Automating Cross-Layer Diagnosis of Enterprise Wireless Networks

Yu-Chung Cheng, Mikhail Afanasyev, Péter Benkö[†], Patrick Verkaik,
Jennifer Chiang, Alex C. Snoeren, Stefan Savage, and Geoffrey M. Voelker

Department of Computer Science and Engineering
University of California, San Diego

ABSTRACT

Modern enterprise networks are of sufficient complexity that even *simple* faults can be difficult to diagnose — let alone transient outages or service degradations. Nowhere is this problem more apparent than in the 802.11-based wireless access networks now ubiquitous in the enterprise. In addition to the myriad complexities of the wired network, wireless networks face the additional challenges of shared spectrum, user mobility and authentication management. Not surprisingly, few organizations have the expertise, data or tools to decompose the underlying problems and interactions responsible for transient outages or performance degradations. In this paper, we present a set of analysis techniques and models to precisely determine all sources of data transfer delay due to media access and mobility in 802.11 networks — from the physical layer to the transport layer — as well as the interactions among them. While some sources of delay can be directly measured, many of the delay components, such as AP queuing, backoffs, contention, etc., must be inferred. To infer these delays from measurements, we develop a detailed model of MAC protocol behavior, both as it is described in the 802.11 specification as well as how it is implemented in vendor hardware. Combined with comprehensive traces of wireless activity taken from an enterprise network, we produce a complete delay breakdown for packet transmissions and pinpoint problems that constrain connectivity or limit performance.

1. Introduction

"Is your wireless working?" The familiarity of this refrain underscores both our increasing dependence on ubiquitous Internet connectivity and the practical challenges in delivering on this promise. The combination of unlicensed spectrum and cheap 802.11 silicon have driven a massive deployment of wireless access capability – which started in the home and was soon followed by the workplace. Today over two-thirds of U.S. corporations provide WiFi-based untethered Internet connectivity [9]. However, there is a significant difference between installing a single wireless access point (AP) in an isolated home – effectively a simple range extender for a wired Ethernet interface – and wireless deployment throughout an enterprise. The latter may comprise hundreds of distinct APs, carefully sited and configured in accordance with an RF (radio frequency) site survey and ideally managed to minimize contention, maximize throughput and to provide the illusion of seamless coverage. Moreover, this intricate machinery is not managed by the 802.11 protocol family itself – which, in all fairness, was never designed for the level of success is has experienced. Instead, the burden falls to the network administrator who must manage the interactions between the RF domain, link-layer dynamics, dynamic addressing and address binding, VLAN setup, as well as the myriad complexities of the wired network itself.

Given this complexity, it is not surprising that even *simple* faults can be difficult to diagnose – let alone transient outages or service degradations. Thus, when a network manager is asked, "Why was the network flaky 10 minutes ago?" the answer is inevitably, "I'm not sure. It looks fine now." While this problem is not unique to 802.11-based networks, these environments impose additional complexities that are unique and qualitatively harder to diagnose.

Among these issues, wireless networks interact via shared spectrum in ways that may not be observable directly (contention and interference) and yet can produce significant end-to-end delays or packet losses. Further complicating such analysis, the 802.11 standards allow considerable "latitude" in the media access protocol and consequently vendors have produced a wide range of "interpretations" – many of which have significant impact on performance. Finally, the promise of seamless coverage is not a property provided by 802.11 itself. Instead, most enterprise deployments implement this property using an undeclared "layer 2.5" patched together from portions of the 802.11 protocol, combined with VLANs, ARP, DHCP and often proprietary mobility management and authentication systems. Unsurprisingly, this Rube-Goldberg contraption has its own unique failure modes.

In truth, even it is technically feasible to diagnose such systems, this process is highly labor intensive and only cost effective for the most severe problems. Even then, the range of interactions and lack of visibility into their causes may stymie manual diagnosis. In one recent episode in our organization, wireless users in a new office building experienced transient, but debilitating, performance problems lasting over a year, despite extensive troubleshooting by local experts and vendor technicians.[1]

We believe that such diagnosis must be automated, and that networks must eventually address transient failure without human involvement. Network administrators have neither the range of domain expertise, nor the visibility into underlying processes, nor the data processing ability to understand why their networks are unreliable. As a first step in this direction, we have built an on-line

[†]Benkö is a visiting researcher at UCSD from the Traffic Analysis and Network Performance Laboratory (TrafficLab) at Ericsson Research, Budapest, Hungary.

[1]We believe we have diagnosed their problem using our system – a subtle bug in the AP vendor's implementation – but it is easy to understand in retrospect why its discovery was challenging to find through trial and error.

analysis system that processes raw wireless trace data and can accurately model the impact of protocol behavior from the physical layer to the transport layer. In particular, we demonstrate techniques to infer the causes of link-layer delays and the effect of mobility management protocols. Using our system we investigate the causes of transient performance problems from traces of wireless traffic in a four-story office building. We find that no one anomaly, failure or interaction is singularly responsible for these issues and that a holistic analysis may in fact be necessary to cover the range of problems experienced in real networks.

The remainder of this paper is structured as follows. We first review the literature we build upon and related efforts in Section 2. We then outline the many potential sources of service disruption – the gauntlet faced by each 802.11 packet – in Section 3. Sections 4 and 5 describe our techniques for modeling media access and mobility management behavior respectively, including an analysis of the problems identified at our location, followed by our conclusions.

2. Related Work

Ever since wireless local-area networks such as 802.11 have been deployed, researchers have sought to understand how these systems behave and perform, based on empirical observations. The monitoring systems used to make these observations have increasingly expanded both in complexity and scope over time. Early systems used existing infrastructure, such as the wired distribution network and the APs, to record wireless traffic and network characteristics [3, 16]. Later systems deployed small numbers of dedicated monitoring nodes, sometimes concentrated near the APs, other times distributed throughout the network, thereby pushing the frontier of observation into the link-layer domain [13, 18, 22]. Recent efforts have substantially scaled monitoring platforms to observe large, densely deployed networks in their entirety [1, 7], providing the ability to observe every link-layer network transmission across location, frequency, and time [7].

These monitoring systems have been used to directly measure a number of interesting aspects of 802.11 behavior and performance, ranging from application workloads and user mobility at the high level to wireless loss, rate adaptation, and handoff delay at the link layer [12, 14, 19] and even physical layer anamolies [21].

Not all interesting wireless performance and behavior, however, can be directly measured. Thus, researchers have developed various techniques to *infer* network characteristics. Perhaps the most common wireless network characteristic inferred is user location [2, 5, 8, 11]. Numerous techniques infer user location based upon AP association, received signal strength, etc.

Other techniques infer more detailed network events and characteristics, such as link-layer loss and the transmitters of packets lacking MAC addresses [5, 7, 18], co-channel interference and overprotective APs [7], misbehaving and heterogeneous devices [7, 10, 5], root causes of physical-layer anomalies [21], and regions of poor coverage [5]. We greatly expand upon these detailed efforts and present techniques to infer critical path delays [4] of media access for every packet, such as AP queuing delay and media contention (mandatory and regular backoff), as well as techniques that infer management delays for supporting intermittent and mobile devices for every user.

To infer critical path delays for wireless transmissions, we develop a detailed model of 802.11 media access (Section 4). Numerous models have been developed previously to estimate various aspects of 802.11 performance, such as the overall network capacity as governed by the 802.11 protocol [6], the maximum throughput of a flow in an 802.11 network [15], and the saturation throughput and expected access delay of contending nodes [17]. Such models are typically analytic. To make analysis tractable, they explicitly make simplifying assumptions such as absence of transmission errors, uniform transmission rates and packet sizes, infinite node demand and steady contention for media access, uniformly random probability of collisions and interference, etc. As a result, these models may be useful for understanding the limits of 802.11 performance under idealized conditions, but omit analysis of important aspects of real networks that we infer with our model: the magnifying effects of bursty traffic that averages and expected values conceal; the complexities of workload, protocol, and environment that lead to correlated and unexpected interactions.

