Appears in the Proceedings of the Tenth Annual International Conference on Mobile Computing and Networking (MobiCom 2004)

Architecture and Techniques for Diagnosing Faults in IEEE 802.11 Infrastructure Networks

Atul Adya
Microsoft Research
One Microsoft Way
Redmond, WA 98052
adya@microsoft.com

Paramvir Bahl
Microsoft Research
One Microsoft Way
Redmond, WA 98052
bahl@microsoft.com

Ranveer Chandra
Department of Computer
Science Cornell University
Ithaca, NY 14853
ranveer@cs.cornell.edu

Lili Qiu
Microsoft Research
One Microsoft Way
Redmond, WA 98052
liliq@microsoft.com

ABSTRACT

The wide-scale deployment of IEEE 802.11 wireless networks has generated significant challenges for Information Technology (IT) departments in corporations. Users frequently complain about connectivity and performance problems, and network administrators are expected to diagnose these problems while managing corporate security and coverage. Their task is particularly difficult due to the unreliable nature of the wireless medium and a lack of intelligent diagnostic tools for determining the cause of these problems.

This paper presents an architecture for detecting and diagnosing faults in IEEE 802.11 infrastructure wireless networks. To the best of our knowledge, ours is the first paper to address fault diagnostic issues for these networks. As part of our architecture, we propose and evaluate a novel technique called *Client Conduit*, which enables bootstrapping and fault diagnosis of disconnected clients. We describe techniques for analyzing performance problems faced in a wireless LAN deployment. We also present an approach for detecting unauthorized access points. We have built a prototype of our fault diagnostic architecture on the Windows operating system using off-the-shelf IEEE 802.11 cards. The initial results show that our mechanisms are effective; furthermore, they impose low overheads when clients are not experiencing problems.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations

General Terms

Management, Reliability

Keywords

Infrastructure wireless networks, fault detection, fault diagnosis, disconnected clients, IEEE 802.11, Rogue APs

1. INTRODUCTION

The convenience of wireless networking has led to a wide-scale adoption of IEEE 802.11 networks [22]. Corporations, universities, homes, and public places are deploying these networks at a remarkable rate. However, a significant number of "pain points" remain for end-users and network administrators. Users experience a number of problems such as intermittent connectivity, poor performance, lack of coverage, and authentication failures. These problems occur due to a variety of reasons such as poor access point layout, device misconfiguration, hardware and software errors, the nature of the wireless medium (e.g., interference, propagation), and traffic congestion.

Figure 1 shows the number of such wireless-related complaints logged by the Information Technology (IT) department of Microsoft corporation over a period of six months. The company has a large deployment of IEEE 802.11 networks with several thousand Access Points (APs) spread over more than forty buildings. Each complaint is an indication of end-user frustration and loss of productivity for the corporation. Furthermore, resolution of each complaint results in additional support personnel costs to the IT department; our research revealed that this cost is several tens of dollars and this does not include the cost due to the loss of end-user productivity.

To resolve complaints quickly and efficiently, network administrators need tools for detecting, isolating, diagnosing, and correcting faults. To the best of our knowledge, there is no previous research that addresses fault diagnostic problems in IEEE 802.11 infrastructure networks. In contrast, the importance of diagnosing these problems in the "real-world" is apparent from the number of companies that offer solutions in this space [2, 4, 20, 16, 37]. These products do a reasonable job of presenting statistical data from the network; however, they lack a number of desirable features. Specifically, they do not do a comprehensive job of gathering and analyzing the data to establish the possible causes of a problem. Furthermore, most products only gather data from the APs and neglect the client-side view of the network. Some products that monitor the network from the client's perspective require hardware sensors, which can be expensive to de-

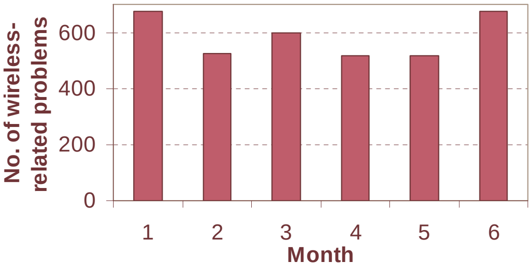


Figure 1: Number of wireless related complaints logged by the IT department of a major US corporation

MobiCom'04, Sept. 26-Oct. 1, 2004, Philadelphia, Pennsylvania, USA.

ploy and maintain. Also, current solutions do not provide any support for disconnected clients even though these are the ones that need the most help. We discuss these products in more detail in Section 8.

This paper presents a flexible architecture for detecting and diagnosing faults in infrastructure wireless networks. We instrument wireless clients and (if possible) access points to monitor the wireless medium and devices that are nearby. Our architecture supports both proactive and reactive fault diagnosis. We use this monitoring framework to address some of the problems plaguing wireless users. We present a novel technique called *Client Conduit* that enables disconnected clients to diagnose their problems with the help of nearby clients. This technique takes advantage of the beaconing and probing mechanisms of IEEE 802.11 to ensure that connected clients do not pay unnecessary overheads for detecting disconnected clients. We also present a simple technique for finding the approximate location of disconnected clients. We present a technique that uses nearby wireless clients for diagnosing wireless network performance problems. Finally, we show how our monitoring architecture naturally lends itself to detecting rogue or unauthorized access points in enterprise wireless networks. We have implemented and evaluated the basic architectural framework, Client Conduit, and Rogue AP detection on the Windows operating system using off-the-shelf IEEE 802.11 network cards; we have evaluated our other mechanisms using tools such as AiroPeek [38] and WinDump [39]. Our results show that our techniques are effective; furthermore, they impose negligible overheads when clients are not experiencing problems.

We summarize the primary contributions of our paper as follows:

- We believe ours is the first paper to identify fault diagnosis in IEEE 802.11 infrastructure networks as an important area of research. The identification of various problems in such environments is an important contribution since wireless fault diagnosis is an area that needs attention.
- We present a flexible client-based architecture for detecting and diagnosing faults in an IEEE 802.11 infrastructure network. Our fault-diagnosis approach is unique in the wireless context since we use clients (and if possible, infrastructure APs) to monitor the network and the radio frequency (RF) environment.
- We describe a simple and efficient technique called *Client Conduit* that allows disconnected clients to communicate via other nearby connected clients; this mechanism can be used to bootstrap wireless clients and resolve certain connectivity problems.
- We present novel solutions that use our architecture for detecting and diagnosing a variety of faults: locating disconnected clients, diagnosing performance problems, and detecting Rogue APs.

Our work is just a first step in the direction of self-healing wireless networks and there are a number of issues that still need to be addressed. From the vast number of wireless problems faced by end-users and network administrators everyday, we have focused only on a subset of those problems; our selection was based on conversations with network administrators [11] along with the high-priority problems observed in user-complaint logs. Even though some of our techniques are applicable to other deployments (e.g., hotspots, homes), our main emphasis has been diagnosing faults in enterprise wireless networks. We ensure that our techniques do not introduce new security attacks but we do not focus on denial-of-service and greedy MAC attacks [31].

The rest of the paper is organized as follows: In Section 2, we discuss the most important problems that users and network administrators complain about with respect to wireless LAN deployment. Section 3 describes the components of our client-based architecture. Section 4 presents the Client Conduit protocol. Section 5 focuses on locating disconnected clients, performance isolation, and Rogue AP detection. Section 6 describes the implementation of our system and Section 7 presents an evaluation of our techniques. Section 8 discusses related work. Finally, we discuss future work in Section 9 and conclude in Section 10.

2. FAULTS IN A WIRELESS NETWORK

We enumerate the most important problems that users and network administrators face when using and maintaining corporate wireless networks. Our list has been derived from interviews and discussions we conducted with network administrators and operation engineers of Microsoft's IT department. These individuals are responsible for managing over 4,400 IEEE 802.11 APs distributed over forty buildings in the company.

Connectivity problems: End-users complain about inconsistent or a lack of network connectivity in certain areas of a building. Such "dead spots" or "RF holes" can occur due to a weak RF signal, lack of a signal, changing environmental conditions, or obstructions. Locating an RF hole automatically is critical for wireless administrators; they can then resolve the problem by either relocating APs or increasing the density of APs in the problem area or by adjusting the power settings on nearby APs for better coverage.

Performance problems: This category includes all the situations where a client observes degraded performance, e.g., low throughput or high latency. There could be a number of reasons why the performance problem exists, e.g., traffic slow-down due to congestion, RF interference due to a microwave oven or cordless phone, multi-path interference, large co-channel interference due to poor network planning, or due to a poorly configured client/AP. Performance problems can also occur as a result of problems in the non-wireless part of the network, e.g., due to a slow server or proxy. It is therefore necessary that the diagnostic tool be able to determine whether the problem is in the wireless network or elsewhere. Furthermore, identifying the cause in the wireless part is important for allowing network administrators to better provision the system and improve the experience for end-users.

Network security: Large enterprises often use solutions such as IEEE 802.1x [21] to secure their networks. However, a nightmare scenario for IT managers occurs when employees unknowingly compromise the security of the network by connecting an unauthorized AP to an Ethernet tap of the corporate network. The problem is commonly referred to as the *"Rogue AP Problem"* [4, 2, 15]. These Rogue APs are one of the most common and serious breaches of wireless network security. Due to the presence of such APs, external users are allowed access to resources on the corporate network; these users can leak information or cause other damage. Furthermore, Rogue APs can cause interference with other access points in the vicinity. Detecting Rogue APs in a large network via a manual process is expensive and time-consuming; thus, it is important to detect such APs proactively.

Authentication problems: According to the IT support group's logs, a number of complaints are related to users' inability to authenticate themselves to the network. In wireless networks secured by technologies such as IEEE 802.1x [21], authentication failures are typically due to missing or expired certificates. Thus, detecting such authentication problems and helping clients to bootstrap with valid certificates is important.

In this paper, we focus on detecting RF holes, diagnosing performance problems, detecting Rogue APs, and helping a client to recover from an authentication problem via Client Conduit. As part of our future work, we will investigate diagnosis of authentication problems as well.

3. SYSTEM ARCHITECTURE

We now describe the components that make up our fault detection and diagnosis architecture.

3.1 System Requirements

Before we describe the system components, we enumerate the requirements for our system:

- We require that the software on clients be augmented for monitoring. In our system, software modifications on APs are needed only for better scalability and for analyzing an AP's performance (Section 5.2). Since our approach does not require hardware modifications, "the bar" for deploying our system is lower.
- For some of our mechanisms, we need the ability to control beacons and probes. We also require that clients have the capability of starting an infrastructure network (i.e., become an AP) or an ad hoc network on their own; this ability is supported by many wireless cards, e.g., Atheros [6], Native WiFi [26]. Whenever faced with a choice of starting an ad hoc or an infrastructure network, we prefer the latter since infrastructure mode is better supported in current cards.
- We rely on the availability of a database that keeps track of the location of all the access points; such location databases are typically maintained by network administrators.
- Some of our techniques require the presence of nearby clients or access points. With the increasing deployment of access points and the use of wireless laptops and PDAs in enterprise wireless networks, this requirement is becoming relatively easy to satisfy in these environments. In fact, based on SNMP data collected from APs over a period of two days, we observed the presence of 13-15 associated wireless clients on our floor (approximately 2500 sq. meters) during working hours of the day; thus, with such client densities, there is a high likelihood that our requirement will be satisfied.

