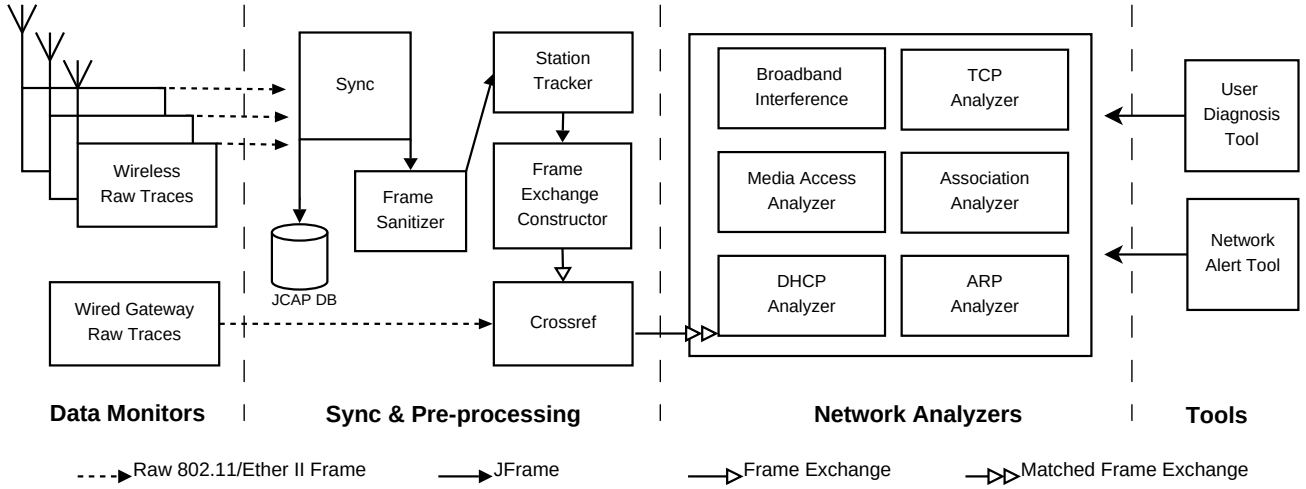

Figure 1: The Shaman system architecture.

We have deployed a prototype Shaman system in our university department building, and we are using it to diagnose problems in our production 802.11 access network. The production network consists of 40 APs serving over 1,000 unique client MAC addresses. Shaman uses a hardware and software monitoring platform similar to that used by the Jigsaw project [9]; this platform consists of over 190 wireless monitors that provide coverage of our building across location, channels, and time.

We draw upon our experiences using the system prototype throughout the rest of the paper to motivate its design and illustrate its use.

3.1 Data monitors

Shaman monitors the wireless network from two different vantage points, the wireless monitors and the wired distribution network. Each wireless monitor passively monitors the channel and records all wireless events including frames with CRC and PHY errors. The monitors capture the first 120 bytes of most received frames; for DHCP frames, we capture 400 bytes to include the necessary DHCP headers. The monitor compresses and streams the traces back to a central storage server.

The wireless monitor hardware has limited processing capability and memory. Extreme packet bursts, e.g., physical errors bursts caused by microwave oven, can cause receiver livelock in the wireless monitor and depletes its memory and triggers reboots. This would leave 2 minutes of trace holes in our raw wireless traces. The wireless monitor disables wireless radios' interrupts for one second when monitor memory goes low.

Shaman also records wireless network traffic as it traverses the wired distribution network. A SPAN port on the router for the building distribution network forwards all packets to a wired tracing machine, which also has a wireless monitor. Tracing packets on the wired distribution network is necessary for inferring detailed media access delays (Section 4.3); in particular, we need to infer when wired packets arrive at the APs as the basis for estimating queuing within the AP. We attach an additional wireless monitor directly to the wired monitor to synchronize across the the wired and wireless time domains.

3.2 Synchronization and preprocessing

Shaman merges and time synchronizes all of the traces from the wireless monitors into a single global unified trace. Shaman implements synchronization based on Jigsaw's algorithms [9]. It merges frame transmissions observed by multiple monitors into a single *jframe*, and timestamps each jframe according to a global virtual clock. For example, if three monitors receive a unicast DATA frame transmitted by an AP to a client, Shaman will identify these frames as equivalent and create a jframe representing that single transmission.

We have augmented Jigsaw's synchronization with multiple enhancements to better support its use in a diagnostic system. First, we track the offset between the synchronized virtual clock and the real-time clock of the wired trace monitor, and timestamp jframes according to the real-time time. Since our wired trace monitor also has a wireless radio that collects wireless traces, we slave the synchronized virtual clock to the real-time clock by putting real-time clock timestamps in that particular wireless radio's traces. Using this timestamp serves two purposes. One, it synchronizes the time domains of the wireless and wired traces. Two, it enables the diagnostic system to easily correlate times specified in user queries with timestamps in the jframe trace. Second, we have tuned Shaman to synchronize and compress the traces in real time with a relatively short delay. Every minute, the synchronization process takes 3-15 second on a Pentium4 2.4GHz machine which enables Shaman to perform real-time analysis.

The output of synchronization is a continuous stream of jframes in a custom *jcap* format. A jcap trace has the format of a gzipped pcap file that combines a jcap header with every jframe. The jcap header includes the transmission rate,

preamble length, etc., in a 24-byte record for each jframe. A record in the jcap trace is 170 bytes on average, resulting in only a moderate trace data rate even for large monitored networks. The enterprise network we monitor generates 100 to 6000 events per second, resulting in a compressed jcap stream rate of 5 GB/day. We simultaneously record the jcap trace to disk as input for future offline analysis, as well as feed it into the frame sanitizer for continued online processing.

The frame sanitizer examines jframes for anomalous or inconsistent fields, and corrects or drops them to ensure that all frames are valid before further processing. Inconsistencies are typically the result of buggy 802.11 firmware implementations. For example, we regularly see frames with anomalous durations (e.g., NAV fields with 0xFFFF) and packets with inconsistent types and lengths (e.g., wireless security Webcams in our building transmit ACKs with impossibly large packet lengths). We mark these frames and adjust frames whose fields we can correct (e.g., by calculating the correct value for the NAV field based on the frame length and rate). A later analysis module can then process these frames without concern.