Three recent systems are closely related to the goals of this paper. Jigsaw uses monitoring nodes distributed throughout a university campus building to capture every wireless event in the building across location, channels, and time [7]. Jigsaw combines and synchronizes traces from all radios into a single, unified trace, but in its original form provides little analysis for diagnosing problems. We have downloaded the Jigsaw software for use in our environment, and extended it with our models and analyses for diagnosis. DAIR uses wireless USB dongles attached to desktop machines in an enterprise wireless network to measure wireless events throughout the network [1]. DAIR applications install filters at the wireless monitors to trace information of interest and store it in an central database; applications (inference engines) then use this data to perform analyses. Very recent work on DAIR develops management applications that take advantage of client location, such as identifying regions in the network experiencing consistently poor coverage [5]. Our goals are similar in that we develop analyses to aid network management, but our approach is to base analysis on a global understanding of network behavior across all protocol layers. The MOJO system develops tools and techniques to identify the root causes of physical-layer performance anomalies, such as broadband interference and the capture effect [21]. While we are interested in the effects of physical-layer issues, we identify them as just one cause among many across the interacting protocol layers.

3. Problems in the life of a packet

There are numerous opportunities for disrupting or degrading a user's connectivity in an 802.11 network. To illustrate these challenges and motivate the analyses we perform subsequently, this section provides a quick primer on the many and varied sources of delay and packet loss.

3.0.0.1 Physical Layer.

The physical layer presents the first obstacle for an 802.11 frame hoping to be delivered. Sharing the 2.4Ghz spectrum are a wide range of non-802.11 devices, ranging from cordless phones to microwave ovens. An 802.11 packet in flight may be corrupted by such signals or it may simply by overpowered at the receiver. Alternatively, the sender may detect the presence of RF energy on the channel and defer transmission — incurring delays until the interfering source is silent.

For example, Figure 1 illustrates the impact of using a microwave oven. The figure depicts the reception of physical error frames over time (the y-axis depicts time in microseconds while the x-axis is the time). The characteristic pattern results from the wave doubler circuit used in consumer microwave ovens to convert AC line power into microwave energy. In this design, a capacitor is alternately charged and discharged according to the phase of the AC line voltage — only generating output during the negative phase of the input sine wave. Roughly speaking a U.S. oven will generate

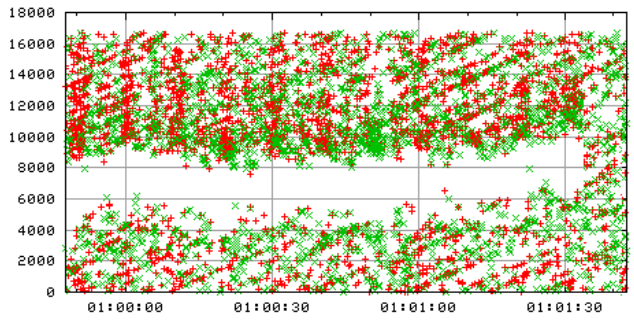

Figure 1: Physical error frame pattern during microwave oven use.

swept broadband interference for 8ms (half of the 60Hz cycle) followed by a similar period of quiescence. This pattern is relatively straightforward to detect although there can be considerable differences between manufacturers. In all cases, in-range 802.11 radios will defer transmission until the medium is idle, building queues and adding delay in the process. Frames in flight when the oven is turned on may be corrupted, depending on the receiver power of the microwave signal.

Such physical layer interactions are not restricted to non-802.11 devices. The 2.4Ghz ISM band combined with the nominal 22Mhz channel bandwidth used by an 802.11 transmitter can easily overlap neighboring transmitters on different channels. Indeed, while it is received wisdom that 802.11 has three "non-overlapping" channels, this statement is false in practice. While the spectral mask for 802.11b stipulates a power drop of 50dB at 22Mhz off the center frequency, this is insufficient to prevent interference into adjacent "non-overlapping" channels. For example, the Atheros AR5004 802.11 chipset offers an 802.11b power output of +18dBm and a receiver sensitivity of -90dB at 11Mbps (-95dB at 1Mbps). Thus, while a transmitted signal on 802.11 channel 1 will fall off to -32dBm, it easily received by a radio tuned to 802.11 channel 6. This situation is much worse for 802.11g since its OFDM modulation scheme produces an even looser spectral mask — losing only 30dB at 22Mhz off center. In practice, we observed such adjacent channel interactions routinely and we have witnessed many successful packet receptions between radios in which the transmitting and receiving radios were separated by as much as 50 Mhz (i.e., channel 1 to channel 11).

3.0.0.2 Link Layer.

The 802.11 link-layer presents another potential performance landmine for user packets. In particular each 802.11 access point manages two critical functions: media access and bindings between individual stations (clients) and APs. Each of these functions can induce additional and, at times, unnecessary delays. We consider each in turn.

Transmission delays. Sources of link-layer transmission delay include queuing at the AP, protocol delays such as mandatory backoff on transmission, exponential backoff on loss, packet transmission time (a function of the encoded frame rate and the packet size), and contention in the network when users and APs overlap and share a contention domain (or due to interference as mentioned above). A single packet may be delayed by all of these factors and, due to retransmission, it may be impacted multiple times. Moreover, it is common for 802.11 drivers to code data at a slow rate after a loss, although this may have unintended negative effects since it takes longer to transmit and excerbates channel utilization.

For example, consider a packet received by an AP at time t. It may be delayed in a queue waiting for previous packets to be transmitted (each experiencing their own media access delays and retransmission overheads). When it reaches the head of the queue it must perform a mandatory backoff, waiting between 0 and 15 slot times (a normal 802.11b slot is 20μs, although 802.11g permits the use of a "short" 9μs slot time under certain circumstances). After this it must sample the channel for the duration of a "DIFS" interval (50 μs) before sending. If the AP detects a busy channel, it will perform yet another backoff before commencing the transmission. Finally, the packet is transmitted with a delay largely determined by the sender's choice of rate. However, if the sender does not receive the acknowledgement from the receiver, the sender would perform another backffoff before retransmission. Of course, even this explanation is over simplified and any real analysis must also deal with delays from interacting protocol features like power management and vendor irregularities (some vendors allow certain packets to be prioritized in between retransmissions of a frame exchange). Unfortunately, most of the delay components at this level cannot be observed directly since they depend on the internal state of the AP which is not exposed via any protocol feature.

Management delays Another important source of overhead in wireless networks broadly falls into the category of wireless management. 802.11 clients and APs are in a constant dance trying to determine the best pairing. To address the issues of mobility, clients continually scan their environment looking for a better partner. APs respond to these scans, and additionally broadcast beacons to nearby clients. If a client switches APs another set of exchanges takes place that authenticates the client to the network and binds the two (a process called association).

Additionally, APs deal with significant heterogeneity in their client base, which includes distinct capabilities and configurations. Consequently, a negotiation takes place between clients and APs about which features are needed — 802.11b transmission, power savings, etc. Unintuitively, the choice of a single notebook computer to associate with an AP can transform that APs behavior as it tries to accommodate the lowest common denominator among its clients. For example, Cheng et al. report that a single 802.11b client — even one that is not transmitting — will force an AP into 802.11g "protection" mode, thereby degrading service for all 802.11g users. [7]

3.0.0.3 Infrastructure support.

APs are fundamentally bridge devices. In order to obtain Internet connectivity a client must in turn acquire an IP address — typically via DHCP — and the MAC addresses of next-hops to destinations — typically via ARP. These protocols exhibit complex dynamics in themselves, and their failure may isolate a station for some time. Their use with 802.11 exacerbates their complexity since they are used in specialized ways, frequently tied together with VLANs using proprietary mobility management software that authenticates stations via a single-sign on interface and allows IP addresses to remain consistent as a client moves. There is no standard for implementing this functionality and unsurprisingly their failure modes are not well understood.