  Compared with the existing products that require deploying special wireless sensors throughout the enterprise, our approach takes advantage of nearby clients and access points instrumented with software "sensors", thereby imposing a lower deployment cost.

3.2 System Components

Our system consists of the following components — a *Diagnostic Client* (DC) that runs on a wireless client machine, an optional *Diagnostic AP* (DAP) that runs on an Access Point, and a *Diagnostic Server* (DS) that runs on a backend server of the organization (see Figure 2). Below, we describe each of these in detail.

Diagnostic Client module or DC: The Diagnostic Client module monitors the RF environment and the traffic flow from neighboring clients and APs. Note that during normal activity, the client's wireless card is not placed in promiscuous mode. The DC uses the collected data to perform local fault diagnosis. Depending on the individual fault-detection mechanism, a summary of this data is transmitted to the DAPs or DSs at regular intervals, e.g., for Rogue AP detection, the DC in our prototype sends MAC and channel information of nearby APs every 30 seconds. In addition, the DC is geared to accept commands from the DAP or the DS to perform on-demand data gathering, e.g., switching to promiscuous mode and analyzing a nearby client's performance problems. In case the wireless client becomes disconnected, the DC logs data to a local database/file. This data can be analyzed by the DAP or DS at some future time when network connectivity is resumed.

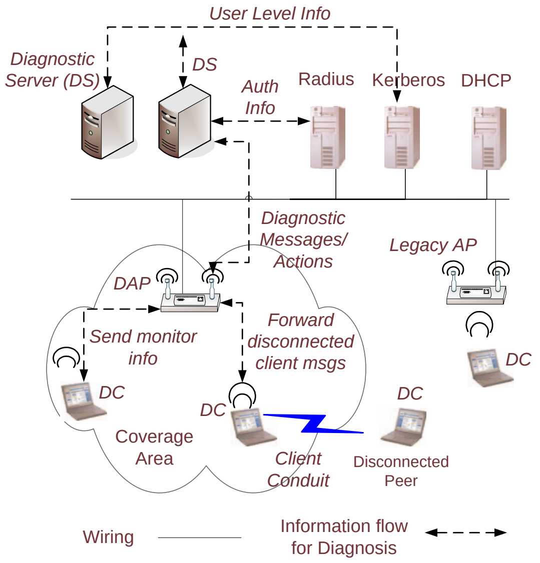


Figure 2: Fault Diagnosis Architecture

Diagnostic Access Point module or DAP: The Diagnostic AP's main function is to accept diagnostic messages from DCs, merge them along with its own measurements and send a summary report to the DS. The Diagnostic AP is not a fundamental requirement of our architecture; it is primarily needed for offloading work from the DS. Most of our techniques can work in an environment with a mixture of *legacy APs* and DAPs: if an AP is a legacy AP, its monitoring functions are performed by the DCs and its summarizing functions and checks are performed at the DS. In the rest of the paper, for the ease of exposition, we assume the presence of DAPs.

Diagnostic Server module or DS: The Diagnostic Server accepts data from DCs and DAPs and performs the appropriate analysis to detect and diagnose different faults. The DS also has access to a database that stores each AP's location. Network administrators may deploy multiple DSs in the system to balance the load, e.g., each AP's MAC address could be hashed to a particular DS. In the rest of the paper, we present our mechanisms as if one Diagnostic Server is present in the system.

Figure 2 gives a schematic view of our fault diagnosis system. As shown, the Diagnostic Server interacts with other network servers e.g., the RADIUS [32] and Kerberos [27] servers, to get client authorization and user information. Our architecture allows disconnected clients to communicate with the DS via a nearby connected client using the Client Conduit protocol; this mechanism is presented in Section 4.

Our system supports both *reactive* and *proactive* monitoring. In proactive monitoring, DCs and DAPs monitor the system continuously: if an anomaly is detected by a DC, DAP, or DS, an alarm is raised for a network administrator to investigate. The reactive monitoring mode is used when a support personnel wants to diagnose a user complaint. The personnel can issue a directive to a DC from one of the DSs to collect and analyze the data for diagnosing the problem. We believe that it is acceptable to increase the network and CPU load (on the DCs, DAPs, DSs) by a small amount during reactive monitoring; of course, in the proactive case, these overheads must be kept low.

Our architecture itself imposes negligible overheads with respect

Fault Diagnosis	Where performed	Support for legacy APs?
Help disconnected client	DC	Yes
Locate disconnected client	DS	Yes
Performance Isolation	DC and DAP	Partially
Detect Rogue APs	DS	Yes

Table 1: Different fault diagnosis mechanisms and entities that can diagnose them; the last column indicates if the solution can be supported using legacy APs

to power management: the individual techniques have to be designed to prevent unnecessary battery wastage. Both the proactive and reactive techniques presented later in this paper consume very little bandwidth, CPU, or disk resources; as a result, they should have negligible impact on battery consumption. Only during data transfer in Client Conduit does a connected client send/receive messages on behalf of a disconnected client. To ensure that the helping client's applications (or battery) are not affected significantly, it is offered a knob to control the amount of resources it wants to devote for this transfer (see Section 4.2.1).

Table 1 shows the various problems diagnosed in this paper, the entities (DCs, DAPs, and DSs) involved in the diagnosis, and whether the solution can be used with legacy APs.

3.3 System Scaling

We have designed our system to scale with the number of clients and APs in the system. The two shared resources in our system are DSs and DAPs. To prevent a single Diagnostic Server from becoming a potential bottleneck in our system, the design allows more DSs to be added as the system load increases. Furthermore, we offload work from each individual DS by sharing the diagnosis burden with the DCs and the DAPs. The DS is used only when the DCs and DAPs are unable to diagnose the problem and the analysis requires a global perspective and additional data, e.g., signal strength information obtained from multiple DAPs may be needed for locating a disconnected client. As stated earlier, the presence of legacy APs degrades scalability since the work usually performed by DAPs would need to be performed by the DSs.

Similarly, since the DAP is a shared resource, making it do extra work can potentially hurt the performance of all its associated clients. To reduce the load on a DAP, different fault diagnosis mechanisms can use a simple technique that we refer to as *Busy AP Optimization*: with this optimization, an AP does not perform active scanning if any client is associated with it; the associated clients perform these operations as needed, e.g., our Rogue AP detection requires such scans. The AP continues to perform passive monitoring activities that have a negligible effect on its performance. If there is no client associated, the AP is idle and it can perform these monitoring operations. This approach ensures that most of the physical area around the AP is monitored without hurting the AP's performance.

3.4 System Security

The interactions between the DC, DAP, and DS are secured using EAP-TLS [1] certificates issued over IEEE 802.1x. An authorized certification authority (CA) issues certificates to DCs, DAPs and DSs; we use these certificates to ensure that all communication between these entities is mutually authenticated.

We do not address malicious behavior by legitimate users in our environment. Researchers have developed techniques for detecting greedy and malicious behavior for hotspot environments [31]; others have suggested techniques to handle problems due to false information sent by malicious clients to central entities such as the DS [30]. These approaches are complimentary to our design and could be included in our system.

4. CLIENT CONDUIT

In this section, we present a novel mechanism called *Client Conduit* that allows disconnected wireless clients to convey information to network administrators and support personnel.

If a wireless client cannot connect to the network, the DC logs the problem in its database. When the client is connected later (e.g., via a wired connection), this log is uploaded to the DS, which performs the diagnosis to determine the cause of the problem. However, sometimes it is possible that this client is in the range of other connected clients; this client may be disconnected since it is just outside the range of any AP or due to authentication problems. In this situation, it would be desirable to perform fault diagnosis with the DS immediately and, if possible, rectify the problem. We now focus on this scenario.

On first thought one may ask: why not have the disconnected node simply send a message to its connected neighbor? Unfortunately, this approach does not work because IEEE 802.11 does not allow a client to be connected to two networks at the same time. Since the connected node has already associated to an infrastructure network, it cannot simultaneously connect to an ad-hoc network with the disconnected client D — if it wants to receive a message from D, it first has to disconnect and then join the ad-hoc network started by D. This is inefficient and unfair to a normally-functioning connected client.

One can imagine solving this problem using multiple radios on the connected client (one dedicated on an ad hoc network for diagnosis), or using MultiNet [14] (which allows a client to multiplex a single wireless card such that it is present on multiple networks), or by making a connected client periodically scan all channels. All these approaches have the undesirable property of penalizing the normal-case operation/costs to deal with a problem that is expected to occur infrequently. In the periodic scanning case, switching the wireless card across channels or networks can cause packet drops at the connected client. In the MultiNet case, the wireless card will periodically spend time on the ad hoc network, and will thus consume bandwidth on the connected client. On the other hand, our Client Conduit approach imposes no overheads in the common case when no disconnected clients are present in the neighborhood.

4.1 The Client Conduit Protocol

We now discuss our Client Conduit protocol that allows a disconnected client to be diagnosed by a DS via one of the connected clients. Client Conduit achieves its efficiency (of not penalizing connected clients) by exploiting two operational facts about the IEEE 802.11 protocol. First, even when a client is associated to an AP, it continues to receive beacons from neighboring APs or ad hoc networks at regular intervals. Second, a connected client can send directed or broadcast *Probe Requests* without disconnecting from the infrastructure network. We now present the Client Conduit protocol for a scenario where a disconnected client D is in the vicinity of a connected client C (see Figure 3). In the following description, we refer to the first 4 steps of the protocol aa the Connection Setup phase and the last step as the Data Transfer phase.

1. The DC on the disconnected client D configures the machine to operate in promiscuous mode. It scans all channels to determine if any nearby client is connected to the infrastructure network. If it detects such a connected client on a channel, it starts a new *infrastructure* or an *ad hoc* network on the channel on which it detected the client's packets. For the reasons discussed in Section 3.1, and for the simplicity of exposition, we assume that client D switches mode to become an AP and starts an infrastructure network.

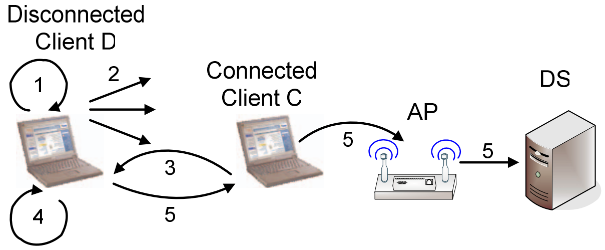


Figure 3: Client Conduit Mechanism (Steps 1 through 5 are described below)

Note: By examining the *ToDS* and *FromDS* fields of IEEE 802.11 data frames [22], client D can determine whether the data packet is part of an infrastructure network and is being sent to/from an AP.

2. This newly-formed AP at D now broadcasts its beacon like a regular AP, with an SSID of the form "*SOS_HELP_<num>*" where *num* is a 32-bit random number to differentiate between multiple disconnected clients.