The station module tracks state related to a station's association with the network as a function of time. This state includes the preamble mode (short or long), slot time (short or long), power save mode, the use of 802.11g protection mode and RTS/CTS, etc. The station module maintains this state directly, by tracking the parameters advertised in AP beacons and client scan probe frames, and indirectly, based on the timing of successive frame transmissions (e.g., sufficiently fast ACK responses preclude long preambles).

The station module provides reconstructs individual link-layer conversations using a *frame exchange constructor*. The constructor starts by identifying all *transmission attempts*. A transmission attempt usually consists of control frames, a data frame, and one ACK from the receiver (e.g., RTS-CTS-DATA-ACK is a common frame exchange pattern for 802.11g clients). The constructor then groups transmission attempts into complete *frame exchanges*. Since 802.11 implements ARQ for unicast frames, a frame exchange may involve multiple distinct transmission attempts. Normally it is sufficient to simply group nearby transmission attempts that share the same frame sequence number. But since the wireless monitors can fail to capture some frames, so the frame exchange constructor performs contextual inference to compensate for any omissions [9].

Typically, a frame exchange represents an packet delivered either to or from the wired distribution network. Thus, sucessful frame exchaages usually also have a counterpart in the wired trace. Relating these two vantage points serves two important functions. First, the timestamp associated with the wired packet indicates unambiguously when the frame entered or left the wireless network. These events are critical for inferring detailed 802.11 behaviors, which we discuss in detail in Section 4.3. Second, since the wired monitor does not drop frames it is critically useful in helping to identify any frames missing from the wireless trace. This is particularly essential for analyzing the TCP protocol, because such analyses can be quite fragile to missing packet data.[13].

The job of the cross referencer is to match the packets in the wired trace (Ethernet II frames) to the frame exchanges on the wireless side with identical data content. It handles both one-to-one cases, where one wired packet corresponds to one wireless frame exchange (e.g., a unicast DATA packet), as well as one-to-many cases, such as when a single broadcast ARP request on the wired network induces broadcast frames at all APs. The cross referencer adds a *matched frame exchange* structure into the trace, linking the wireless trace representation of the packet (the frame exchange structure) with the wired representation of the packet (the captured Ethernet II frame) and combining the two input traces into one output trace. The transitive extent of a matched frame exchange can be considerable. Consider, for example, a DHCP request from a client to an AP. The client transmits the DHCP request to the AP as a unicast DATA frame destined to the AP. Because the request is also a network broadcast, the AP bridges it by broadcasting it on the AP wired distribution network. Each AP, including the bridging AP, will then transmit a broadcast frame for the DHCP request. The matched frame exchange for this scenario will therefore encompass the initial unicast frame exchange, the wired ethernet frame for the bridged broadcast, and N frame exchanges for each of the broadcast frames from the N APs. Finally, the output of the cross reference module is the matched frame exchanges which will be used to driver various analysis modulers in the next sections.

4 Shaman network analyzers

The network analyzers are where the magic happens in Shaman, and in this section we describe each of them in detail. Each analyzer models the operation of a specific protocol (e.g., TCP, DHCP, ARP, etc.) or network layer (e.g., link-layer media access or physical-layer broadband interference). To provide insight into their operation, we also illustrate the use of the analyzers on real problems reported by users of our enterprise network.

Our previous work develops the analysis techniques and models for inferring sources of transfer delay due to media access and mobility [8]. This work is under submission, and we have provided a technical report version of the submitted paper as part of the SOSP reviewing process. With Shaman, we implement those techniques and models as software modules within a diagnostic system. These modules analyze the traffic an entire enterprise network both in real-time and on demand; we also further extend our previous work with new techniques and models. Where appropriate when describing the various analyzers, we first summarize and reference the analysis techniques developed in our previous work and then detail their use as software analysis modules as part of a diagnostic system.

4.1 Mobility

Stations need to complete a number of intermediate steps in order to obtain IP connectivity at a particular location. By "IP connectivity" we mean the ability to communicate with the Internet through a gateway. The steps are typically MAC-level association (including authentication) to obtain link-level connectivity, DHCP to obtain IP connectivity in the subnet, and ARP to allow forwarding by the gateway linking the station to the Internet. The time duration of this critical path directly affects the user experience on the network.

To diagnose user problems along this critical path, we implement three network analyzers that track exchanges in each of these protocols. In particular, we keep track of how much time each protocol spends on a critical path and look for failures, poor performance, and other indicators and anomalies (e.g., excessive time spent on the critical path, DHCP leases for private (unusable) addresses, etc.). The analyzers query the media access analyzer to identify cases where loss of particular wireless packets may provide the explanation for an anomaly. This information will be used by the diagnostic tools to explain user reports and detect network-wide events.

4.1.1 MAC-level association

The MAC-level association protocol consists of two main steps: choosing the access point and connecting to it. AP selection starts with the client station broadcasting Probe Request packets which specify the client's capabilities, and might also contain the SSID of the desired network. All APs which hear those requests will respond (via unicast) with Probe Response packets. Stations will wait before switching to the next channel. This procedure is repeated on all channels (which might take a while, especially on multi-band cards with 802.11a enabled), and a proprietary algorithm picks the best AP to connect to. If no AP is satisfactory, the scanning will start over. APs also send Beacons to announce their presence, but stations do not usually use them for the AP selection.

Once a station selects an AP, the station sends an Association Request to it. Since our network is open, this step rarely fails, and the AP immediately responds with an Association Response. The station then sends an Authentication Request. If no previous packets were lost, the AP sends an Authentication Response, and the station proceeds to the next step (DHCP).

The protocol is straightforward, but clients performing it still often have problems. The first problem is an unsuccessful scan. If the wireless loss rate is high, probe responses might not get delivered. Worse, if there is interference on a channel during the short window a station was scanning there, the access point might not be detected at all. These losses can make stations scan again, until a satisfactory access point is found. Or they can cause a station to pick a suboptimal access point, connect to it, and suffer high packet losses until scanning again.