3.0.0.4 Transport Layer.

Finally, any underlying delays or losses are ultimately delivered to the transport layer, usually TCP, which may amplify their effects believing these behaviors to be indicative of congestion.

While the process described usually works, when it doesn't it can fail spectacularly and expose the user to significant response time delays. It is the goal of this paper to systematize the analysis

of these issues to better understand the source of such transient problems.

4. Media access model

In this section we describe a media access model for measuring and inferring the critical path delays of a monitored frame transmission. The model consists of a representation of the wired distribution network, queuing behavior in the AP, and frame transmission using the 802.11 MAC protocol. The goal of the model is to determine the various delays an actual monitored frame encountered as it traversed through the stages of the wireless network path. At a high level, the model first determines a series of timestamps of a frame as it traversed this path and was finally transmitted by the AP. From these timestamps we can then compute the delays. Table 1 summarizes the definitions of the timestamps and delays in our model, and Figure 2 illustrates where in the network path they correspond.

The model uses measurements of the frame both on the wired network and the wireless network to determine some of these timestamps. The challenge, however, lies in inferring the remaining timestamps and, hence, delays. The inference techniques we develop, along with the representations of AP queuing and the transmission behavior necessary to perform the inferences, represent a key contribution of this paper.

In the following sections we describe in detail our model components and how we measure and infer these timestamps and delays. We then show how the critical path delays determined by the model can provide valuable, detailed insight into the media access behavior of wireless users. Finally, we show how we can use the model to diagnose problems with TCP throughput.

4.1 Critical path timestamps

Our system directly measures the timestamp of the packet as it leaves the router on the wired distribution network, t_w. A SPAN port on the router for the building distribution network forwards all packets to a tracing machine. The router forwards copies of the packet both when it enters and leaves the router, and we define the time of the packet when it leaves as t_w.

To calculate delays we use observations of the packet on the wired network together with observations of the packet on the wireless network. As a result, our system must match the packet from the wired trace with the packet as observed in the wireless traces. It compares the wired packet contents with wireless packets within a one second window by converting frame formats from 802.11 to Ethernet II. The window size is determined by the maximum wireless forwarding delay of a wired packet. Under heavy load the AP can queue a packet close to one second. Most matches are one-to-one, meaning one wired packet corresponds to wireless packet, but there are cases for one-to-many matches. For instance, for broadcast frames like ARP request, one wired frame might match multiple wireless frames because all APs will forward the broadcast frame in the wireless. Further more, broadcast frames like ARP requests or NetBios are often sent in short bursts (2-5 packets) with identical contents within one second. We match these frames by matching the frame ordering in addition to matching the packet payload. Another one-to-many case is the broadcast frames from the client like DHCP discover messages. Such a message is first sent by the client to its AP; it then is forwarded through the wired network to all APs, then all APs forward the packet into the wireless again.

The next step is to determine when the AP has received the frame from its wired interface, t_i. Since we do not have taps on the APs or control the AP software, we cannot directly measure this time and instead must infer it. t_i is not a constant offset from t_w because APs have different propagation latency from the router and AP ethernet I/O delays depend upon packet size. For each AP, we estimate t_i by first measuring the distribution of the interval $(t_s - t_w)$, the difference between the wireless transmission time and the wired timestamp of the packet. The minimum value of this distribution, minus DIFS, should be the sum of wired network delay and AP input processing delay.

From here, we determines the transmit queue timestamps of the packet inside the AP, both when the packet enters the transmit queue (t_q) and when it reaches the head of the queue (t_h). We model the AP as having three FIFO packet queues, the transmit ready queue and two waiting queues. If the packet is broadcast or multicast, the AP schedules it onto the broadcast queue; the AP flushes this queue into the transmit queue after the next beacon transmission. In addition, the Avaya APs always prioritize broadcasts frames once they are scheduled on to the transmission queue, even the AP is in the middle of a frame exchange. We will discuss this effect in detail later in this section. If the packet is destined to a power-saving client, the AP buffers it on a power-save queue. The AP flushes the appropriate packets from the power-save queue into the transmit queue when the client wakes up (by receiving a PSM-reset data or management frame, or a PsPoll frame from the client); for example, Intel clients typically sleep for 50ms and wakeup for 50ms. Otherwise, the AP places the packet directly on the transmit queue.

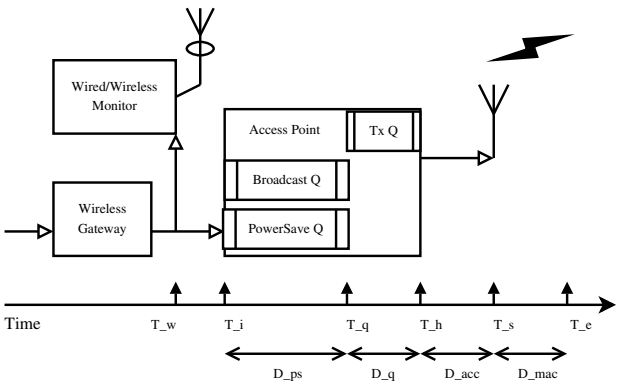

Figure 2: Representation of wired distribution network, queuing behavior in the AP, and frame transmission. The arrows indicate where in the network we measure and infer timestamps as frames traverse the network, and the corresponding delays we calculate.

Timestamp Definitions	
t_i	AP receives frame from wired interface
t_q	Frame enters radio transmit queue
t_h	Frame reaches head of transmit queue
t_s	First bit of the frame transmitted
t_e	End of last ACK or estimated ACK timeout
Delay Definitions	
d_{ps}	t_q - t_i: AP power-save buffering delay
d_q	t_h - t_q: AP (transmission) queuing delay
d_{acc}	t_s - t_h: Media access delay to transmit first bit
d_{mac}	t_e - t_s: MAC delay
d_{dcf}	t_e - t_h: DCF delay (d_{acc} + d_{mac})

Table 1: Summary of timestamps and delays determined by the media access model.

It is critical to model the queuing behavior precisely to estimate further wireless delays. For example, if we did not model packets sent to clients in power-save mode correctly, they would appear to be sent out of order and delayed at the AP for tens of milliseconds. We track whether clients are in power-save mode when packets for them arrive at the AP, by tracking either the PSM bit of client frames in the wireless trace, or when beacons indicate that the AP has buffered packets for clients (TIM). For broadcast, the 802.11 specification says an AP should deliver broadcasts at DTIM intervals if power-saving clients exist because these clients only wake up at that time. The Avaya AP uses the same DTIM interval as its beacon interval (102.4 ms). In addition, the Avaya APs always do this even if they do not have any power-saving clients. This effect could lead to performance problems, which we will discuss later.

Based on the frame destination and client power status, we tag each frame with the appropriate queue type. Subsequently, we can estimate the time when the AP places the frame on the transmission queue, t_q. For a broadcast/multicast frame, t_q is the time of the latest beacon prior to its transmission in the wireless. For frames destined to power-saving clients, t_q is the time the client notifies the AP that it has woken up by sending a frame with the PSM bit off such as a PsPoll control frame. For the remainder of the frames, $t_q = t_i$ because the AP schedules them immediately after it has received them from the wired interface.