3. The DC on the connected client C detects the SOS beacon of this new AP. At this point, C needs to inform D that its request has been heard and it can stop beaconing. If client C tries to connect to D, it would need to disconnect from the infrastructure network, thereby hurting the performance of C's applications. Instead, we utilize the "active scanning" mechanism of IEEE 802.11 networks — C sends a *Probe Request* of the form "*SOS_ACK_<num>*" to D. Note that the Probe Request is sent with a different SSID than the one being advertised by the AP running on D. This approach prevents some other nearby client that is not involved in the Client Conduit protocol from inadvertently sending a Probe Request to D (as part of that client's regular tests of detecting new APs in its environment).

4. When D hears this Probe Request (and perhaps other requests as well), it stops being an AP, and becomes a station again. Note that in response to the Probe Request, a Probe Response is sent out by D; client C now knows that it does not need to send more Probe Requests (it would have stopped anyway when D's beacons stopped). More importantly, D's Probe Response indicates if D would like to use client C as a hop for exchanging diagnostic messages with the DS. This response mechanism ensures that if multiple connected clients try to help D, only one of them is chosen by D for setting up the conduit with the DS.

5. Now D starts an ad hoc network and C joins this network via MultiNet [14]. At this point, C becomes a conduit for D's messages and D can exchange diagnostic messages with the DS through C.

The key advantage of the Client Conduit protocol is that connected clients do not experience unnecessary overheads during normal operation. Their overheads during the execution of the protocol are discussed later in this section.

It is important to note that the Client Conduit mechanism can also be used for *bootstrapping* clients. For example, suppose that a client tries to access a wireless network for the first time and does not have EAP-TLS certificates, but has other credentials such as Kerberos credentials; Client Conduit can be used to authenticate the user/machine with the backend Radius/Kerberos servers. New certificates can then be installed on the client machine; similarly, a client's expired certificates can also be refreshed without requiring a wired connection.

It is possible that a client D is within the range of an AP and is disconnected because of IEEE 802.1x authentication problems [11]. Client Conduit can be used if a connected client is in range as well. If there is no such client, one can dynamically configure the AP to allow D's diagnostic messages to the back end DS (or to the RADIUS servers who can forward to the DS) via the uncontrolled port [21].

4.2 Client Conduit Security and Attacks

We must ensure that the Client Conduit protocol does not introduce any new security leaks or opportuniues for denial-of-service attacks in the system. To ensure that a malicious/unauthorized client does not obtain arbitrary access to the network, the connected client allows a disconnected client's packets to be exchanged only with the DS or backend authentication servers.

We now discuss two potential abuses of Client Conduit: hurting the performance of helping clients and disguising a Rogue AP as a disconnected client.

4.2.1 Performance Degradation of Helping Clients

When a connected client C helps a disconnected client via Client Conduit, we need to ensure that C's application's performance is not adversely affected. During the Connection Setup part of Client Conduit, the connected client C simply requires processing the beacon message and sending/receiving probe messages; no messages are forwarded by C on the disconnected client's behalf. These steps not only consume negligible resources on C but they also do not result in any security leak or compromise on C; of course, C can further rate-limit or stop performing these steps if it discovers that the disconnected client is making it perform these steps often.

We now consider the Data Transfer part of the protocol for possible security and denial-of-service attacks. Switching to MultiNet mode can consume bandwidth at the connected client [14]. There are two problems that need to be addressed. First, a malicious client should not be allowed to waste a connected client C's resources by making it enter MultiNet mode unnecessarily. Second, even when helping a legitimate client, C should be able to control the amount of resources that it wants to allocate for the disconnected client D during the MultiNet transfer. The second problem can be addressed by providing a knob to the client that allows it to limit the percentage of time that it spends on the ad hoc network relative to the infrastructure network; client C may also limit this usage to save battery power. Section 7.2 characterizes the disconnected client's performance overheads due to this tradeoff.

To prevent the first problem due to malicious clients, we add the following authentication step before Data Transfer to ensure that only legitimate clients are allowed to connect via client C.

After the Connection Setup phase, client C switches to MultiNet mode for performing authentication. To prevent a denial-of-service (DoS) attack where C is forced into MultiNet mode repeatedly, C can limit the number of times per minute that it performs such an authentication step. As part of the authentication step, client C obtains the EAP-TLS machine certificate from the disconnected client and validates it (for ensuring mutual authentication, client D can perform these steps as well). If the disconnected client has no certificates or its certificates have expired, client C acts as an intermediary for running the desired authentication protocol, e.g., C could help D perform Kerberos authentication from the back end Kerberos servers and obtain the relevant tickets. If the disconnected client D still cannot authenticate, C asks D to send the last (say) 10 KBytes of its diagnosis log to C and C forwards this log to the DS. To prevent a possible DoS attack in which a malicious client tries to send this unauthenticated log repeatedly (e.g., while spoofing different MAC addresses), the connected client can limit the total amount of unauthenticated data that it sends in a fixed time period, e.g., C could say that it will send at most 10 KBytes of such data every 5 minutes.

4.2.2 Preventing Disguised Rogue APs

As discussed in Section 2, unauthorized APs are a serious security problem in an enterprise wireless network. An attacker who wants to set up an unauthorized AP and remain undetected may try to exploit the properties of Client Conduit. The attacker's AP can be set up to beacon with an SOS SSID; our Rogue AP detection mechanism (Section 5.3) will assume that this beaconing device is actually a disconnected client and not declare it as a Rogue AP. Thus, we need to distinguish between the cases where the beaconing device is a legitimate client and where it is actually a Rogue AP.

In Client Conduit, when a disconnected client becomes an AP or starts an ad hoc network during the Connection Setup and starts beaconing, it does not send or receive any data packets. Thus, if a DC ever detects an AP (or a node in ad hoc mode) that is beaconing the SOS SSID and sending/receiving data packets, the DC can immediately flag it as a Rogue device. There is another test that can be used to detect such a Rogue device: when the helping client hears the Probe Response in step 4 of the Client Conduit protocol, it knows that the disconnected client no longer needs to beacon. Thus, if the helping client continues to hear the SOS beacons after a few seconds, it can flag the device as a disguised Rogue device.

5. FAULT DETECTION AND DIAGNOSIS

This section discusses our techniques for detecting and diagnosing faults in a IEEE 802.11 wireless network. Section 5.1 describes a simple technique for locating disconnected clients. Section 5.2 presents our mechanisms for isolating performance problems and Section 5.3 describes how we detect rogue access points.

5.1 Locating Disconnected Clients

The ability to locate disconnected wireless clients automatically in a fault diagnosis system is valuable for proactively determining problematic regions in a deployment, e.g., poor coverage or high interference (locating RF holes) or for locating possibly faulty APs. A disconnected client can determine that it is in an RF hole if it does not hear beacons from any AP (as opposed to being disconnected due to some other reason such as authentication failures). To approximately locate disconnected clients (and hence help in locating RF holes), we now discuss a technique called *Double Indirection for Approximating Location* or DIAL.

As discussed earlier, when a client D discovers that it is disconnected, it becomes an AP or starts an ad hoc network and starts beaconing. To determine the approximate location of this client, nearby connected clients hear D's beacons and record the signal strength (RSSI) of these packets. They inform the DS that client D is disconnected and send the collected RSSI data. At this point, the DS executes the first step of DIAL to determine the location of the connected clients: this can be done using any known location-determination technique in the literature [8, 23]. In the next step of DIAL, the DS uses the locations of the connected clients as "anchor points" and the disconnected client's RSSI data to estimate its approximate location. This step can be performed using any scheme that uses RSSI values from multiple clients for determining a machine's location [8, 12, 23]. Since locating the connected clients results in some error, consequently locating disconnected clients with these anchor points further increases the error. In Section 7.3, we show that this error is approximately 10 to 12 meters which is acceptable for estimating the location of disconnected clients.

5.2 Network Performance Problems

Our proposed design to diagnose network performance problems comprises two lightweight components: a proactive/passive monitoring component and a reactive diagnosing component. The monitoring component runs in the background at the DC and informs the diagnosing component when it detects connections experiencing poor performance. At this point, the diagnosing component analyzes the connections and outputs a report that gives a breakdown of the delays, i.e., the extent of the delays in the wired and the wireless part, and for the latter, a further breakdown into delays at the client, AP, and the medium. Note that the monitoring component can be conservative in declaring that network problems are being encountered; a false alarm simply invokes our diagnosing component. Since this component has low overheads, invoking it has a small impact on the performance of clients and APs. These components have not been implemented yet in our current prototype but we have evaluated the effectiveness of some of these techniques using tools such as AiroPeek and WinDump.

5.2.1 Detecting Network Performance Problems

We focus on diagnosing performance problems for TCP connections since TCP is the most widely used transport protocol in the Internet. For a TCP connection, we can passively diagnose performance problems by leveraging the connection's data and acknowledgment (ACK) packets. For other transport protocols, we can determine end-to-end loss-rate and round-trip times using either active probing or performance reports (e.g., RTCP reports [33]).

Network performance problems can manifest themselves in different ways, such as low throughput, high loss rate, and high delay. We do not use throughput as a metric for detecting a problem since it is dependent on the workload (i.e., the client's application may not need a high throughput) and on specific parameters of the transport protocol (e.g., initial window size, sender and receive window size in TCP). Instead, we use packet loss rate and round-trip time for detecting performance problems.

Estimating the round trip time (RTT) in a TCP connection is simple: if the client is a sender, it already keeps track of the RTT; if the client is a receiver, it can apply the heuristic proposed in [40] to estimate the round-trip time.

To estimate the loss rate, we use heuristics suggested in [18] and [5] on the client side. We compute different loss rates for packets sent and received by the client. For data packets sent by the client, the loss rate is estimated as the ratio of retransmitted packets to the packets sent over the last L RTTs [5]. This estimation mechanism assumes that the TCP implementation uses Selective ACKs so that loss rate is not overestimated unnecessarily; this is a reasonable assumption since a number of operating systems now support this option by default, e.g., Windows, Linux, Solaris. As shown in [5], this estimate can be higher than the actual loss rate when timeouts occur in a TCP connection. For our purposes, this inaccuracy is acceptable for two reasons: first, if a TCP connection is experiencing timeouts, it is probably experiencing problems and is worth diagnosing; second, the only consequence of a mistake is to trigger our diagnosis component, which incurs low overhead. If more accurate analysis is needed, the LEAST approach suggested in [5] can be used.

For the data packets received by the client, we use an approach similar to the one suggested in [18] to estimate the number of losses: if a packet is received such that its starting sequence number is not the next expected sequence number, the missing segment is considered lost. The loss rate is estimated as the ratio of lost packets to the total number of expected packets in the last L RTTs. Note that the expected number of bytes is calculated as the maximum observed sequence number minus the minimum during the last L RTTs; we apply the idea in [40] to estimate maximum segment size (MSS), and estimate the number of packets by dividing the number of bytes by MSS. Our assumption is that segments are rarely delivered out-of-order in a TCP connection (which has been observed by others [10]).