The access point detection is challenging to automate since we do not know if the repeated scans observed by a client were caused by the user, by the AP selection algorithm, or by bad connections. The steps can also vary substantially depending on the target OS, network driver, and usage scenario. For example, some OS/driver combinations start sending probe requests immediately on OS boot. Thus, multiple scans with an empty SSID do not necessarily indicate a problem. OS delays are also important. For example, when the computer becomes active from standby or sleep mode, frequently the network card is initialized very early in the resume process and starts scanning immediately. However, it would not associate until resuming is complete. As a result, client experiences long periods of scanning with a non-empty SSID which are not caused by network anomalies.

The MAC-level association analyzer tracks the state of clients as they perform the association protocol, and measures the time spent on each step of the protocol. If a client's association progresses through the normal sequence of steps, we call it a successful association. Otherwise, we call it an unsuccessful association. The analyzer keeps track of the number and total duration of unsuccessful associations.

Unsuccessful associations are not uncommon, and we have observed a variety of behaviors. Sometimes, a station that have just associated decides that the current access point is unsatistactory, so just 40 seconds after the association was successful, the station disconnects and starts searching for a better access point. After searching for 5-10 seconds, it comes back to the same access point it left.

MAC-level associations can also affect established connections. Bad station selection algorithms degrade the connection by invoking re-association and blocking all connectivity for 10 to 30 seconds.

More unsusual cases happen. For instance, one station is working normally and has multiple active TCP connections. The AP suddenly sends a deauthentication frame with 'Timeout' in the reason field. According to the protocol specification, access point should send this packet when it have not heard from a station for a long time. The station is supposed to immedeately cease any high-level transmissions and reassocate. However, the station ignores the frame and continues sending data, thus violating the protocol. The AP also ignores it and forwards data to station. Then the normal association procedure is started, in parallel with the active TCP flows. The procedure fails, and all connections get reset.

4.1.2 DHCP

Clients use DHCP to obtain IP connectivity on the bridged wireless subnet. A station (DHCP client) that needs to obtain an IP address broadcasts a DISCOVER message on the subnet. Any DHCP servers that are reached by the message and are able to provide the client with an IP address respond by unicasting an OFFER message containing the IP address to the client's MAC address. After collecting OFFER messages, the client selects one and broadcasts a REQUEST message based on the selected OFFER message. The REQUEST message confirms to the selected server that its offer was accepted, and also indicates to the other servers that their of-

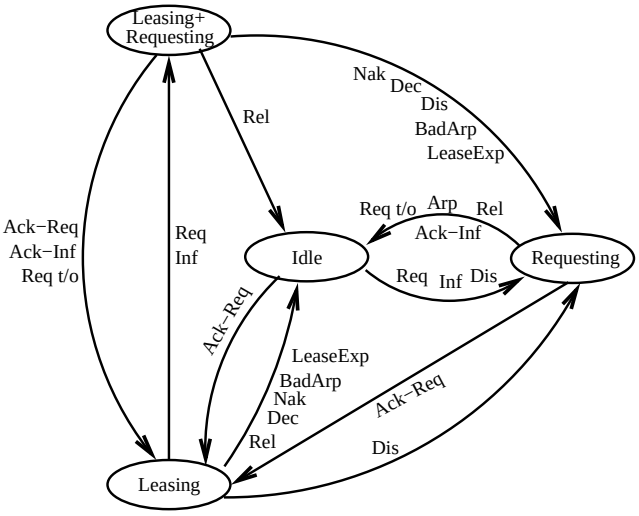

Figure 2: Finite state machine to track DHCP.

fers were declined. Finally, the selected DHCP server commits its offer, and unicasts an ACK message to the client containing a lease time. The ACK message grants the client use of the IP address until the lease time expires. (Alternatively, the server may send a NAK message to cancel its offer.) The client may renew the lease by unicasting or broadcasting a REQUEST message containing the IP address, upon which the server may return an ACK or NAK message. To accommodate client TCP/IP stacks that cannot receive unicast messages until an IP address has been configured, a client may request the server to broadcast its responses to the client.

On receiving an ACK message, the client may discover that the IP address is already in use by another host (e.g., using ARP probes). For example, this may occur if the other host is statically configured with an IP address. In this case the client sends the server a DECLINE message. In addition, the client may relinquish use of the address by sending a RELEASE message to the server, e.g., when the client is shut down.

DHCP is also used to provide a host with parameters such as the IP address of the gateway, DNS name servers, etc. DHCP provides a separate mechanism to allow clients that have been externally configured with an IP address to learn these remaining parameters. Such a client broadcasts an INFORM message, and receives parameters from a server through an ACK message.

The DHCP analyzer tracks DHCP message exchanges using a finite state machine. Given that a station is attempting to establish IP connectivity we can define expected state changes and flag other state changes as anomalous. Figure 2 shows the finite state machine. We track two properties of the station: whether it is making requests (sending DISCOVER, REQUEST or INFORM messages) or not, and whether it owns a lease or not. These two properties combine to make four different states, as shown in the figure.

When the station is in **Requesting** it does not have a lease and is on the critical path towards acquiring IP connectivity. The goal of DHCP is satisfied when the station has obtained a lease. We have observed that stations are quite aggressive in renewing their leases, and so we treat attempts to renew a lease before a current lease has expired as being off the critical path. (Indeed, we have observed that in our network such renewals are often not answered, yet this is harmless if the station still owns a lease.) Therefore **Leasing** as well **Leasing+Requesting** are not on the critical path. Finally, when the station is in state **Idle**, it may or may not be on the critical path. If the station is simply switched off, it is not on the critical path. If the user is attempting to communicate, it is on the critical path.

A station that needs IP connectivity progresses through states **Idle** (switched off) to **Requesting** to **Leasing** in that order, then alternates between states **Leasing** and **Leasing+Requesting**, and ultimately reverts to **Idle**. Based on this the analyzer flags as anomalous:

- Time spent in **Requesting** longer than two seconds. This situation indicates that the station is getting poor service.
- Transitions from **Leasing** or **Leasing+Requesting** to **Requesting** or **Idle**, as well as transitions from **Requesting** to **Idle**. However, we allow transitions to **Idle** periods longer than a minute, since this is expected for stations that have switched off.
- **Leasing** and **Leasing+Requesting** if the lease is for an IP address that is invalid in our network or for less than one minute. We have observed cases where stations send REQUEST messages for private network addresses that are granted by the DHCP server. However, these addresses are unusable for communication. Leases of twenty seconds also appear to be common in our network.
- A change of IP address. This is disruptive for ongoing transport-layer sessions. Also it may cause re-ARPing.