Next, we infer the time when the packet reaches the head of the queue, t_h, and the AP is ready to transmit it using 802.11 DCF. We determine t_h under three conditions based upon the end time of the previous transmission attempt, t_{pe}. First, if the AP places the frame on the transmit queue before the previous transmission completes, then the frame experiences head-of-line blocking. We conclude the frame reaches the head of the queue after the previous transmission finishes. Thus $t_h = t_{pe}$, and we label this frame as "head-of-line blocked." According the 802.11 standard, the AP must perform a mandatory backoff at the end of each frame exchange regardless of the transmission result. We can not directly measure this backoff window but we know the maximum of this window from the standard. Therefore, if the frame enters the queue beyond the maximum mandatory backoff window after t_{pe}, the frame must find the transmit queue empty and the AP can transmit immediately. Hence, $t_h = t_q$, and the packet is labeled as "no head-of-line blocking." Finally, if the AP places the frame on the transmit queue during the maximum mandatory backoff window of the previous attempt, the frame may or may not experience head-of-line blocking by the previous frames. Since this backoff window is very small (300 μs), less than 1% of the frames fall into this category. We assume the transmit queue was empty at t_q and the frame does not encounter head of line blocking. Thus $t_h = t_q$ as well.

We determine the starting and ending transmission times of the frame exchange, t_s and t_e, directly from the synchronized trace. The start time t_s is the start of the first transmission attempt, including the control overhead of RTS/CTS and CTS-to-self. The end time t_e is the end of the frame exchange: the end of the ACK of the last transmission attempt, including all retransmissions and contention. For unacked unicast frames, t_e should be the scheduled end time of the transmission (end of the NAV). However, the Avaya AP will start sending the retry 60 μs (50 μs + SIFS) after sending the previous data frame if it has not hear the ACK. This behavior occurs only with 11b data frames (because 11g ACK takes only 28us). Consequently, for unacked unicast frames t_e is end time of the data frame plus 60 μs.

Frames internally generated in the AP represent a special case because we cannot observe when the AP generates the frames. For example, we do not know when the AP has scheduled a scan response because there is no external information to correlate from our wired trace. Fortunately, typically these frames are management responses to client requests, such as scan responses and association/authentication responses. We assume that the AP generates these responses and places them on the transmit queue (t_q) immediately after it receives the requests.

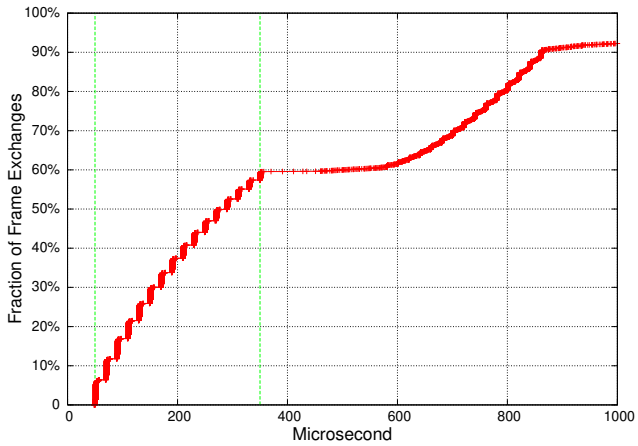

Figure 3: Access delay (d_{acc}) distribution of one hour of frame exchanges from an AP to a client doing a bulk TCP download.

4.2 Critical path delays

Finally, we calculate the critical path delays as intervals between timestamps. In particular, the buffering delay for power-saving clients and broadcast frames is $d_{ps} = t_q - t_i$, the time from when the frame reaches the AP and when the AP places the frame on the transmit queue. We label broadcast buffering delay as power saving delay because they are buffered for power-saving clients who periodically wake up at beacon intervals. The AP transmission queuing delay is $d_q = t_h - t_q$, the time between when the AP places the frame on the queue and the time when it reaches the head of the queue (i.e., the AP is ready to transmit it). After the frame becomes the head of the transmit queue, d_{dcf} is the time the AP takes to perform all DCF operations to transmit the data Thus, $d_{ps} + d_q + d_{dcf}$ is the total time the packet spent in the wireless distribution network.

We further categorize AP queuing delay d_q into three components. When an AP places a frame on the transmit queue, we examine what kinds of frames are already on the queue. But we can not measure the AP queue directly, we exam the AP queue by observing the previous transmitted frames from the AP. Based upon the wireless transmission delays d_{dcf} of each of these frames, we break down AP queuing delay d_q into delay caused by background management frames such as beacons, scan responses, etc. (d_{qb}), unicast frames to the same client (d_{qs}), and unicast frames to other clients (d_{qo}); $d_q = d_{qb} + d_{qs} + d_{qo}$.

d_{dcf} includes the access delay and MAC delay. The access delay is $d_{acc} = t_s - t_h$, the time from the frame reaches the head of the queue till the AP starts to transmit the first bit of the frame. The MAC delay $d_{mac} = t_e - t_s$ is the entire duration from the AP sends first bit of the frame until it finishes trying, regardless of whether the frame exchange succeeds or not. Thus d_{mac} may include multiple retransmission attempts. Moreover, d_{mac} also includes the exponential back offs and channel busy periods in between retransmissions.

Even though we must infer the timing of some of the events that determine critical path delays, our experience has been that our analysis can be accurate even for very fine-grained phenomena.

For instance, Figure 3 shows the cumulative distribution of access delays d_{acc} in microseconds for one hour of frame exchanges from an AP to a client doing a bulk TCP download. During this time, another two clients of this AP are also actively doing TCP downloads. Thus the three clients and the AP contend the channel to send TCP-ACKs or TCP data packets. Non of the clients are in power saving mode during that hour.

Remember the access delay d_{acc} is the time from the frame reaches the head of the transmit queue (t_h) till the AP starts transmitting the frame for the first time as part of a frame exchange (t_s). For clarity, we focus on those frames that experience head-of-line blocking in the transmit queue — in other words, the AP is transmitting frames in succession as frames queue up, a common situation with bulk downloads.

For frames sent in succession, the AP first waits a mandatory backoff delay. The mandatory backoff delay is $m*20$ μs, where the AP randomly chooses the integer slot m between 0–15. After the mandatory backoff, the AP will start a regular DCF operation. First it listens on the channel for the DIFS interval (50 μs). If the channel is idle, the AP transmits the frame right away. In this case, d_{acc} is the mandatory backoff delay plus the DIFS delay. If the channel is busy, the AP further performs a regular backoff sequence. In this case, d_{acc} includes the initial mandatory backoff and DIFS, as well as the regular backoff time plus the contention time due to the busy channel (the backoff timer stalls while the channel is busy); this additional delay will include all regular back-offs performed.

The distribution of access delays shown in Figure 3 reflects the various components that comprise the overall access delay d_{acc}. Every frame will have to wait at least a DIFS interval during the transmit process; hence, the distribution starts at a delay of 50 μs marked by the first vertical line. The "steps" immediately following correspond to the mandatory backoff delay. The frames have a DIFS delay plus the mandatory backoff delay, a random multiple of 20 μs slots from 0–15; each step in the graph corresponds to one of the slots. The second vertical line marks the end of this category of frames (about 60% of the frame transmitted). The next group of frames (through 850 μs) are frames experiencing contention at the end of the DIFS interval, and therefore incur additional contention delay plus a regular backoff — hence the second set of "stairs." The remaining 10% of transmitted frames with the largest delays are frames that experienced long contention delays.

The classification is not exact. For example, a frame that has 200 μs access delay might experience mandatory backoff, busy channel in DIFS probe, and contention during the regular backoff. But the AP picked small backoff slots by chance. The CDF in Figure 3 is an aggregate distribution of the above classes of frames, hence the "steps" do not have same height because the line will be slanted toward right by other classes of access delays. For frames sent without head-of-line blocking (i.e., d_q = 0, not shown in Figure 3), the d_{acc} = DIFS if no contention is detected during the DIFS channel probe interval. Otherwise, d_{acc} = DIFS + contention + regular back-offs.

4.3 Applying the model

The media access model makes it possible to measure the critical path delays for every packet sent from APs to the client. As an example, we focus on a particular AP in the building where three clients (X_b, Y_b, Z_g) are using TCP to download different files from the same Internet server, and the downloads overlap in time. The clients compete with each other for both AP resources and airtime. Two clients use 802.11b (X_b, Y_b) and the third uses 802.11g (Z_g).