Our detection component triggers the diagnosis component if a connection is very lossy or it experiences high delay. A connection is detected as experiencing high delays if the RTT of a particular packet is more than 250 msec or is higher than twice the current TCP RTT [41]. To avoid invoking our diagnosis algorithm for high delays that occur temporarily, we flag a connection only when D or more packets experience a high delay. A connection is classified as lossy if its loss rate (for transmitted or received packets) is higher than 5% [28, 41].

Both D and L are configurable parameters and each represents a tradeoff between responsiveness of the detection component and unnecessary invocation of the diagnosis component. That is, with a low value of D or L, any change in delays/losses will be detected quickly but it may also result in invoking the diagnosis component unnecessarily. For high values, apart from slow responsiveness, another problem occurs: the TCP connection may end before sufficient number of samples have been collected. Such a situation can occur with short Web transfers. We can alleviate this problem by aggregating loss rate and delay information between the client and remote hosts across TCP connections. We are currently exploring such techniques along with choosing appropriate values of D and L.

5.2.2 *Isolating Wireless and Wired Problems*

When the DC at a client detects a network performance problem for a TCP connection, it communicates with its associated DAP to differentiate between the delays on the wired and wireless parts of the path. The DAP then starts monitoring the TCP data and ACK packets for that client's connection. If the DC is the sender in the TCP connection, the DAP computes the difference between the received time of a data packet from the client to the remote host and the corresponding TCP ACK packet; this time difference is an estimate of the delay incurred in the wired network. To ensure that the roundtrip time estimate is reasonable, various heuristics used by TCP need to be applied to these roundtrip measurements as well, e.g., Karn's algorithm [34]. The DAP sends this estimate to the DC who can now determine the wireless part of the delay by subtracting this estimate from the TCP roundtrip time. A similar approach can be used to compute this breakdown when the client is a receiver: the DAP determines the wireless delay by monitoring the data packets from the remote host to the client and the corresponding ACK packets. Note that the amount of state maintained at the DAP is small since it corresponds to the number of unacknowledged TCP packets; this can be reduced further by sampling.

5.2.3 *Diagnosing Wireless Network Problems*

A client may experience poor wireless performance due to a number of reasons, such as an overloaded processor at the AP or the client, problems in the wireless medium, some driver or other kernel issues at either the AP or the client. We quantify the effect of these problems by observing their impact on packet delay in the wireless network path. We group these performance problems into three categories: packet delay at the client, packet delay at the AP, and packet delay in the wireless medium. In this section, we describe a collaborative scheme, called *Estimating Delay using Eavesdropping Neighbors* or EDEN, which leverages the presence of other clients to quantify the delay experienced in each of the above categories. Since electromagnetic waves travel at the speed of light, we can safely assume that RF propagation delays are negligible relative to the client or AP delays.

When a client D's performance diagnosis component is triggered, it starts broadcasting packets asking for diagnosis help from nearby clients. All clients who hear these packets switch to promiscuous mode and ask the DAP to start the diagnosis (Section 7.1 shows that the CPU overheads of entering promiscuous mode are low on modern processors). Security mechanisms similar to the ones discussed in Section 4.2 can be used to prevent attacks on these clients. Note that we use multiple snooping clients in EDEN primarily for robustness: multiple clients increase the likelihood that at least one client hears the EDEN protocol requests and responses discussed below.

EDEN proceeds in two phases. In the first phase, the DAP to which client D is associated estimates the delay at D. The DAP periodically (say every 2 seconds) sends *Snoop request* packets to client D. When D receives a Snoop request packet, it immediately replies with a *Snoop response* message. The eavesdropping clients log the time when they hear a Snoop request and the first attempt by D to send the corresponding Snoop response packet, i.e., we only record the times of response packets for which the retransmission bit is clear. If an eavesdropping client misses either of these packets, it ignores the timing values for that request/response pair. The difference between the recorded times is the *client delay*, i.e., application and OS delays experienced by D after receiving the request packet. For robustness, Snoop requests are sent a number of times (say 20); the client and AP delays are averaged over all these instances.

In the second phase, a similar technique is used to measure the *AP delay*, i.e., client D sends the Snoop request packets and the AP sends the responses. Client D also records the round trip times to the AP for these Snoop requests and responses along with the number of request packets for which it did not receive a response, e.g., the request or response was lost.

Strictly speaking, this client and AP delay also includes the delay due to contention experienced in the wireless medium. In Section 7.4, we discuss the extent of inaccuracy introduced in EDEN's estimates due to traffic congestion.

At the end of the protocol, all the eavesdropping clients send the AP and client delay times to the client D. The difference between the round trip time reported by D, and the sum of the delays at the client and the AP, approximates the sum of the delay experienced by the packet in the forward and backward wireless link. The client can then report the client/AP/medium breakdown to the network administrator; it can also report the percentage of unacknowledged request packets as an indicator of the network-level loss rate on the wireless link.

5.3 Rogue AP Detection

As discussed in Section 2, Rogue APs are unauthorized APs that have been connected to an Ethernet tap in an enterprise or university network; such APs can result in security holes, and unwanted RF and network load. Rogue APs are considered a major security issue for enterprise wireless LANs [4, 2, 15].

Our architectural framework of using clients and (if possible) APs to monitor the environment around them naturally lends itself for detecting Rogue APs. Our basic approach is to make clients and DAPs collect information about nearby access points and send it to the DS. When the DS receives information about an AP X, it checks the AP location database and ensures that X is a registered AP in the expected location and channel.

5.3.1 *Assumptions*

We assume that all Rogue APs and the corresponding connected "rogue" clients use off-the-shelf IEEE 802.11-compliant hardware. Our approach essentially "raises the bar" such that non-compliant APs with low-level modifications are needed to defeat our scheme: to avoid detection, an attacker must modify the Rogue AP to not beacon and not respond to probe requests. Of course, an attacker can simply use a proprietary access point or one with different technology, e.g. HIPERLAN. Detecting such intruders requires special hardware and is not our goal. We simply want a low-cost mechanism that addresses the (common case) Rogue AP problem being faced in current deployments: for many networks administrators, the main goal is to detect APs inadvertently installed by employees for experimentation or convenience [11]. As part of future research, we will investigate the detection of non-compliant Rogue access points and clients as well.

If two companies have neighboring wireless networks, our mechanisms will classify the other companies' access points as Rogue APs. If this classification is unacceptable, the network administrators of the respective companies can share their AP location databases.

5.3.2 *Monitoring at clients and APs*

In our system, each DC monitors the packets in its vicinity (non-promiscuous mode), and for each AP that it detects, it sends a 4-tuple $<$ *MAC address, SSID, channel, RSSI* $>$ to the DS. Essentially, the 4-tuple uniquely identifies an AP in a particular location and channel. To get this information, a DC needs to determine the MAC addresses of all APs around it.

The DC can obtain the MAC address of an AP by switching to promiscuous mode and observing data packets (it can use the FromDS and ToDS bits in the packet to determine which address belongs to the AP). However, we can achieve the same effect using a simpler approach: since IEEE 802.11 requires all APs to broadcast beacons at regular intervals, the DC can obtain the MAC addresses from the APs' beacons from all the APs that it can hear. In Section 7.5.1, we show that a DC not only hears beacons on its channel but it may also hear beacons from overlapping channels as well; this property increases the likelihood of a Rogue AP being detected.

To ensure that we do not miss a Rogue AP even if no client is present on any channel overlapping with the AP, we use the *Active Scanning* mechanism of the IEEE 802.11 protocol: when a client wants to find out what APs are nearby, the client goes to each of the 11 channels (in 802.11b), sends Probe Requests and waits for Probe Responses from all APs that hear those Probe Requests; from these responses, the DC can obtain the APs' MAC addresses. Every IEEE 802.11-compliant AP must respond to such requests and in some chipsets [26], no controls are provided to disable this functionality. Consistent with our framework, we use the Busy AP Optimization (see Section 3.3) so that active scans in an AP's vicinity are performed by the AP only when it has no client associated with it.

5.3.3 *Analysis at the DS*

When the DS receives information for an AP from various clients, it uses DIAL to estimate the AP's approximate location based on these clients' locations and the AP's RSSI values from them.

The DS classifies an AP as rogue if a 4-tuple does not correspond to a known legal AP in the DS's AP location database, i.e., if the MAC address is not present in the database, or if the AP is not in the expected location, or the SSID does not correspond to the expected SSID(s) in the organization. Note that if an AP's SSID corresponds to an SOS SSID, the DS skips further analysis since this AP actually corresponds to a disconnected client that is executing the Connection Setup phase of the Client Conduit protocol. The channel information is used in a slightly different way. As stated above, if an AP is on a certain channel, it is possible to be heard on overlapping channels. Thus, an AP is classified as rogue only if it is reported on a channel that does not overlap with the one on which it is expected. Note that if the channel on an AP is changed, the DAP can ask the DS to update its AP location database (recall that the communication between the DAP and the DS is authenticated; if the AP is a legacy AP, the administrator can update the AP location database when the AP's channel is changed). The checks that the DS executes are summarized in Figure 4.

A Rogue AP R may try to use MAC address spoofing to avoid being detected, i.e., send packets using the MAC address corresponding to a real AP G. However, the DS can still detect R since R will reside in a different location or channel than G (if it is on the same channel and location, G would immediately detect it). Note that our approach will also detect a Rogue AP even if it does not broadcast the SSID in its beacons since a DC can still obtain the AP's MAC address from the beacon. Of course, we can detect such unauthorized APs in an even simpler manner by disallowing APs that do not broadcast SSIDs in an enterprise LAN.

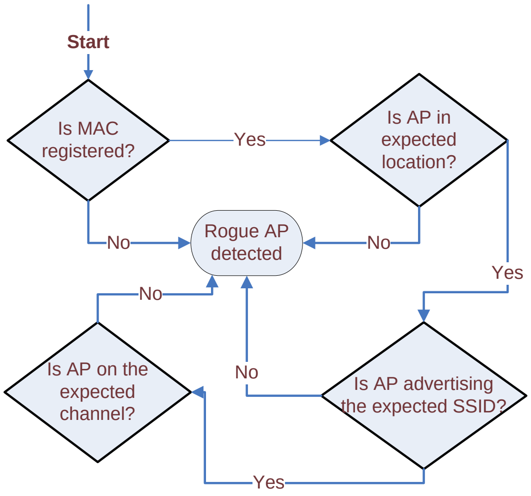

Start
Is MAC registered?
Yes
Is AP in expected location?
No
Rogue AP detected
No
Yes
No
Is AP on the expected channel?
No
Is AP advertising the expected SSID?
Yes

Figure 4: Decision steps taken by the DS to determine if an AP is a Rogue AP or not

Thus, given the above strategy, an unauthorized AP may stay undetected for a short time by spoofing an existing AP X near X's location, beacon a valid SSID in the organization, and stay on a channel on which *no DC or AP* can overhear its beacons. However, when a nearby client performs an active scan, the Rogue AP will be detected; as we show in Section 7.5.2, a DC can easily perform such a scan every 5 minutes.