For each of the above anomalies, we attempt to provide further detail. For example, for each state change we store the event (e.g., message type) that causes the state change. In addition, in the case of excessive time spent in state **Requesting**, we break down the time spent and assign it to specific causes such as wireless packet loss, station-side and server-side delay, and lack of server response.

Since the wireless monitoring system cannot guarantee complete coverage of all transactions, we conservatively assume that an ACK has not been delivered to the station unless we see evidence that it has. Examples of such evidence are: wireless loss inference indicates a MAC acknowledgment was seen, a subsequent RELEASE or DECLINE, an ARP message sent from the IP address in the ACK, or a lack of DHCP activity in the station. In particular, we assume the ACK is not delivered if the client proceeds sending request-type messages (DISCOVER, REQUEST or INFORM) soon after.

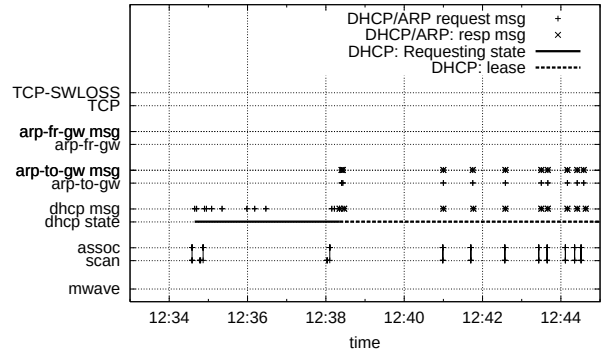

Figure 3: DHCP analyzer example.

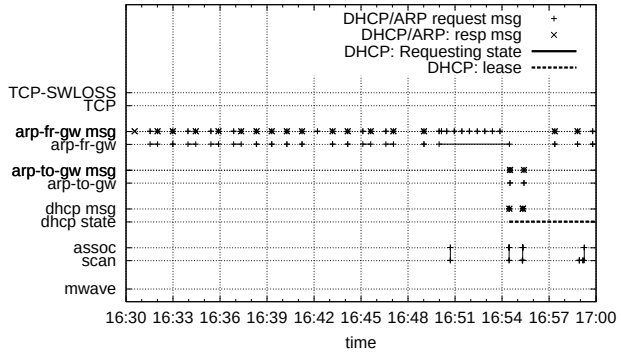

Figure 4: ARP analyzer example.

Figure 3 shows an example of DHCP activity for a user reporting frustratingly poor connectivity. The client begins without having a lease. Just after 12:34 it requests a lease, the analyzer enters the **Requesting** state (solid DHCP line). During this period the client sends a number of request messages that are not answered. Finally after nearly 4 minutes, it receives an ACK and obtains a lease (dashed DHCP line). The subsequent request messages for lease renewal cause the analyzer to alternate between states **Leasing** and **Leasing+Requesting** (not shown). The **Requesting** period is longer than two seconds and flagged as an anomaly. While the client owns a lease, the analyzer ignores any request timeouts.

4.1.3 ARP

The ARP protocol establishes the mapping of IP address to MAC address in a subnet. ARP allows a host that wishes to send a packet to a host in the subnet for which it has only the IP address to discover what MAC address to send the packet to. We are interested in ARP between a station and gateway: a station has the IP address of the gateway but must discover the gateway's MAC address before it can send packets to the gateway for forwarding to the Internet. Similarly, traffic from the Internet destined to the station's IP address needs to be forwarded by the gateway to the station, requiring the gateway to map the station's IP address to a MAC address.

In a typical ARP exchange, a host that needs to know the IP address of another host on the subnet broadcasts an ARP WHOHAS packet. The owner of the IP address unicasts a reply IS-AT to the requester giving its MAC address. The ARP cache in the requester stores the results. A host periodically refreshes its cache entries by repeating WHOHAS queries. Cache entries are also updated using the source information of received WHOHAS queries.

We describe the case when a station ARPs the gateway; the reverse case is analogous. We define an ARP attempt as the station starting to send WHOHAS packets until it receives an IS-AT, or until it gives up after not receiving an IS-AT. If the ARP analyzer sees successive unanswered WHOHAS messages, it groups these together in the same ARP attempt using a timeout of two minutes. The analyzer measures the time of successful ARP attempts and sets a threshold of two seconds to indicate that the station is getting poor service. Similar to DHCP, before processing an IS-AT we first look for evidence that the IS-AT was received by the station. If the station gives up ARPing after we observe an IS-AT we infer that the IS-AT is received.

ARP is a simpler protocol than DHCP, but, due to its simplicity, we have less information about the internal state of the client than with DHCP. For example, ARP attempts may be made to refresh cache entries, in which case the ARP attempt is not on a critical path. How do we know whether an ARP attempt is on the critical path? Initially we assume the attempt is on the critical path, and then rule out critical path based on the following heuristics. We use heuristics based on observing IP packets that make use of the address mapping that the ARP attempt is trying to establish. Our first heuristic is to check if the ARP attempt is immediately followed by such an IP packet (one second). If not, we rule out the attempt being on the critical path. Second, we observe IP packets sent during the ARP attempt. IP packets sent to the gateway by the station are evidence that the station is aware of the gateway's MAC address. If the station is in the midst of an ARP attempt when we see such evidence, we infer that the ARP attempt thus far was not on any critical path and ignore it. (The remainder of the ARP attempt may still be on a critical path.)