We apply the media access model to Y_b's TCP flow to measure the critical path delays for each of the packets sent from the AP to the client. Figure 4 shows the delay breakdown for this client's packets over four minutes. Each spike in the graph corresponds to the combined queuing and wireless transmission delays for transmitting one frame. The wireless transmission delay ("DCF delay") includes both the initial access delay d_{acc} and MAC delays d_{mac}. These delays are quite small (even with contention among three clients) and are shown at the top tip of each spike. We break down the queuing delay into three components: "other" is the delay d_{qo} waiting for frames to other clients to leave the queue; "self" is the delay d_{qs} waiting for frames to this client; and "bg" is the delay d_{qb} waiting for background management frames (beacons, scans, etc.). Overlayed across the spikes is the TCP goodput achieved by the client. Above the spikes we show points in time where a frame was lost during wireless transmission (triangles) and on the Internet (diamonds).

This detailed breakdown shows a number of interesting interactions and behavior. First, queuing delay in the AP is the dominant delay on the wireless path to the client. These delays are orders of magnitude larger than the wireless transmission delay; although this client was using 802.11b rates, using 802.11g rates would not have improved client performance. Second, roughly half of the time client Y_b's frames were queued for its own frames, and the other half was caused by delay encountered by frames for the other two clients. Examining the frame delays of the other clients, most of those other frames were for client X_b and the minority were for Z_g. Third, Y_b experiences occasional wireless loss, but wireless loss does not have a substantial impact on achieved goodput. Fourth, Y_b experiences a burst of Internet loss at 14:39:38, substantially impacting goodput. The AP queue drains as Y_b times out and recovers. Finally, Y_b's download goes through a phase change just after 14:40:00. The other clients finish downloading (the frames in the AP queue are for Y_b) and Y_b no longer has to share the channel. AP queue occupancies drop and goodput increases substantially beyond the level when it was contending with other clients.

4.4 TCP Throughput

Next we describe how we can use the media access model as a basis for diagnosing problems with TCP throughput for wireless users, and show that there can be many causes that can limit TCP throughput. Given a TCP flow using wireless, we first identify whether throughput is the critical bottleneck of a TCP flow. We then examine the flow and determine whether throughput performance appears to be limited by wireless network conditions. If so, we then use the media access model to determine critical path delays for packets in the flow, evaluate how those delays interact with TCP, and assign a root cause for why the TCP flow was limited when using the network.

The first step is to determine whether a TCP flow contained data transfer periods whose throughput could be limited by wireless conditions. Since a given TCP flow may have idle periods (e.g., think times during persistent HTTP connections), we identify periods of time during a TCP flow when it is performing a bulk data transfer. We call such a period a TCP transaction. A TCP transaction period starts when we observe new, unacknowledged TCP data and ends when all outstanding data packets generate an acknowledgment. Most of the packets in this period must also be MSS-sized except for the last data packet, reflecting a period when a bulk of data is being sent. We then calculate the amount of data transferred during the flow to identify flows of sufficient size that they could potentially take full advantage of the wireless channel; we currently use a threshold of 150 KB.

We then take these flows and determine whether throughput performance appears to be limited by wireless network conditions,

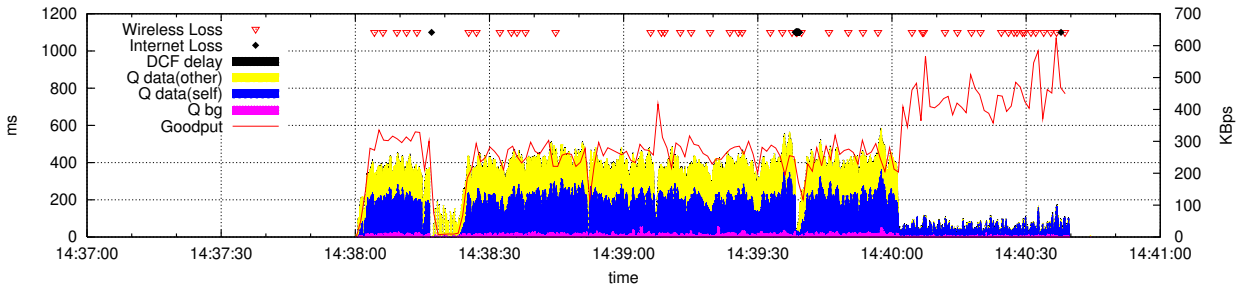

Figure 4: Time-series showing critical path delays, goodput, and losses for frame exchanges from an AP to a client doing a bulk TCP download.

and, if so, why. In our approach, we assume that there is a single root cause and that factors are largely independent (e.g., wireless loss is independent of Internet loss). We then analyze the flow through a series of filters. First, if the flow is achieving near optimal throughput for the rate used, we label it ideal and perform no further analysis. If the flow announces a zero receiver window, we label it as receiver window limited.

We then determine if the Internet part of the connection was the bottleneck. To do this, we estimate what the TCP throughput for the flow *might have been* if using the wireless network under ideal conditions (no wireless loss, no contention, no wireless RTT, etc.). We use Padhye's TCP throughput estimation analysis [20] to perform this estimation, calculating idealized throughput just using the measured Internet RTT, measured Internet loss rate, and an estimated RTO. If the estimated Internet throughput is close to measured, we label the flow as Internet limited.

We then examine wireless losses and add wireless loss rate into the throughput estimation; if throughput drops substantially, we label the flow as wireless loss limited. At this point, remaining flows are usually victims of high wireless RTTs. Using the media access model, if the AP-to-client delay is larger than client-to-AP, we label the flow as being either limited by queuing delays (d_q) (background traffic, frames to self, or frames to other clients), power-save delays (d_{ps}), or DCF transmission (d_{dcf}). If the primary delay is DCF, and the transmission time only takes less than half of the DCF delay, we label it as contention limited. Otherwise, we check whether the client potentially could have used a higher rate; if so, we label it as rate limited. If the client is using an 802.11g rate and the AP is in protection mode, we estimate the potential benefits of removing the CTS-to-self overhead [7]; if this benefit is higher compared with using protection mode at the highest 802.11g rate, we label the flow as protection-mode limited.

What are the sources of wireless behavior that impact TCP throughput in a production network? We apply the above analysis to all measured TCP flows on the building wireless network for 24 hours on a typical weekday. Figure 5 shows the breakdown of root causes across flows identified as bulk data transfer flows.

The graph shows four interesting results. First, flows can be limited by a wide range of different causes; for a particular user experiencing poor TCP throughput, we must model and check all such causes to diagnose their particular problem. Second, over 25% of the flows are limited by the faulty 802.11g link-level retry policy used by the APs in the building. At 802.11g rates, the APs only perform one link-level retry before giving up when the AP is in protection mode; not surprisingly, this policy limits TCP performance when using those rates. Third, over 30% of the flows are limited by the use of 802.11g protection mode. Fourth, nearly 30% of the flows turn out to be limited by the receiver window size — indicating that although wireless conditions may initially be suspect, throughput can be limited simply by the TCP stack configuration. Any diagnosis system must suspect causes outside of wireless as well.

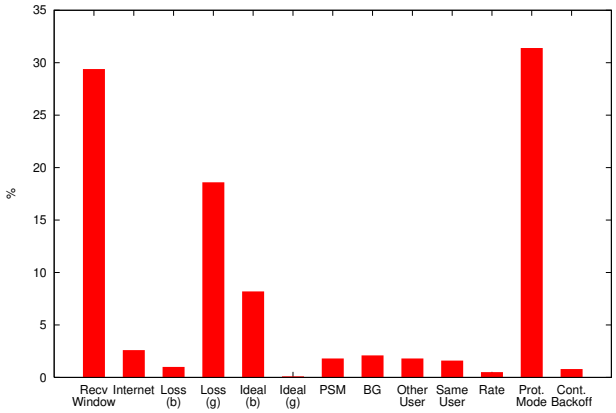

Figure 5: Root wireless causes that limit TCP flow throughput.