6. IMPLEMENTATION

We now describe the details of our fault diagnosis implementation. We have implemented the basic architecture consisting of the DC, DAP and DS daemons; the authentication and logging mechanisms have not been implemented. We have also implemented the Client Conduit protocol and the Rogue AP detection mechanism. The support for DIAL and EDEN is currently being added.

Our system has been implemented on the Windows operating system with Netgear MA 521 802.11b cards. On the DS, we simply run a daemon process that accepts information from DAPs. The DS reads the list of legitimate APs from a file; support for reading this information from a database can be easily added. The structure of the code on the DC or DAP consists of a user-level daemon and kernel level drivers (see Figure 5). These pieces are structured such that code is added to the kernel drivers only if the functionality cannot be achieved in the user-level daemon or if the performance penalty is too high.

Kernel drivers: There are two drivers in our system — a miniport driver and an intermediate driver (IM driver) called the Native WiFi driver [26].

The miniport driver communicates directly with the hardware and provides basic functionalities such as sending/receiving packets, setting channels, etc. It exposes sufficient interfaces such that functions like association, authentication, etc. can be handled in the IM driver. The IM driver supports a number of interfaces (exposed via ioctls) for querying various parameters such as the current channel, transmission level, power management mode, SSID, etc. In addition to allowing the parameters to be set, it allows the user-level code to request for

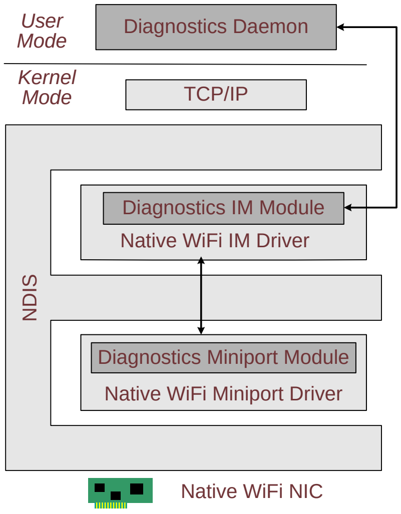


Figure 5: Components on DC and DAP

active scans, associate with a particular SSID, capture packets, etc. In general, it provides a significant amount of flexibility and control to the user-level code.

Even though most of the required operations were already present in the IM driver, we still had to make some modifications to expose certain functionalities and to improve performance of our specific protocols. The miniport driver was changed minimally to expose certain types of packets to the IM driver. In the IM driver, we added the following support:

- Capturing packet headers and packets: We allow filters to be set such that only certain packets or packet headers are captured, e.g., filters based on specific MAC addresses, packet types, packet subtypes (such as management and beacon packets), etc.
- Storing the RSSI values from received packets: We obtain the RSSI value of every received packet and maintain a table called the *NeighborInfo table* that keeps track of the RSSI value from each neighbor (indexed on the MAC address). We maintain an exponentially weighted average with the new value given a weighting factor of 0.25. The RSSI information is needed for estimating the location of disconnected clients and APs using Dial.
- Keeping track of AP information: In the NeighborInfo table, we keep track of the channels on which packets were heard from a particular MAC address, SSID information (from beacons), and whether the device is an AP or a station. This information needs to be sent to the DAP/DS for Rogue AP detection.
- Kernel event support for protocol efficiency: We added an event that is shared between the kernel and user-level code. The kernel triggers this event when an "interesting" event occurs; this allows some of our protocols to be interrupt-driven rather being polling-based. Currently, the kernel sets this event whenever it hears an SOS beacon from a disconnected client during Client Conduit, thereby resulting in a lower protocol latency (see Section 7.2).
- We added a number of ioctls to get and clear the information discussed above.

Fault Diagnostic daemon: This daemon gathers information and implements various mechanisms discussed in this paper, e.g.., collect MAC addresses of APs for Rogue AP detection, perform Client Conduit, etc. If the device is an AP, it communicates diagnostic information with the DS and the DCs; if the device is just a DC, it communicates with its associated AP to convey the diagnostic information.

The Diagnostic daemon on the DC obtains the current NeighborInfo table from the kernel every 30 seconds. If any new node has been discovered or if the existing data has changed significantly (e.g., RSSI value of a client has changed by more than a factor of 2), it is sent to the DAP. The DAP also maintains a similar table indexed on MAC addresses. However, it only sends information about disconnected clients and APs to the DS; otherwise, the DS would end up getting updates for every client in the system, making it less scalable. The DAP sends new or changed information about APs to the DS periodically (30 seconds in our current prototype). Furthermore, if the DAP has any pending information about a disconnected client D, it informs the DS immediately so that D can be serviced in a timely fashion.

All messages from the DC to the DAP and DAP to the DS are sent as XML messages. A sample message format from the DC is shown below (timestamps have been removed):

```
<DiagPacket Type="RSSIInfo" TStamp="...">
 <Clients TStamp="...">
  <MacInfo MAC="00:40:96:27:dd:cc" RSSI="23"
    Channels ="19" SSID="" TStamp="..."/>
 </Clients>
 <Real-APs TStamp="...">
  <MacInfo MAC="00:20:a6:4c:c7:85" RSSI="89"
    Channels="12" SSID="UNIV_LAN" TStamp="..."/>
  <MacInfo MAC="00:20:a6:4c:bb:ad" RSSI="7"
    Channels="10" SSID="EXPER" TStamp="..."/>
 </Real-APs>
 <Disconnected-Clients TStamp="...">
  <MacInfo MAC="00:40:96:33:34:3e" RSSI="57"
    Channels="2048" SSID="SOS_764" TStamp="..."/>
 </Disconnected-Clients>
</DiagPacket>
```

As the sample message shows, the DC sends information about other connected clients, APs, and disconnected clients. For each such class of entities, it sends the MAC address of a machine along with RSSI, SSID, and a channel bitmap which indicates the channels on which the particular device was overheard.

7. SYSTEM EVALUATION

We now evaluate our mechanisms and show that they are not only effective but they also impose low overheads. For the basic architecture evaluation, Client Conduit, and Rogue AP detection, we use our prototype. To demonstrate the effectiveness of Eden and Dial, we use a combination of tools such as AiroPeek [38] and WinDump [39].

Section 7.1 presents the timings for individual operations that are used by our protocols. Section 7.2 presents the breakdown of the costs involved in the Client Conduit mechanism and shows that it can be used to help disconnected clients in a timely manner. Section 7.3 show the effectiveness of our Dial technique for locating disconnected clients. In Section 7.4, we evaluate the effectiveness of the Eden technique to isolate performance problems. Section 7.5 shows that the scanning requirements of our Rogue AP detection mechanism imposes low overheads on client machines. Finally, in Section 7.6, we discuss scalability issues with respect to the Client Conduit protocol, Dial, Eden, and Rogue AP detection mechanisms.

7.1 Cost of Individual Operations

To better understand the cost of various operations involved in our detection and diagnosis mechanisms (e.g., Client Conduit), we ran a series of micro-benchmarks. We believe that these numbers are valuable for other researchers for modeling purposes as well. Table 2 shows the results. Note, the cost of changing a machine from AP to Station mode is less than 2 seconds (731 msecs for the actual change and then waiting for a few hundred msecs as specified by the hardware specifications).

Operation	Time (msecs)	Std. dev
Mostly No-op Ioctl (U)	0.096	0.0008
RPC-based Ioctl (U)	5.72	0.29
Set channel	177.56	7.52
Set beacon period	71.43	7.73
Set AP/STA mode	731.77	232.53
Active Scan	1901.04	14.73
Set SSID	64.73	5.47

Table 2: Times for different operations: U means time measured from user-level code; rest are times taken for the corresponding *ioctl* to complete

Additionally, we ran some experiments to understand the overheads of placing a card in promiscuous mode. We first ran an experiment with 4 machines, A, B, C, and D to determine if placing a machine in promiscuous mode has any effect on the machine's incoming/outgoing bandwidth. We setup the machines such that machine A did a TCP transfer to C at full blast and B performed a full blast TCP transfer to D. The experiment was performed three times; in each case, machine C was placed in normal mode first and then in promiscuous mode. We observed that C's throughput was largely unaffected by being in promiscuous mode: C achieved an incoming bandwidth of 254.7 KB/sec (standard deviation of 63.7 KB/sec) in the normal mode case and a bandwidth of 252.3 KB/sec (standard deviation of 21.7 KB/sec) in the promiscuous mode case.

We ran another experiment to determine a machine's CPU utilization when it is placed in promiscuous mode. In this case, we ran a full blast TCP transfer between two machines A and B; during this process, we first placed machine M in normal mode and then in promiscuous mode. Figure 6 shows the CPU overhead for machine M (a 1 GHz Pentium III machine). Even for such a relatively old machine, the CPU overhead of placing it in promiscuous mode is quite low, mostly staying below 10%; we also observed that none of the packets were dropped. Thus, these results show that the CPU overheads on a machine due to promiscuous mode are reasonably low.

7.2 Client Conduit

To measure the performance of the Client Conduit protocol, we set an experiment with one AP, one connected client C and a disconnected client D. The connected client is a 1 GHz Pentium III machine and the disconnected machine is a 800 MHz Pentium III machine. Both machines have 512 MB of memory and Netgear MA521 802.11b cards.

Figure 7 shows the total time taken along with a breakdown of the Connection Setup part of the protocol. "User time" indicates the end-to-end time taken by our user-level implementation whereas "Kernel time" indicates the time taken by the relevant *ioctls* for the same functionality. The costs in both cases are similar thereby justifying our approach of implementing only the essential mechanisms at the kernel level and driving most of the protocol from the user-level (for ease of debugging). In the first two bars, the user-level daemon at the connected client shares an event with the kernel who immediately informs the daemon when a disconnected client's beacon is detected (See Section 6). Thus, the disconnected client needs to wait only a short time before it hears the Probe Request message from the connected client C indicating that C is ready to help (see the "Get ACK" times). This delay would be much higher if the daemon obtained the disconnected machine information from the kernel periodically instead of being interrupt-driven. The third bar shows the delay breakdown for an implementation where the daemon client polls for this information every 10 seconds from the kernel (from a disconnected client's perspective, the "Get ACK" delay is higher).

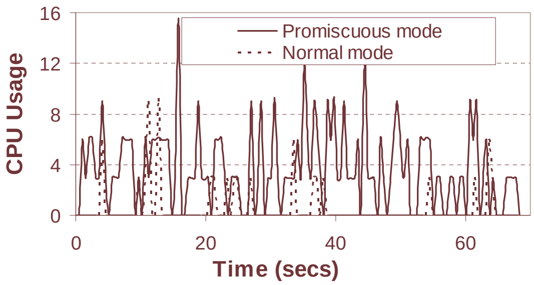


Figure 6: CPU usage in Promiscuous mode (1 GHz machine)

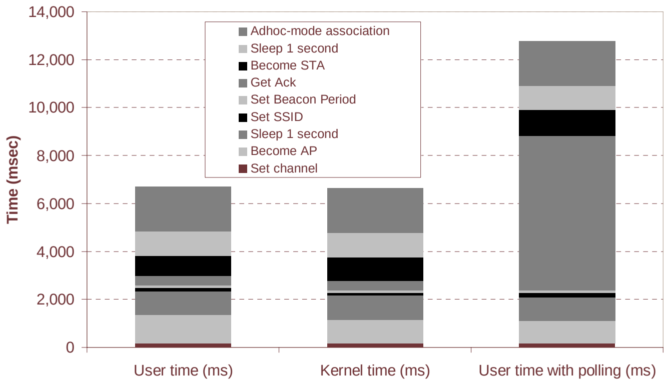


Figure 7: Breakdown of costs for Client Conduit. The protocol steps are executed from the bottom entry in the legend to the topmost, i.e., starting at "Set channel".