Another case, specific to ARP attempts by the gateway, is when a gateway is sending WHOHAS packets to the station and is not getting IS-AT responses. While this may indicate a problem with the station's networking stack, flagging these cases as errors leads to a large number of false positives: the station may simply be switched off. In fact we have observed that our gateway periodically probes dead stations using ARP. (Possibly in order to reclaim DHCP leases, or in an attempt to forward traffic for the station's IP address.) To distinguish between these two cases we use the following heuristic. We only treat a gateway's ARP attempt as critical if we observe evidence that the station is alive during the attempt. To test for liveness we check if the station sends any 802.11 DATA frames (which include the ARP IS-AT response).

We illustrate our techniques to suppress false positives using Figure 4. 'arptogw' represents ARP queries for the gate-

way sent by the client, and 'arpfrgw' represents ARP queries for the client from the gateway. (Note that the long request period is ended by a WHOHAS query *from* the client.) There are many queries from the gateway. However, most (including the long request period from around 16:50 to 16:54) are not followed by an IP packet and so are not considered critical. (During most of these, the analyzer observes data frames sent by the client, so the client is actually alive.) As a result the analyzer finds only 1.9 seconds of critical time spent in ARP queries from the gateway. In contrast, disabling our heuristics results in 446 seconds of critical time.

4.1.4 Summarizing mobility

The purpose of keeping track of critical time spent in each mobility protocol is to aid the user diagnostic tool in assigning blame on a particular mobility. However, protocols interact and time spent by one protocol may in fact be due to malfunctioning of another. In particular we shift to DHCP the blame for critical time spent on an ARP attempt by a station in following cases of DHCP mishaps: leased a non-UCSD address, caused station's IP address to change, leased for less than one minute. In each of these cases, ARP time spent is useless. Also if DHCP allows its lease to expire and later acquires another lease, the subsequent ARP attempt by the station is blamed on DHCP. We only shift blame across a small period of time: to shift blame of an ARP attempt to DHCP, the ARP attempt must start no later than five seconds after the DHCP anomaly. After that, we assume DHCP is not the reason for ARP time. After shifting blame, the result is for each mobility protocol the critical time that it is responsible for and a set of detected failures and anomalies.

4.2 Broadband interference

Broadband interference at the physical layer from non-802.11 devices can significantly impact 802.11 performance. In our network, the primary source of broadband interference is from microwave ovens. Microwaves generate interference in the 2.4 GHz band used by 802.11b/g that causes frames to be delayed or corrupted. The 802.11 MAC can detect and adapt to such interference, but fundamentally performance degrades during such periods to such an extent that the user experience noticeably suffers.

As a result, our broadband interference analyzer currently focuses on detecting intervals of microwave interference. The analyzer takes synchronized jframes, which include counts of physical error bursts, as input. It analyzes the trace to produce two kinds of output into an internal trace related to microwave interference. First, for each wireless monitor, it determines the time intervals during which microwaves are actively generating interference as observed by that monitor. Second, it estimates which MAC addresses are likely affected by the interference for each interval. When queried on demand about a particular client at a particular time, the analyzer uses its internal traces to determine whether a client was experiencing broadband interference due to an active microwave.

The analyzer can detect microwave interference intervals by monitoring the number of physical error packets received in a time frame. Under normal circumstances, the number of physical errors remains roughly at a constant background level. However, in the presence of microwave interference, the number of physical error packets spikes, increasing to 4–10 times above the nominal level.

We take advantage of this evidence to determine time intervals of active microwaves as follows. First, we map the physical error frames to a specific offset in the period of a microwave signal. To determine this offset, we use the formula $\textit{offset} = time - \lfloor \frac{time}{T} \rfloor * T; T = \frac{1}{60}$. This offset corresponds to the phase in the 60-Hz cycle where the error occurred.

Then we sample the distribution of physical error frames across the offsets over time. Typical household microwaves do not have full-duty synchrotrons. As a result, an active microwave will generate interference for only part of the period interval. When a microwave is active within this period, a monitor will observe many more physical error frames than the default level of background physical errors; when the microwave is inactive, a monitor will observe just the background level of physical errors across an entire period.

To sample this distribution, we first partition time into intervals of 15 microwave periods (250 ms). Within an interval, we then bin the offsets and count the number of physical errors within each bin across the entire interval. We take the difference between the 2nd-largest and 2nd-smallest bins and compare it against a threshold. At the threshold or above, we suspect a microwave is in use for this time interval because of the large difference between the number of physical errors during the active phase and the background physical errors during the inactive phase. We use an initial threshold of 400 physical errors based on experience; background physical error counts in our network are 100 or less for a 250-ms interval.

The threshold is dynamically adjusted every five minutes thereafter, by averaging the difference value computed earlier across the five minute period and taking the 4th standard deviation above the mean. The threshold is then adjusted to the sum of 90% of the prior threshold and 10% of the newly computed threshold. Dynamically adjusting the threshold allows us to more accurately determine microwave periods since the number of physical errors seen by a sniffer may vary depending on how far the sniffer is from the microwave and the amount of background noise heard by the sniffer.

The microwave analyzer performs this analysis for all wireless monitors in each time interval. As a last step, the analyzer will only declare that a microwave is in use for a time interval if it suspects an active microwave for more than one monitor; in our network, at least 2–3 monitors observe physical error bursts when one microwave is active. In this case, it outputs the list of all such monitors as observing an active microwave in its internal trace.

The analyzer then estimates which clients are likely affected by the microwave during those intervals using a simple heuristic: clients in the neighborhood of monitors that have

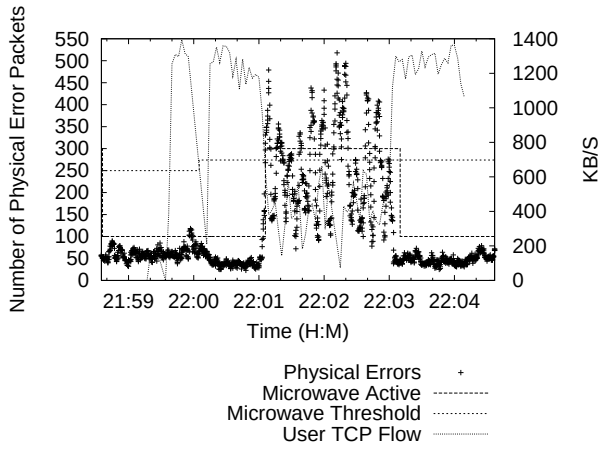

Figure 5: Physical errors observed by one wireless monitor and the output of the microwave analyzer during short periods of microwave use, and the TCP throughput of a user reporting a slow connection during this time.

detected an active microwave are also affected by the microwave. The analyzer assumes that a client is in the neighborhood of the monitors if the monitors observe most of the frames from the client. It tracks the clients seen by each wireless monitor over time by tracking unique MAC addresses per monitor. It also counts the number of frames observed for each MAC by a monitor and compares the count against the total number of frames from the client observed by the entire system. If the percentage of frames observed by the monitor is at least 70% of the total frames, then the analyzer assumes the client is in the monitor's neighborhood. For all such clients, it outputs their MAC addresses into its internal trace. The user diagnostic tool can then query the analyzer by MAC address for a given time, and the analyzer can then quickly respond whether the client was affected by a microwave at that time.