5. Mobility

The second class of overhead in the 802.11 environment is the expense of the various types of mobility management, including scanning for access points, association, ARP, DHCP, authentication, etc.

5.1 Overhead analysis

In order to determine how efficiently 802.11 clients are using the network we categorize each packet into one of eight categories: scanning, PSM sleep, association (including Auth, Assoc, ReAuth, ReAssoc), DHCP, DNS, ARP, TCP, and other. In our environment, other includes WEP/WPA (while none of our APs support encryption, clients may occasionally send such packets), IPv6, mDNS, Windows networking, and miscellaneous other IP traffic; for the purposes of our study, we do not consider these packets any further. We then organize the categorized packets into contiguous *spans*; we consider outgoing packets only and ignore packets in-bound to the client (with the exception of DeAuth packets sent by the base-station, which terminate the current span).

Figure 6 presents a time series of the average fraction of time an active station spends in each type of span. The graph plots five-second bins, averaged over one-minute intervals for clarity. Within each five-second bin, we calculate the fraction time each active station spends in each type of span, and normalize for the number of

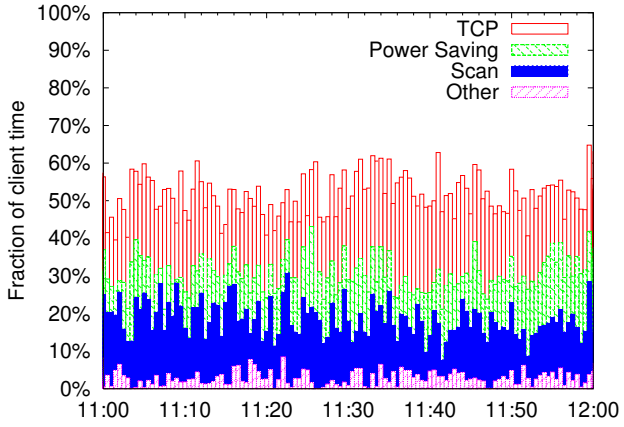

Figure 6: Time series of different types of spans.

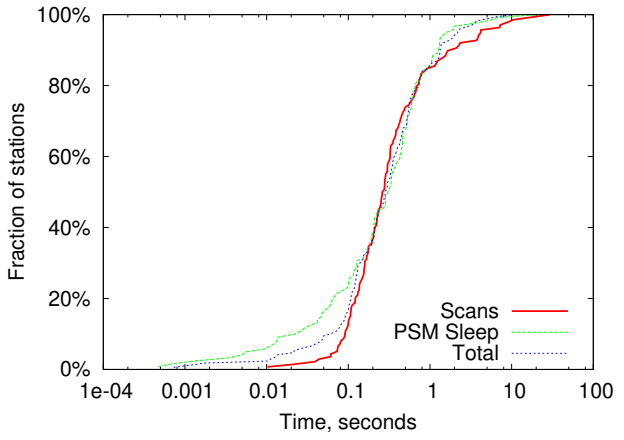

Figure 7: CDF of duration for scans and sleep periods that interrupt active communication.

stations active in that bin. If a station sends no packets in a five-second interval, it is not counted. While the absolute fraction of active time in any interval depends on the bin size (stations are bursty; the longer the sample period the less dense the activity), the relative length of each type of span remains relatively constant. From the graph we can conclude that roughly one third of a station's active time is spent scanning or in some other maintenance activity (ARP, DHCP, association, etc.)—overhead due to mobility maintenance.

5.2 Impact of scanning

Figure 6 shows that 802.11 clients are constantly scanning for other APs that may be better suited for them to associate to. If the station is otherwise idle at the time, scanning is inconsequential—at least from the point of view of the client. If the interface is busy, on the other hand, this behavior results in observable delay. Here, we attempt to quantify the delay observed by 802.11 clients due to scanning.

Our goal is to isolate those scan events that occur while the station was otherwise occupied. Because we do not know what a given client is doing at any point in time, we have to make a conservative estimate. For the purposes of this study, we consider a station busy when actively sending TCP packets; in other words, during TCP spans. Hence, we are interested in occasions when a TCP span is

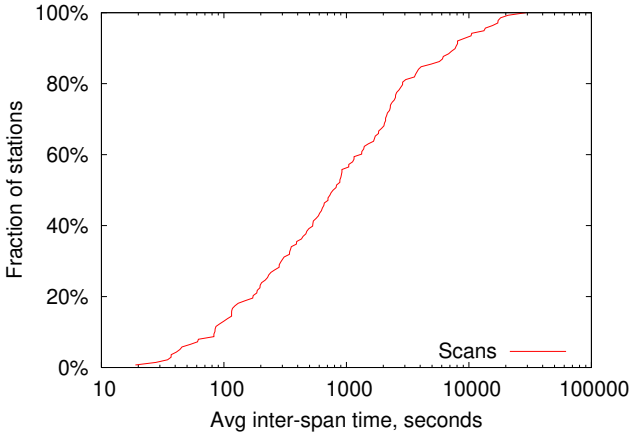

Figure 8: CDF of periods between scans that interrupt active communication on a client.

interrupted by another type of span. Specifically, we declare a station to be busy during a non-TCP span if the non-TCP span starts less than 100 milliseconds after a TCP packet and the TCP span resumes less than 500 milliseconds after the interruption. In addition, in an attempt to screen out occasions when the TCP spans on either side are unrelated (meaning that the station was not actually busy when the "interrupting" span occurred), we focus on interruptions of less than a minute. Each of these timeouts were arrived at experimentally, but are fairly arbitrary: the CDF of time before and after TCP spans is quite smooth and has no obvious modalities.

About 40% of stations have no interruptions at all, possibly because our criteria is too strict, the cards are smart enough to avoid interrupting, or the stations are just not active enough during our monitoring period. For the remaining 60% of the stations, Figure 7 shows the CDF of the duration of the interruptions. The average interruption lasts for roughly 250 ms, and over 20% of the interruptions last longer than one second. Most interruptions are caused by scanning behavior, but we also observe a substantial number of occasions where the station goes into power-save mode (i.e., sends a NULL packet with power save on, followed eventually by NULL packet with power save off). The "PSM Sleep" line in Figure 7 shows that while such interrupts can be much shorter, the average duration is roughly comparable, and is very unlikely to take longer than a second.

Short interruptions might be tolerable if they occurred infrequently, but Figure 8 shows that, for hosts that experience interruptions, they occur with wildly varying frequency. The average interrupted host is interrupted only once every 10 minutes or so, but 5% of the interrupted hosts are interrupted more than once a minute.

In addition to delaying traffic at the scanning station itself, probes also tend to exacerbate the hidden terminal problem. Recall that the hidden terminal problem occurs when two stations attempt to transmit to the same third station simultaneously. A scan probe might be received by multiple nearby APs which are hidden to each other. Then hidden APs will attempt to respond simultaneously and cause interference at the client. We are able to detect overlapping transmissions by comparing the start timestamp of every packet destined for an AP with end timestamps of previous packets directed to the same AP. If they overlap, we mark both packets as having collided due to hidden terminal.