We now clarify a couple of details about our experiment. First, the initial step of setting the channel and checking for available clients takes approximately 190 msecs. In the worst case, the disconnected client may have to scan all channels and check for connected clients; in that case, this step may take an 2-3 seconds. Second, the steps in which we set the AP/Station mode of the machine take approximately 730 msec; however, the hardware specifications require that the operating system must wait for a few hundred milliseconds before using the card in the new mode. For robustness, we added a one second delay after such a mode change; the figure includes these delays after each mode change.

From the figure, one can see that the Connection Setup and association time for the disconnected client is quite reasonable: it takes less than 5 seconds to run the setup and another 1.9 seconds to associate with a connected client C in ad-hoc mode so that the MultiNet protocol can be started on C.

After MultiNet starts running on the connected client, the disconnected client can interact with the DS to diagnose its problems, e.g.,

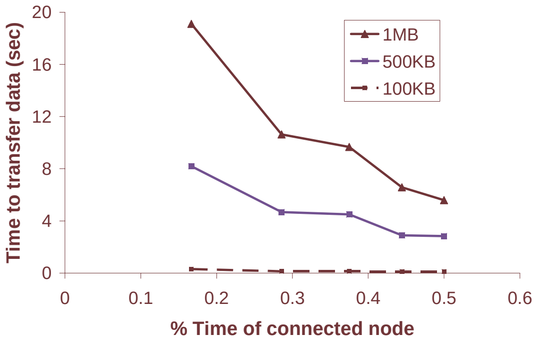


Figure 8: Time taken by a disconnected client to transfer data via Multinet

transfer certificates or log files to the DS. To evaluate the time taken to perform these transfers via MultiNet, we ran an experiment in which a machine D sent files of different sizes (100KB, 500KB and 1MB) to the DS through connected client C. Figure 8 shows the time taken when the connected client C allows 17-50% of its time to be used for ad hoc mode; client C stays on the infrastructure network for 500 msecs, and the time on the ad-hoc network is varied between 100 to 500 msecs. In our experiment, the time to switch from ad-hoc to infrastructure mode is 500 msecs and from infrastructure to ad hoc mode is 300 msecs.

As expected, the results show that the file transfer speed is a direct function of the time a connected client stays in the ad hoc network. We expect that as the switching delay overhead reduces (as in newer cards) the transfer speeds will improve.

Thus, our results show that Client Conduit allows a disconnected client's problem to be reported (and may be even resolved, e.g., updating expired certificates) in a few seconds.

7.3 Location Determination

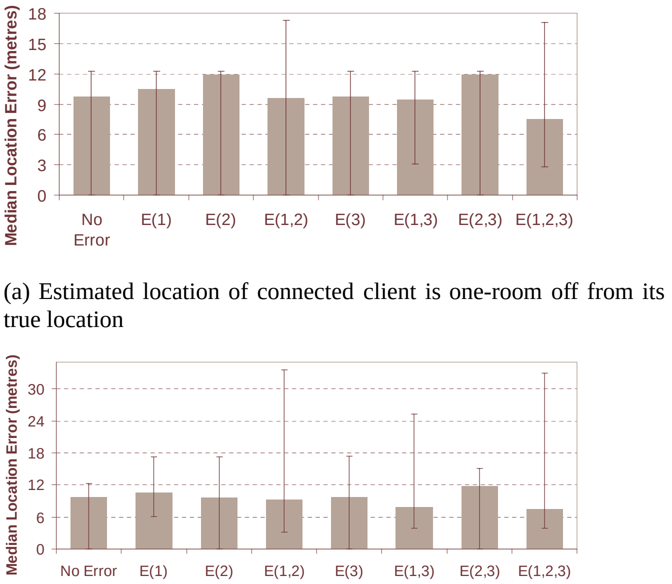
We now evaluate the accuracy of locating disconnected clients (or Rogue APs) using our DIAL scheme described in Section 5.1. Unlike previous work on location determination, DIAL incurs extra error since the location of reference points themselves may not be known accurately.

We evaluated DIAL using RADAR [8] for locating the disconnected clients from the anchor points due to its simplicity; more sophisticated RSSI-based schemes such as the one suggested in [23] can be used for reducing the errors of DIAL even further.

In our experiment, we placed 3 connected clients in 3 offices on the same floor of our building. We obtained the floor map, and applied the Cohen-Sutherland line-clipping algorithm [19] to compute the number of walls between each of the three connected clients and the other rooms. We placed a disconnected client at 7 different locations while it sent out broadcast packets. We used AiroPeek [38] to measure the RSSI of the disconnected client's packets received at the connected machines. We then applied the equation specified in [8] to compute the wall attenuation factor (WAF). Based on the WAF, we inferred that the disconnected client is in location X if the predicted signal strength at X is closest to the observed signal strength at the three connected clients.

We ran the RADAR algorithm on the collected RSSI data for locating the disconnected client D using the precise location of the connected clients. We computed the error in D's predicted location with respect to its actual location; the "No Error" bar in Figure 9(a) shows this error. Then, we ran the algorithm again by assuming that there was an error in estimating the location of one connected client by a distance of 3.3 meters; this distance corresponds to the average width of a room in our building. For example, if connected client A was placed in room X, we assume its estimated location to be a neighbouring room Y when using it as an anchor point in RADAR. The second bar in Figure 9(a) shows this error when such a situation occurs. The rest of the bars show the error in locating the disconnected client when the location of either one, two or three connected clients is estimated incorrectly by one room; Figure 9(b) shows the error when estimated location is off by a distance equivalient to that of two rooms.

The results show that when there is no error in the known location of the connected clients, the median error is 9.7 meters. This error increases to at most 12 meters when the estimated location of one or more clients is one or two rooms off from its true location. Of course, when the estimated locations of the connected clients are off by two rooms, the maximum error is substantially higher, e.g., 33 meters for the case when the location of all three clients is incorrect. This case occrs when the estimated locations of the connected clients are off in different directions, e.g., client A's location is off towards north and client B's location is off towards south.



(a) Estimated location of connected client is one-room off from its true location

(b) Estimated location of connected client is two-rooms off from its true location.

Figure 9: Median error in locating disconnected clients. The lower and upper bounds of the error bars correspond to the min and max error. $E(i)$ denotes that the i^{th} connected client's location contains error.

Note that the error in the location of the anchor points (i.e., connected clients) can be kept low (less than one room off) by using mechanisms such as Cricket [29] and Active Badges [36] for locating connected clients. With accurate location of anchor points, DIAL's error would be similar to that of the best-known RSSI-based location mechanism. Note that even an error of 10-12 metres (for our experimental setup using RADAR) is acceptable since the goal of DIAL is to approximately locate disconnected clients or Rogue APs. Thus, based on our results, we can say that DIAL is a practical approach for helping network administrators estimate the approximate location of problematic areas.

7.4 Estimating Wireless Delays

In Section 5.2.3, we presented the EDEN scheme that uses nearby clients to measure the delay encountered by a wireless station or an AP. We now show that EDEN can estimate the delay encountered at these endpoints with reasonable accuracy.

The EDEN technique measures the time spent on a client (or an AP) by measuring the times of the Snoop request and response packets at nearby clients. However, this measurement includes the delay at the machine due to medium contention. To understand the extent of this congestion delay, we set up a simple experiment with 4 machines A, B, C and D on the same channel. Machine A performed a full-blast data transfer to machine B, thereby creating traffic congestion in the medium. Then we associated client C with the Native WiFi AP machine D. The Native WiFi AP then sent 20 ping packets to the associated client, which in turn sent ping reply packets. We ran the experiment twice: once with no extra client delays and next when an extra 40 msec were added at the client between the ping request and replies. Using a fifth machine running Airopeek, we ob-

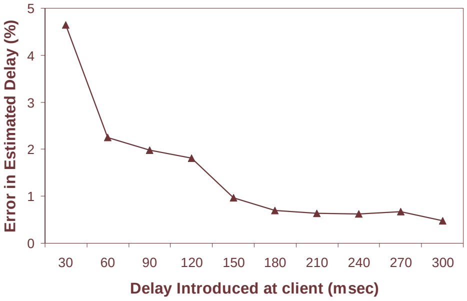


Figure 10: EDEN**'s accuracy of estimating the delay at a client**

served that EDEN over-estimated the client delay by approximately 3 msec. When examining scenarios where the client or the AP are the bottlenecks, such inaccuracies may be acceptable. However, when these entities are not bottlenecks or when EDEN is examining a scenario with low delays or when contention is even worse (e.g., the contention delay can even be more than 20 msec in 802.11b), a better estimation is needed; we are currently exploring mechanisms to reduce such inaccuracies.

Next, we ran an experiment to determine EDEN's accuracy in determining delays at an endpoint. In this setup, a client machine was associated with another machine running as an access point; both machines had Netgear MA521 802.11b cards and the corresponding Native WiFi drivers. We then injected delays in the path of all packets at the client (varying from 30 to 300 msecs). To emulate the EDEN protocol, the AP sent 20 ping packets to the client; the ping packets and replies emulate the Snoop request and response messages in EDEN. A third machine running AiroPeek was used to snoop on these ping packets; this machine effectively emulates the eavesdropping client in EDEN. The collected Airopeek data was then analyzed to estimate the delays at the client. Figure 10 shows that EDEN is reasonably accurate in estimating the delays at an endpoint: EDEN can estimate client delays with an error less than 5% of the actual introduced delay.

Finally, we studied EDEN's effectiveness in classifying the delays at the client, AP, and the medium. We used a 3-machine setup similar to the one in the previous experiment; in this case, to estimate delays at the AP, the client also sent ping packets to the AP. To introduce delays in the medium, we increased the distance between the client and the AP. The medium delay increased relative to the case when the AP and client were nearby beacuse there were more retries (the increased distance resulted in an increase in the number of walls between the two machines, thereby weakening the received signal). For better accuracy, we ran these experiments in the night when the wireless traffic was expected to be low (since the corporate LAN is actively used by employees during the day, we did not want traffic interference to affect our measurements).

Figure 11 shows EDEN's breakdown for three different scenarios. The 40-40-near bar corresponds to the scenario when the AP and client were placed near each other, and we added a 40 msec delay to all packets at both machines. The 40-40-far scenario is similar except that client and the AP were placed far from each other. Finally, the 0-0-far case is one in which we did not introduce any delays at the client or the AP but they were placed far from each other.