Figure 5 shows the microwave analyzer in use while diagnosing a problem reported by a user doing a bulk TCP transfer. The figure shows the number of physical errors observed by one wireless monitor over time (other nearby monitors make similar observations); each count is the number of errors in successive 250 ms intervals. It also shows the output of the analyzer as a step function. The analyzer correctly identifies the periods of microwave use; the "Microwave Active" line is low during periods of small error counts and high when the microwave is in use, resulting in large error counts. Finally, it also shows the effect of the microwave on the user's TCP throughput. When the microwave is in use, the throughput drops by over a factor of three. During this time, the analyzer would output when the microwave is active and which clients are likely to be affected. When queried using the user diagnostic tool during this time period, the tool detects the throughput degradation using the TCP analyzer (Section 4.4), determines that the microwave is active using the microwave analyzer, and decides that the microwave is most likely to blame.

4.3 Media access

The media access analyzer uses a detailed model for inferring the critical path delays of every monitored frame exchanges sent by the APs [8]. The model consists of a representation of the wired distribution network, queuing behavior in the AP, and frame transmission using the 802.11 MAC protocol. The analyzer determines the various delays an actual data (e.g. TCP packet) encountered as it traversed through the stages of the wired and wireless network path. For each AP, the analyzer uses the matched wireless frame exchanges and wired Ethernet packet from the cross reference module as external input and output events from the AP. It then emulates the AP's internal queuing and transmission states to infer various delays. These delays include fine-grained phenomena on the critical path such as AP queuing, power-save buffering, 802.11 media access and contention delays. It is used primarily by other analyzers, in particular the TCP analyzer, rather than directly by the diagnostic tools.

4.4 Transport layer

The transport analyzer models TCP performance and behavior for interactive and bulk transfer flows. First we reconstruct per-flow TCP state to infer detail TCP events based on the analysis of [11, 17], designed for wired passive monitors. The analyzer observes the TCP data sequence and ACK sequence to infer the cause of any out-of-order events like fast or regular retransmission, spurious/unneeded retransmission, and data reordering. In addition, since we have detailed information on the wireless transmissions (frame exchanges), we can further resolve some ambiguities. For example, a retransmission could be caused by the AP failing to deliver the TCP-DATA packet to the client, or the client failing to deliver the TCP-ACK packet to the AP. But if we have seen the client respond with an L2-ACK to the TCP-DATA packet to the AP, we can confirm that the client has received the packet at the TCP layer. Together with L2 information, the TCP analyzer can detect TCP-DATA and TCP-ACK losses at both sides.

Once we have reconstructed the flow characteristics, we can begin to diagnose any problems associated with the TCP flow. We first classify the flow as interactive or bulk transfer by checking the number of bytes sent by the client and server, and the number of full-sized TCP packets; bulk flows send substantially more bytes often with all but the last packet being full-sized.

For interactive flows, we calculate the response time for a TCP data sequence segment to be acknowledged. Note that this time may not be the RTT during loss recovery as it may take the sender several round trips to finally get the ACK for the data; we mark a particular data segment as having a slow response time if it exceeds a certain threshold (e.g., 200 ms for interactive connections [4]) and inspect the delay and loss characteristics during that period. If most losses happen at the Internet side, the analyzer simply returns with that error because it does not have enough information for further diagnosis.

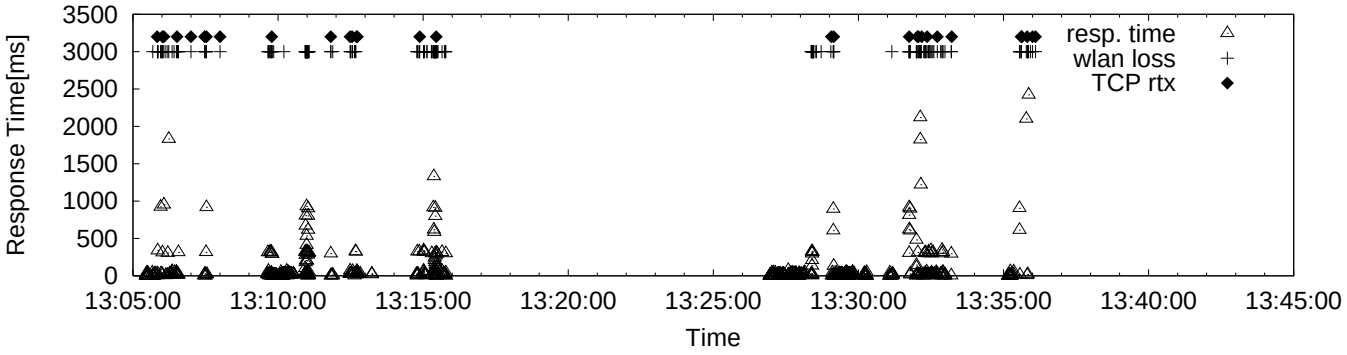

Figure 6: Response time and TCP behavior of a user who reported substantial delay with an interactive SSH session.