We compute the fraction of all packets from a given client that result in a hidden terminal collision, and use that to plot CDF shown

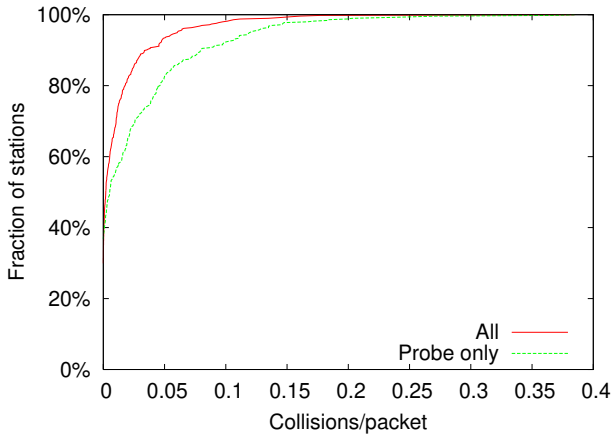

Figure 9: CDF of periods between scans that interrupt active communication on a client.

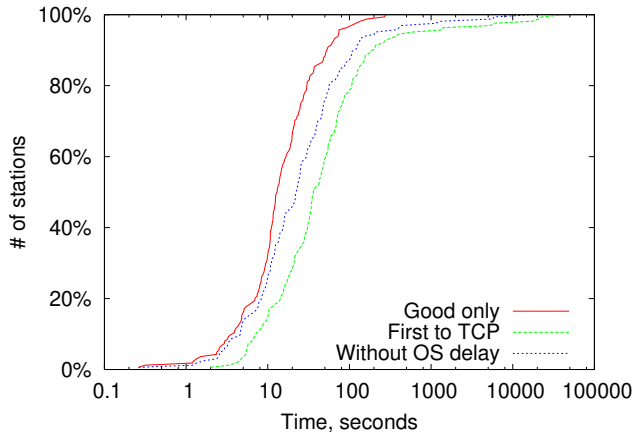

Figure 10: CDF of the delay experienced on startup by 802.11 clients in our network.

in Figure 9. In general, roughly 40% of clients hardly ever see a collision, while 10% of the clients have 5% of all their packets colliding due to hidden terminal. Focusing on just probe traffic, however, we observe that over half of the stations sent probes that collided with another station's packets, and, for the worst offenders, over 10% of their probes collided with other stations' packets.

5.3 Startup

When a client first appears on the 802.11 network it must initiate a sequence of steps to effectively join the network before it can communicate at the IP level. The standard behavior of a host is as follows:

- Scan. Determine a candidate AP to associate with.
- Associate. Attempt to associate with the chosen AP.
- DHCP. Once the host has successfully joined the 802.11 network, it must obtain an IP address to begin communicating. In our environment, hosts obtain a dynamic IP address through DHCP.
- ARP. Equipped with an IP address, the first thing a host must do is determine the MAC address of the next-hop router in order to route IP packets towards their destination. Hence, the host will issue an ARP "who has" for the IP address of the next-hop router indicated by the DHCP server.
- DNS. Finally, once IP routing is established, the host can begin communicating with a non link-local IP address. Typically remote hosts are identified through domain names, so the host must resolve the name using the domain name service. Once DNS resolves the IP address of the destination, the host can begin sending actual data.

We begin by considering the delay associated with end-system startup. In an attempt to isolate those stations that are truly starting up—as opposed to simply re-associating after a period of idleness—we define a set of candidate selection rules. A station is deemed to be starting up if the first packet we see from it is a scan request. Because we are interested in the behavior of clients that should be able to use the network, we only consider stations that eventually succeed in associating with one of our access points and send at least one TCP packet.

Figure 10 shows the distribution of startup times for those clients that do successfully connect to our network. There are three curves; "First to TCP" is the total wall-clock time from the first probe request to the first TCP segment. Surprisingly, most hosts take more than 10 seconds before they begin communicating on the network, and the average host takes almost a minute. We conjecture, however, that the bulk of that time is spent idling—meaning the machine is not actively trying to make progress towards sending data.

In order to validate our conjecture, we attempt to determine if each successive span was successful or not—if successful, time between spans is due to the Operating System. Time between failed spans, however, is likely due to some sort of network timeout. We define a scan to be successful if it is not followed by a subsequent scan; association, DHCP, and DNS are successful if the last packet in the span was outgoing from base station to client (i.e., an ACK). The "without OS delay" line removes estimated OS delays from the measured startup latency by subtracting idle time between successful spans under the presumption that any delay in initiating the subsequent span is due to the end host (i.e., the operating system has not yet initialized the network stack).

The average host spends almost eight seconds idling, presumably because the operating system is booting or resuming from power-save mode. Interestingly, however, if we sum only the duration of successful spans, we observe that the average host spends over 20 seconds during or after unsuccessful spans. The "good only" line represents a best-case scenario, with no idle time between stages. The question, then, is what's going wrong—why the big gap between optimal and common case? In order to address this question, we first examine the successful spans.

Even the successful spans take a non-trivial amount of time. Figure 11 shows the breakdown of the various stages in the startup process. This uses span durations only, and ignores time after. (Summing all curves from this graph together yields the "Good only" curve from above.) The vast majority of the time is spent scanning for an appropriate access point with which to associate. Association itself generally takes less than 10 ms, at which point network-level communication can begin. The process of association is generally quite quick (typically about 10 ms). DHCP, on the other hand, because it depends on a remote server, can take a variable amount of time. We will expand on the performance of DHCP in our environment in the next section. For now, however, we note it generally takes somewhere between 10 ms and five seconds to obtain an IP

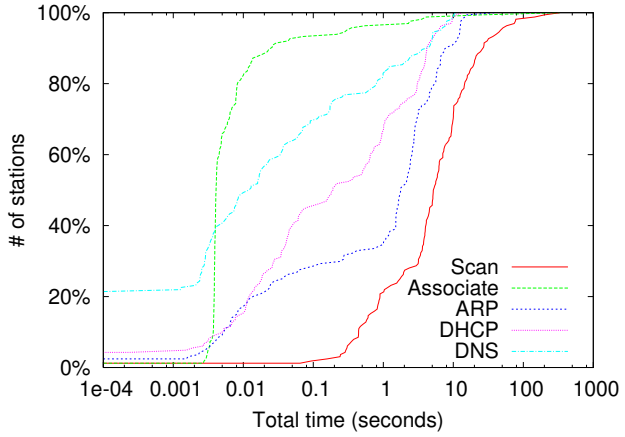

Figure 11: CDF of time spent in each successful phase of startup.

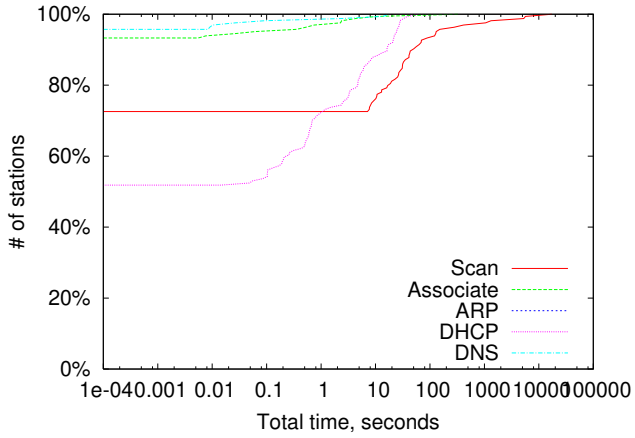

Figure 12: CDF of delays experiences by 802.11 clients due to timeouts.

address.

Surprisingly, ARP, while frequently fast, takes longer than one second in more than half the cases. This is because most stations issue an "ARP to self"—an ARP "who has" request for their own IP address—to ensure no other station is using that IP address before they begin communication. By design, such an ARP request must timeout, hence the one-second delay. Note that some graphs start at greater than 0%; this means some clients do not send those packets during startup. For example, over 20% of the hosts do not issue a DNS query before starting a TCP connection, presumably because they are communicating with a manually specified IP address or because the corresponding DNS entry was found in cache.