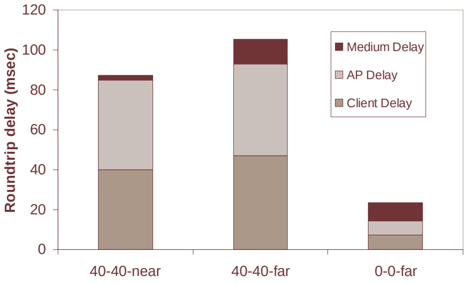


Figure 11: Breakdown of delay at the client, AP, and the medium as estimated by EDEN

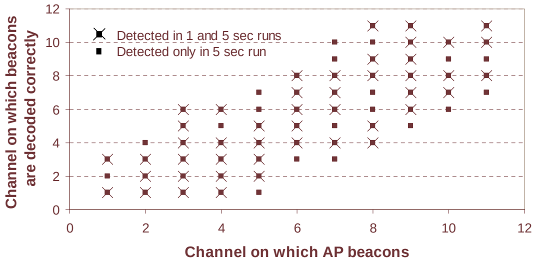


Figure 12: Overlapping channels on which an AP is overheard

In the 40-40-near case, EDEN estimates approximately equal delays for the client and the AP. With an increase in the distance (the 40-40-far and 0-0-far cases), the medium delays increase and EDEN is able to estimate this change as well. Note that the client and the AP delays increased in the the latter two cases by a few milseconds beacuse the wireless cards transmitted the packets at a lower transmission rate (1 Mbps) in order to decrease the error rate. These results show that EDEN is an effective mechanism for obtaining a delay breakdown in a wireless setting.

7.5 Rogue AP Detection

In this section, we explore two issues related to Rogue AP detection. Section 7.5.1 shows that overlapping channels helps in quicker detection of Rogue APs that are hiding on channels where no AP or client is present. Section 7.5.2 shows that even if Rogue APs are not overheard on overlapping channels, there is ample opportunity for clients to perform active scanning without hurting their performance. To check the effectiveness of our implementation, we ran our Rogue AP detection mechanism on our building floor and were able to detect all "known" Rogue APs (these were experimental APs being used by our colleagues).

7.5.1 Overlapping Channels

It is known that overlapping channels in IEEE 802.11 not only interfere with one other but it is sometimes possible for a NIC on one channel to decode packets from another overlapping channel. This characteristic is helpful in detecting Rogue APs: if a client is present on a channel that overlaps with a Rogue AP's channel, it will detect the AP's presence if it is able to hear the AP's beacons.

To verify the extent of this overlap, we performed an experiment in which an AP was placed on channel 1 and a nearby client checked for the AP's beacons on all 11 channels. We repeated this experiment by placing the AP in all channels from 2 to 11 and document where it could be heard. In one run, the client lingered on each channel for 1 second and in the second run, it stayed for 5 seconds. Figure 12 plots the channels on which the AP is heard (Y-axis) when it is placed on a specific channel (X-axis). Clearly, the overlap across various channels is non-negligible and is helpful for detection of Rogue APs. Furthermore, given sufficient time (see the 5-second run), there is an even higher likelihood that some packet from a Rogue AP leaks through to a monitoring DC.

In the above experiments, the AP and the client were placed 5 feet apart with one obstacle between them. We wanted to study the change in leakage across overlapping channels on increasing the the distance between the AP and the client. For this we placed an AP machine at 10 different locations on our floor in various rooms and repeated the above experiment. Figure 13 shows that as the distance between the AP and the monitoring client increases, the AP is heard on fewer channels (the decrease is not monotonic due to obstructions).

The above results show that even though one cannot rely on overlap as a guaranteed mechanism for detecting Rogue APs, it does reduce the need of performing frequent active scans. This observation also implies that there are more opportunities for detecting Rogue APs: for a Rogue AP to go undetected, it must be far away from *any* client that is on an overlapping channel.

7.5.2 *Availability of Idle Times for Active Scans*

As shown in Section 7.2, active scans can take up to 2 seconds. Our current implementation performs an active scan every 5 minutes; we refer to this period as the *Active Scan Period*. Even though 2 seconds out of 300 seconds is a small fraction of the time, it is important for clients to perform these scans at appropriate times; otherwise, network traffic on a client may get disrupted: packets sent to this client may be dropped, TCP may timeout, etc.

Ideally, these scans should be done when the node is idle and has no ongoing network transfers. To determine whether such idle times exist in current usage, we used Ethereal [17] to obtain traces from 3 desktop machines of our colleagues over multiple days. Note that even though these traces are from desktops attached to wired networks, they still give us a reasonable estimate of network traffic generated by users; as users start using laptops as their primary machines, it is likely that the network and idle time behavior will be similar to that of desktop clients.

We divided the traces into 5-minute periods (the Active Scan Period) and for each period, we determined the maximum period of time for which the network was idle. Figure 14 presents the maximum idle period in every 5-minute interval during a 24-hour period. Each point in the graph (e.g., for 12:00 pm to 12:05 pm) is obtained by averaging the maximum idle time value across multiple days and multiple machines for the same 5-minute period. The figure shows that there are large chunks of idle periods available for performing active scans: the smallest idle period available in a 5-minute interval was 118 seconds and typically, idle periods of more than 2.5 to 3 minutes were easily available. Thus, a large window of opportunity is available to our rogue detection scheme for performing active scans every 5 minutes.

Given the availability of such opportunities, one can use any heuristic to predict idle times for launching an active scan (which takes 2 seconds). We studied the effectiveness of a simple history-based heuristic: if the network has been idle for X seconds, it predicts that the network will be idle for the next 2 seonds. Thus, after every 5 minutes, the Rogue AP detection module can perform an active scan whenever it observes that the network interface has been idle for X seconds. We evaluated the effectiveness of this heuristic over our 3-machine traces with two different values of X: 5 and 10 seconds. With both values of X, we observed that the active scan would complete within the idle period for more than 95% of the cases. The effectiveness of this simple heuristic demonstrates that active scans can be easily performed for Rogue AP detection by wireless clients without hurting their performance.

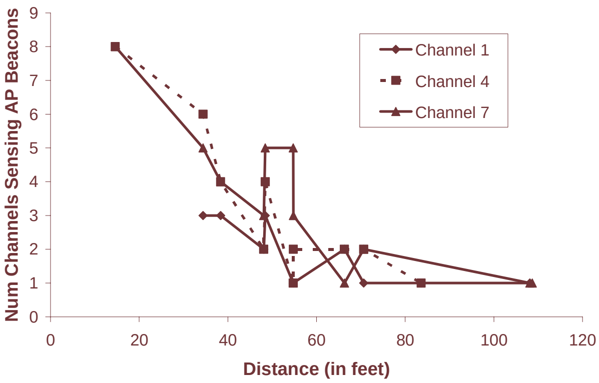


Figure 13: Overlapping channels heard relative to distance

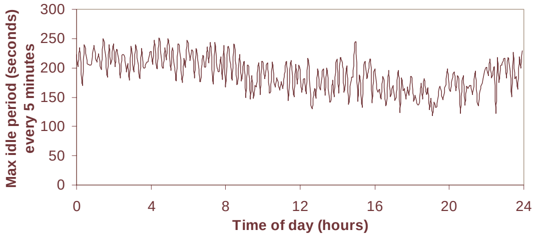


Figure 14: The maximum idle time duration available during every 5-minute period at different times of the day

7.6 Scalability Analysis

As discussed in Section 3.3, our architecture is designed to scale with the number of access points and clients in the system. We now discuss why our proactive and reactive techniques maintain the scalability property. We also argue why our reactive mechanisms impose low network overhead even if a number of clients are experiencing wireless problems in an area.

As discussed in Section 6, each DC pro-actively sends the RSSI, SSID, and MAC address information about nearby devices to the DAP 30 seconds; this information is necessary for Rogue AP detection. The DAP filters this data and sends information about APs every 30 seconds. To understand the network bandwidth consumed on the wireless link, we set up an experiment with a single DC, DAP and DS for 4 hours. We observed that the bandwidth consumption by the DC was less than 0.2 Kbps and the DAP's bandwidth requirements were less than 0.01 Kbps. This result implies that even if a large number of clients were present, the bandwidth usage is still low, e.g., 20 Kbps for 100 clients by DC. Thus, for pro-active monitoring, our techniques have negligible bandwidth requirements.

We now analyze the bandwidth overheads of our reactive diagnosis mechanisms, i.e., Client Conduit and EDEN; we do not discuss DIAL's overheads since DIAL's beaconing messages are part of Client Conduit and the overheads of sending the RSSI information to the DAP has already been discussed above.

The bandwidth requirements of EDEN and the Connection Setup part (beacons and probe messages) of Client Conduit are low since these protocols send small broadcast or beacon packets at a low frequency, e.g., every 100 msecs in Client Conduit and every 2 seconds in EDEN. The bandwidth consumption while using MultiNet can also be controlled: as stated in Section 4.2, the connected client can limit the amount of bandwidth that it allocates to the disconnected client. Thus, if a single client needs help, our reactive mechanisms impose little overhead.

We now analyze the overheads when a large number of clients (say 50) in an area have wireless faults and are utilizing our reactive mechanisms to diagnose their problems. Our basic idea for ensuring that the performance of the network does not deteoriate is to rate-limit our mechanisms; we have not implemented these protocol extensions in our current prototype. In Client Conduit, when a disconnected client overhears the beacons on N disconnected clients, instead of choosing a fixed beacon period of 100 msec, it sends out a beacon every K msecs where K is a random number between 0 and 100*N msecs. This self-regulation ensures that the network is not swamped out by Client Conduit beacons if a sudden loss of coverage occurs in an area.

A similar self-regulatory mechanism is used to limit the rate at which the initial broadcast packets are sent in EDEN. Furthermore, to limit the overheads on a connected client C (and possibly reduce the reactive scheme's load on the DAP and DS), we can use a policy such that C helps only one client at any given point. Thus, with these policy decisions, we can ensure that Client Conduit and EDEN impose low bandwidth overheads even when a large number of clients are experiencing problems.

8. RELATED WORK

To the best of our knowledge, there has been no previous research on fault diagnosis in IEEE 802.11 infrastructure networks. However, there are a number of commercial products that provide varying degrees of support for network management tasks, e.g., AirWave [4], Network Systems and Management (NSM) [16], Wireless Security Advisor [20], AirDefense [2], SpectraMon/SpectraGuard [37], AirMagnet [3], and Symbol [35]. Due to their propriety nature, the available description typically describes the feature-set and not the techniques; the comparison below is based on our understanding of their brochures.

The emphasis in most of these products is more towards managing wireless networks rather than diagnosing faults. These tools allow network administrators to obtain and visualize data from access points, upgrade firmware, manage security policies, etc. Some of them also provide real-time WLAN performance monitoring through IEEE 802.11 statistics such as packet throughput, number of retries, number of dropped packets at the AP, etc. Even though these low-level statistics are useful for network administrators, it is more desirable to provide higher level fault detection and diagnosis, e.g., our approach detects network performance problems and pinpoints the components that are problematic.