If the client has wireless losses, the TCP analyzer checks for broadband interference from microwaves or co-channel interference from other clients in the nearby area [9]. It also checks if the loss of the particular frame exchange is caused by loss of L2-DATA or L2-ACK transmissions. For instance, we have observed clients who suffer heavy AP to client L2-ACK losses. Although these losses would not result in a loss to higher layers since the DATA has been delivered, the client would retry excessively because it cannot receive the L2-ACKs from its AP. The excessive retries would backlog the client's TCP-ACKs, eventually causing the server to timeout and retransmit (spuriously).

If the client does not have wireless losses but instead suffers from high link delay, the TCP analyzer returns the major cause of the link delay. The cause could be contention from other clients, from other connections from the same client itself, or contention due to excessive backoff and retries [8].

As an example of the operation of the TCP analyzer, Figure 6 shows a time series graph of the response time and TCP behavior of a user reporting a slow SSH connection during a lab seminar. The graph shows the user suffers from long response times (200 ms – 2.5 seconds) even though the connection is to a server in the same department.

The TCP analyzer reports high losses from the user's client device to the AP (shown as gray crosses), meaning the client's TCP-ACK or TCP-DATA packets do not get through. It finds that most server retransmissions (shown as black diamonds) are not fast retransmissions, but caused by TCP timeout. Therefore the retransmissions usually occur after 200 ms (WindowsXP default [17]). Further, often the TCP-ACK of the retransmission is also lost, causing the SSH server to exponentially backoff the RTO timer.

The analyzer does not find microwave or hidden terminal events, and continues to look for other causes. It analyzes the rate used by the wireless transmissions, and finds that the client has 10-times higher losses for 802.11g transmissions than 802.11b transmissions. The TCP analyzer concludes the final diagnosis as high client losses caused by a poor rate adaption algorithm.

Since the user typically does not indicate the specific connections that are slow, the TCP analyzer performs diagnosis on all connections from the user. Based on the diagnosis from each connection, the analyzer returns the major cause across all diagnosis reports. We have noticed that the diagnoses across active simultaneous connections from the same user are very consistent (except server side Internet losses). More surprisingly, the analyzer reports bursts of server or client-side Internet losses across different clients. While Internet losses should be more dependent on the Internet paths and the end server, these results may indicate that our wireless gateway is dropping packets. We are still investigating this issue with our campus network operations staff.

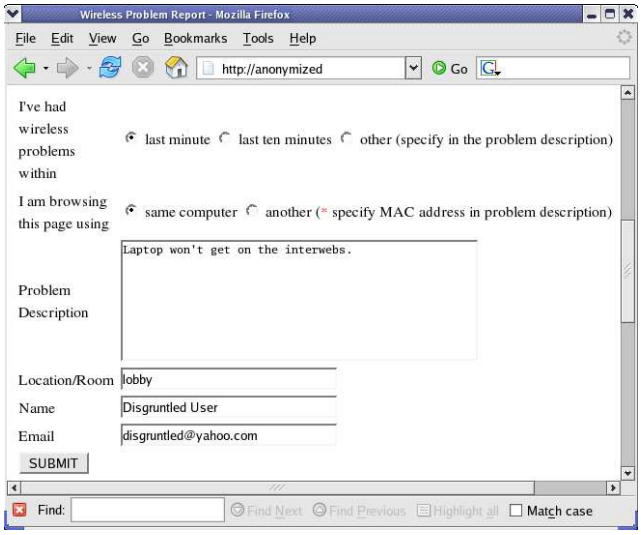

Figure 7: Shaman problem report form.

5 Shaman diagnostic tools

This section describes the two Shaman diagnostic tools, one designed for users to invoke on demand about wireless problems that they are experiencing and a second designed to alert network administrators about pervasive network problems.

5.1 User diagnostic tool

The goal of the Shaman user diagnostic tool is to answer queries on demand from users about performance problems

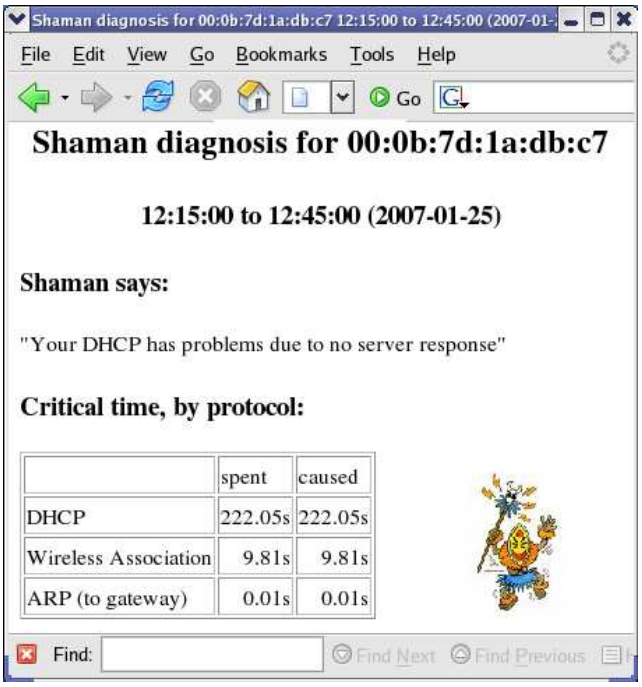

Figure 8: Shaman basic diagnosis. The problematic device has MAC address 00:0b:7d:1a:db:c7.

that they are experiencing with the enterprise wireless network. The user interface to the tool is a simple Web form with fields for identifying a user's network device, an approximate time that the problem occurred, a contact email address, and any notes about the problem for archiving. Figure 7 shows a screenshot of a form filled in to report the DHCP behavior illustrated in Figure 3.

When a user submits the form, the tool executes on a back-end server with the fields from the form as input, invokes the various network analyzers on the trace covering the time of the reported problem, evaluates the output of the analyzers, and makes a decision about the primary cause of the problem. The tool then reports this decision back to the user, accompanied with additional results for context if the user is interested in more details. Figure 8 shows a screenshot of results reported from the tool for the user experiencing the DHCP problem described in Section 4.1.2. The tool correctly identifies DHCP as the main problem in Figure 3.

The decision algorithm of the user diagnostic tool works as follows. First it examines the wireless loss rates returned by the various analyzers, including both mobility operations (e.g., DHCP or association) and during network communication (e.g., media access and TCP). If the loss rates are high, it uses the broadband analyzer to determine whether a microwave is active during that period. If so, it reports the microwave as the dominant cause.