Many of these products (e.g., AirWave, Unicenter) operate from the AP or the server side only, i.e., clients are not instrumented. Given the asymmetry and variability of the wireless medium, observing data from the client-side is important for fault diagnosis, e.g., since conditions such as interference near the client can be drastically different than the conditions near the AP, client-side information is needed to do a detailed performance breakdown. Furthermore, our approach of modifying clients allows us to help disconnected clients via Client Conduit, locate Rogue APs and disconnected clients, and obtain better coverage for detecting Rogue APs.

Some products like AirMagnet and AirDefense obtain the complete view of the enterprise by deploying specialized sensors throughout the organization; these sensors pass all the packets to the server for analysis. Anecdotal evidence from talking to various network administrators suggests that products that use sensor-based monitoring are expensive to deploy; furthermore, their performance degrades significantly even when very few sensors are deployed due to the network traffic. Our approach uses regular wireless clients to avoid extra hardware deployment costs. Of course, a limitation of our approach is that we rely on the presence of nearby clients for diagnosing some of the wireless faults; however, the increasing usage of wireless clients in organizations is making it easier to satisfy this requirement.

Since Rogue APs are a serious security problem, all the products listed above perform Rogue AP detection. Unlike our solution, most of these products achieve this goal either by using other APs [4, 16] or by using specialized sensors [2, 3, 37]; as discussed above, these approaches have deployment and fault-detection limitations. Our technique of using both clients and APs for detecting Rogue APs is similar to the Symbol technique [35]. However, unlike their approach, our technique can also detect Rogue APs that use MAC address spoofing of real APs; furthermore, we leverage our client and AP instrumentation to approximately locate Rogue APs using DIAL.

None of the above products provide solutions for assisting disconnected clients even though they need the most help. Our Client Conduit mechanism allows live and reactive diagnosis to be performed for such clients that are unable to access the infrastructure wireless network.

The notion of making wireless clients snoop the environment for ensuring secure and correct routing has been suggested for ad hoc networks. In [25], the authors propose a *watchdog* mechanism to detect network unreliability problems stemming from selfish nodes. The basic idea is to have watchdog nodes observe their neighbors and determine if they are forwarding traffic as expected; this approach for detecting routing anomalies has been further refined by others as well [7, 13]. Inspired by the watchdog mechanism, we also use nearby clients to monitor the RF conditions and traffic flow around them; in our architecture, the watchdog mechanism is used for fault detection (e.g., Rogue APs) and fault diagnosis (e.g., Client Conduit, locating disconnected clients, performance isolation). Recent work [31] has used snooping wireless clients for detecting greedy and malicious behavior in hotspots environment; these techniques are orthogonal to our work and can be incorporated in our framework as well.

Researchers have developed techniques for diagnosing performance problems over the Internet. For example, Barford et al. [9] use traffic traces at the end points and classify delays as occurring due to a slow server, a slow client, or the network. While EDEN has similar goals over a wireless network, it does so without requiring tracing support from both end points. Tulip [24] is another approach for diagnosing delays over Internet paths. The client sends ICMP packets and uses their responses from different components to determine the cause, such as lost packets, packets reordering, or queueing delay. EDEN also uses ICMP packets. However, the broadcast nature of the wireless medium enables EDEN to use a novel approach of snooping these packets as a mechanism for diagnosing component delays.

9. FUTURE WORK

There are a number of additional problems in wireless fault diagnosis that require further research. We plan to pursue these in the near future.

- We presented a technique for detecting Rogue APs in a deployment. A related problem is to detect *Rogue Ad-hoc Networks*. Such networks are created when a user connected to the corporate network (e.g., via a wired network) sets up an IEEE 802.11 ad-hoc network with an unauthenticated client. Thus, like the Rogue AP scenario, such a network can compromise the security of the corporate network.

- The problem of performing root-cause analysis on client authentication problems was not discussed in this paper. For example, the system could analyze the IEEE 802.1x protocol messages to determine the point at which authentication failed.

- In Section 5.1, we show how the location of disconnected clients can be determined when a few connected clients are present nearby. The question remains, what should be done when there are no connected clients in the neighborhood. One approach may be to have the client log its last known location where connectivity was available. Using heuristics, such as movement trajectory, it might be possible to determine the approximate location of the dead spot.

- The next logical step after diagnosis is recovery. Once a fault has been detected, one needs to determine what automatic steps should the system take to resolve the situation without necessarily involving a network administrator.

10. CONCLUSION

The rising popularity of IEEE 802.11 networks has made fault detection and diagnosis an important problem for IT managers responsible for maintaining these networks. Interestingly, the wireless research community has overlooked these problems, perhaps because maintenance issues surface only after large deployments are in place, which is a relatively recent phenomenon.

In this paper, we presented novel solutions for detecting a variety of faults and proposed approaches for analyzing performance problems experienced by end-users. Our initial results show that our mechanisms of locating RF holes, detecting Rogue APs, and diagnosing performance problems are effective and impose low overheads. Furthermore, we show that a novel mechanism called Client Conduit can be used for assisting disconnected clients in real-time. These techniques in conjunction with our general architecture that uses clients, APs, and backend servers together for diagnosing wireless networks make our system unique and practical.

The general problem space of effective network management for IEEE 802.11 networks is large. This paper is a first attempt at addressing some of the critical problems identified to us by network administrators managing a large 802.11 deployment. It is our hope that this paper will stimulate other researchers to investigate such problems further and propose solutions that will eventually result in the smooth operation of IEEE 802.11 networks.

Acknowledgments

We are grateful to Graham Breeze and Don Berry for informing us about various customer and administrator pain-points. We appreciate Alec Wolman's comments for helping us improve the final version of this paper. He also provided us with packet traces of desktop machines; Smitha Sarangarajan provided us with SNMP data from various access points in our building. We would like to thank Abhi Abhishek, Hui Shen, and Jiandong Ruan for their help on the IM driver, Adeel Siddiqui for helping us fix bugs in the miniport driver, and Taroon Mandhana for his help on the RPC service. We are also grateful to Pradeep Bahl, Jitendra Padhye, Venkat Padmanabhan, Siamak Poursabahian, Dan Simon, Dave Thaler, and the anonymous referees for their comments.

11. REFERENCES

[1] B. Aboba and D. Simon. PPP EAP TLS Authentication Protocol. In *RFC 2716*, October 1999.

[2] AirDefense. Wireless LAN Security. http://airdefense.net.

[3] AirMagnet. AirMagnet Distributed System. http://airmagnet.com.

[4] AirWave. AirWave Management Platform. http://airwave.com.

[5] M. Allman, W. Eddy, and S. Ostermann. Estimating Loss Rates With TCP. In *ACM Perf. Evaluation Review 31(3)*, Dec 2003.

[6] Atheros Communications. http://www.atheros.com.

[7] B. Awerbuch, D. Holmer, and H. Rubens. Provably Secure Competitive Routing against Proactive Byzantine Adversaries via Reinforcement Learning. In *JHU Tech Report Version 1*, May 2003.

[8] P. Bahl and V. N. Padmanabhan. RADAR: An Inbuilding RF-based User Location and Tracking System. In *Proc. of IEEE INFOCOM*, Tel-Aviv, Israel, March 2000.

[9] P. Barford and M. Crovella. Critical Path Analysis of TCP Transactions. In *Proc. of ACM SIGCOMM*, Stockholm, Sweden, Aug 2000.

[10] J. Bellardo and S. Savage. Measuring Packet Reordering. In *Proc. of ACM Internet Measurement Workshop*, Marseille France, Nov 2002.

[11] D. Berry and G. Breeze. Microsoft IT division. Private Communication, 2004.

[12] Bluetooth SIG. Location Working Group. http://bluetooth.org.

[13] S. Buchegger and J. Le Boudec. The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-Hoc Networks. In *Proc. of WiOpt*, France, March 2003.

[14] R. Chandra, P. Bahl, and P. Bahl. MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card. In *Proc. of IEEE INFOCOM*, Hong Kong, Mar 2004.

[15] Cisco. CiscoWorks Wireless LAN Solution Engine. http://cisco.com.

[16] Computer Associates. Unicenter Solutions: Enabling a Successful Wireless Enterprise. http://www.ca.com.

[17] Ethereal: A Network Protocol Analyzer. http://ethereal.com.

[18] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-Based Congestion Control for Unicast Applications. In *Proc. of ACM SIGCOMM*, Stockholm, Sweden, Aug 2000.

[19] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics Principles and Practice (2nd Edition)*. Addison Wesley, 1990.

[20] IBM Research. Wireless Security Auditor (WSA). http://www.research.ibm.com/gsal/wsa.

[21] IEEE Computer Society. IEEE 802.1x-2001 IEEE Standards for Local and Metropolitan Area Networks: Port-Based Network Access Control, 1999.

[22] IEEE Computer Society. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Standard 802.11*, 1999.

[23] A. Ladd, K. Bekris, A. Rudys, G. Marceau, L. Kavraki, and D. Wallach. Robotics-Based Location Sensing using Wireless Ethernet. In *Proc. of ACM MobiCom*, Atlanta, GA, Sept 2002.

[24] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. User-level Internet Path Diagnosis. In *Proc. of ACM SOSP*, Bolton Landing, NY, October 2003.

[25] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *Proc. of ACM MobiCom*, Boston, MA, August 2000.

[26] Microsoft Corp. Native 802.11 Framework for IEEE 802.11 Networks. http://microsoft.com.

[27] B. Neuman and T. Tso. An Authentication Service for Computer Networks. In *IEEE Communications*, Karlsruhe, Germany, Sept 1996.

[28] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: a Simple Model and its Empirical Validation. In *Proc. of ACM SIGCOMM*, Vancouver, BC, September 1998.

[29] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *Proc. of ACM MobiCom*, Boston, MA, August 2000.

[30] L. Qiu, P. Bahl, A. Rao, and L. Zhou. Fault Detection, Isolation, and Diagnosis in Multihop Wireless Networks. Technical Report MSR-TR-2004-11, Microsoft Research, Redmond, WA, Dec 2003.

[31] M. Raya, J. P. Hubaux, and I. Aad. DOMINO: A System to Detect Greedy Behavior in IEEE 802.11 Hotspots. In *Proc. of MobiSys*, Boston, MA, June 2004.

[32] C. Rigney, A. Rubens, W. Simpson, and S. Willens. Remote Authentication Dial In User Service (RADIUS). In *RFC 2138, IETF*, April 1997.

[33] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. In *RFC 1889, IETF*, Jan. 1996.

[34] R. Stevens. *TCP/IP Illustrated (Vol. 1): The Protocols*. Addison Wesley, 1994.

[35] Symbol Technolgies Inc. SpectrumSoft: Wireless Network Management System. http://www.symbol.com.

[36] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1), January 1992.

[37] Wibhu Technologies Inc. SpectraMon. http://www.wibhu.com.

[38] WildPackets Inc. Airopeek Wireless LAN Analyzer. http://www.wildpackets.com.

[39] WinDump: tcpdump for Windows. http://windump.polito.it.

[40] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the Characteristics and Origins of Internet Flow Rates. In *Proc. of ACM SIGCOMM*, Pitsburgh, PA, August 2002.

[41] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the Constancy of Internet Path Properties. In *Proc. of ACM Internet Measurement Workshop*, San Francisco, CA, Nov 2001.