Otherwise it compares the results of the various analyzers in more detail to determine a dominant cause. As a common basis for comparison, it uses a time duration metric for each analyzer that reports a problem. We chose time as the common metric because it intuitively matches how users tolerate and react to network problems; when a problem persists long enough to be affect the user network experience, users are sufficiently motivated to invoke a service to diagnose the problem.

For the mobility protocols, this duration is the critical time spent in the various protocols (association, DHCP, ARP). For TCP, this duration is the total time during which TCP was under performing (low throughput or high delay). The tool assigns the dominant cause for the poor user experience to the analyzer with the longest time metric. The tool then reports this time, which analyzer reported the cause, and any detailed explanation returned by the analyzer (e.g., excessive AP disassociation and re-association, timeouts while requesting DHCP leases, contention constraining TCP throughput, losses or delays causing TCP timeouts, etc.).

5.2 Network alert tool

The goal of the Shaman network alert tool is to pro-actively report serious pervasive problems to network administrators. Serious pervasive problems are those that simultaneously affect multiple clients at one or more APs and require the intervention of a network administrator to correct. Typically these problems are due to failures of critical network components (DHCP or DNS servers, routers, wireless management gateways, etc.), or persistent performance issues (e.g., poor coverage within a building [6]).

The design of the network alert tool includes analyzers that mirror the network analyzers described in 4. These alert analyzers operate continuously by invoking the network analyzers to detect pervasive problems. The tool counts the fraction of the client population affected by a specific type of event (e.g., clients performing DHCP requests who timeout). If this count is significant (we currently use a threshold of 75%), the tool triggers an alert by sending email to a wireless administration mailing list detailing the event, results from the analyzers, and the clients affected.

Motivated by experience, we have currently implemented one alert module for DHCP. The majority of our pervasive problems in which multiple users have simultaneously submitted wireless reports that all have the same underlying cause have been linked to DHCP — the DHCP server itself has failed, the router between the wireless distribution network and the DHCP server was overwhelmed by a denial-of-service attack, etc. Specifically, our DHCP analyzer informs the alert tool of clients that remain in the **Requesting** state for more than two seconds. As an example, our tool produced an alert for a particular day around noon. The alert lasted for twelve minutes, during which 6–10 clients were trying to obtain new DHCP leases. In each of the successive two-minute bins, at least 83% percent of the requesting clients were unsuccessful. The clients were associated with different APs, indicating a network-wide DHCP problem. Further investigation revealed that the DHCP server was unreachable from parts of the wireless network.

6 Conclusions

In this paper we have described Shaman, a system for performing comprehensive and automatic diagnosis on wireless problems down to fine-grained low-level root causes. Shaman combines wireless monitoring infrastructure, trace synchronization, a collection of network analyzers, and two diagnostic tools into a single diagnostic sytem for enterprise wireless networks. We have deployed a Shaman prototype in our university department building, and draw upon our experiences using the system prototype throughout to motivate its design and illustrate its use.

For more widespread deployment, we envision migrating the monitoring infrastructure used by Shaman into the access points themselves. Merging this functionality into the APs reduces the deployment cost and simplifies analysis, albeit reducing monitoring coverage. Exploring this evolution in wireless monitoring and automated diagnosis remains an exciting open problem.

References

[1] Air magnet. `http://www.airmagnet.com`.

[2] Airwave. `http://www.airwave.com`.

[3] Kismet. `http://www.kismetwireless.net`.

[4] Rsponse time overview. `http://www.useit.com/papers/responsetime.html`.

[5] A. Adya, P. Bahl, R. Chandra, and L. Qiu. Architecture and techniques for diagnosing faults in ieee 802.11 infrastructure networks. In *Proceedings of Mobicom*, 2004.

[6] P. Bahl, R. Chandra, J. Padhye, L. Ravindranath, M. Singh, A. Wolman, and B. Zill. Dair: A framework for managing enterprise wireless networks using desktop infrastructure. In *Proceedings of MobiSys*, 2006.

[7] R. Chandra, J. Padhye, A. Wolman, and B. Zill. Where are they and what are they doing: On the importance of locating clients for managing enterprise wlans. In *Proceedings of NSDI*, 2007.

[8] 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 *Proceedings of the ACM SIGCOMM Conference*, Kyoto, Japan, Aug. 2007.

[9] 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 *Proceedings of the ACM SIGCOMM Conference*, pages 39–50, Pisa, Italy, Sept. 2006.

[10] C. C. Ho, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer. A scalable framework for wireless network monitoring. In *Proceedings of the Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*, 2004.

[11] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Measurement and classification of out-of-sequence packets in a tier-1 ip backbone. *IEEE/ACM Transactions on Networking*, 15(1), 2007.

[12] Juniper Research. VoIP handsets & equipment: Current markets (2005-6) & forecasts (2007-12), 2007.

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

[14] L. Qiu, P. Bahl, A. Rao, and L. Zhou. Troubleshooting wireless mesh networks. *ACM SIGCOMM Computer Communication Review*, Oct. 2006.

[15] K. N. Ramachandran, E. M. Belding-Royer, and K. C. Almeroth. Damon: A distributed architecture for monitoring multi-hop mobile networks. In *Proceedings of SECON*, 2004.

[16] M. Raya, J.-P. Hubaux, and I. Aad. Domino: a system to detect greedy behavior in ieee 802.11 hotspots. In *Proceedings of MobiCom*, 2004.

[17] S. Rewaskar, J. Kaur, and F. D. Smith. A passive state-machine approach for accurate analysis of tcp out-of-sequence segments. *ACM SIGCOMM Computer Communication Review*, 36(3), 2006.

[18] A. Sheth, C. Doerr, D. Grunwald, R. Han, and D. Sicker. Mojo: A distributed physical layer anomaly detection system for 802.11 wlans. In *Proceedings of MobiSys*, 2006.

[19] J. Yeo, M. Youssef, and A. Agrawala. A framework for wireless lan monitoring and its applications. In *Proceedings of ACM WiSe*, 2004.