Returning to unsuccessful spans, we can observe that timeouts can be quite expensive. Figure 12 shows that while some stages, like DNS and association, frequently timeout in about 10 ms, they can take 10s of seconds to complete in the worst case. The minimum DHCP timeout appears to be 100 ms, and goes up from there. Failed scans are extremely expensive (a minimum of 7 seconds) because a failed scan probably means there are no desireable access points in range. In this situation, it makes no sense to retry after short timeout, and most stations would wait for at least 10 seconds before re-scanning the network. More interestingly, some

DHCP Transactions	611 (100%)
Client had no known current lease	204 (33.39%)
Client had used 25% of current lease	288 (47.14%)
Client newly associated shortly before	193 (31.59%)
Client re-associated shortly before	56 (9.17%)
No valid reason determined	76 (12.44%)

Table 2: Potential reasons why clients initiated DHCP transactions over a day. For some DHCP transactions multiple potential reasons exist.

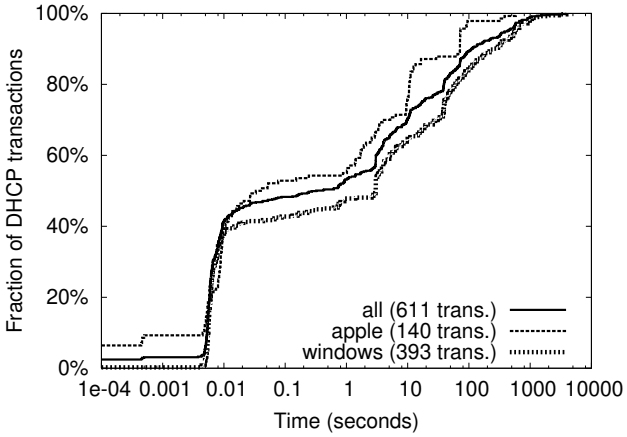

Figure 13: Distribution of DHCP transaction durations for an entire day.

hosts continue to scan for extremely long periods of time, presumably because they never find an AP they wish to join; i.e., they're looking for a non-existent SSID.

5.4 Dynamic address assignment

Dynamic address assignment using DHCP is an inherent aspect of 802.11 wireless networks. It is convenient for both users and network administrators, but the results above also indicate that DHCP can potentially impose noticeable and annoying delays to wireless users who desire and expect to be able to use the network quickly. In this section, we look at DHCP delays more closely.

Clients initiate DHCP transactions for a variety of reasons: their last least expired (or they had none), their existing lease starting to expire (conservatively, 75% of lease time remaining), they newly associate with an AP, or they reassociated with an AP. For instance, Table 2 shows a breakdown of the reasons why clients in our building initiate DHCP transactions for an entire typical weekday of use. The dominant reason for DHCP transactions are clients contacting the server to start the lease renewal process. The vast majority of leases in our network are for three hours, so stable clients, once connected, initiate DHCP transactions throughout the day.

How long are DHCP transactions? Figure 13 shows the distribution of the duration of DHCP transactions for a typical weekday in the building ("all"). These results show that the majority of transactions complete in a reasonable amount of time: 75% of transactions complete in under six seconds. From the perspective of diagnosing problems, however, as long as there are users experiencing annoying delays, these delays are a problem that plague both users and administrators. On this day, over 10% of transactions took over a minute; for users connecting to the network for the first time that day, such a delay is quite noticeable.

Based on well-known Ethernet vendor codes for MAC addresses

and the "Vendor Class" option in the DHCP protocol, we can determine the manufacturer of the operating system and networking hardware for most of the 186 stations in the trace. For comparison, we further group the various versions of Microsoft Windows as "Windows" (118 stations) and hardware manufactured by Apple as "Apple" (51 stations) and show distributions for these groups as well. Apple clients consistently experience longer DHCP transactions than other clients, in particular Windows. Apple clients use the Zeroconf standard, which cause them to spend an additional 10 seconds on startup by optimistically requesting based on an invalid lease.

6. Conclusion

Modern enterprise networks are of sufficient complexity that even *simple* faults can be difficult to diagnose — let alone transient outages or service degradations. Nowhere is this problem more apparent than in the 802.11-based wireless access networks now ubiquitous in the enterprise. We believe that such diagnosis must be automated, and that networks must eventually address transient failure without human involvement. As a first step in this direction, we have built an on-line analysis system that processes raw wireless trace data and can accurately model the impact of protocol behavior from the physical layer to the transport layer. While some sources of delay can be directly measured, many of the delay components, such as AP queuing, backoffs, contention, etc., must be inferred. To infer these delays from measurements, we develop a detailed model of MAC protocol behavior, both as it is described in the 802.11 specification as well as how it is implemented in vendor hardware. We also explore an inherent class of overheads due to mobility management in 802.11 networks, including scanning for access points, association, ARP, DHCP, authentication, etc. Using our system we investigate the causes of transient performance problems from traces of wireless traffic in a four-story office building. We find that no one anomaly, failure or interaction is singularly responsible for these issues and that a holistic analysis may in fact be necessary to cover the range of problems experienced in real networks.

7. REFERENCES

[1] P. Bahl, J. Padhye, L. Ravindranath, M. Singh, A. Wolman, and B. Zill. Dair: A framework for managing enterprise wireless networks using desktop infrastructure. In *Proceedings of the Fourth Workshop on Hot Topics in Networking (HotNets)*, Nov. 2005.

[2] P. Bahl and V. N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *Proceedings of IEEE Infocom Conference*, Mar. 2000.

[3] A. Balachandran, G. M. Voelker, P. Bahl, and P. V. Rangan. Characterizing User Behavior and Network Performance in a Public Wireless LAN. In *Proceedings of ACM SIGMETRICS*, 2002.

[4] P. Barford and M. Crovella. Critical path analysis of TCP transactions. In *Proceedings of the ACM SIGCOMM Conference*, Stockholm, Sweden, Aug. 2000.

[5] R. Chan, J. Padhye, A. Wolman, and B. Zill. A location-based management system for enterprise wireless lans. In *Proceedings of the 3rd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Mar. 2006.

[6] P. Chatzimisios, A. C. Boucouvalas, and V. Vitsas. Performance analysis of the ieee 802.11 mac protocol for wireless lans. *Wiley International Journal of Communication Systems*, 18(6):545–569, June 2005.

[7] Y.-C. Cheng, J. Bellardo, P. Benko, A. C. Snoeren, G. M. Voelker, and S. Savage. Jigsaw: Solving the puzzle of enterprise 802.11 analysis. In *Proceedings of the ACM SIGCOMM Conference*, Pisa, Italy, Sept. 2006.

[8] Y.-C. Cheng, Y. Chawathe, A. Lamarca, and J. Krumm. Accuracy characterization for metropolitan-scale wi-fi localization. In *Proceedings of the ACM/USENIX Conference on Mobile Systems, Appliations and Services (MobiSys)*, Seattle, WA, June 2005.

[9] E. Daley. Enterprise LAN Grows Up, 2005. `http://www2.cio.com/analyst/report3401.html`.

[10] K. N. Gopinath, P. Bhagwat, and K. Gopinath. An empirical analysis of heterogeneity in ieee 802.11 mac protocol implementations and its implications. In *Proceedings of WiNTECH 2006*, 2006.

[11] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki. Practical robust localization over large-scale 802.11 wireless networks. In *Proceedings of the ACM MOBICOM Conference*, pages 70–84, 2004.

[12] T. Henderson, D. Kotz, and I. Abyzov. The Changing Usage of a Mature Campus-wide Wireless Network. In *Proceedings of ACM Mobicom*, 2004.

[13] A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer. Understanding Congestion in IEEE 802.11b Wireless Networks. In *Proceedings of ACM IMC*, 2005.

[14] A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer. Understanding Link-Layer Behavior in Highly Congested IEEE 802.11b Wireless Networks. In *Proceedings of ACM E-WIND*, 2005.

[15] J. Jun, P. Peddabachagari, and M. Sichitiu. Theoretical maximum throughput of ieee 802.11 and its applications. In *Proceedings of the 2nd IEEE International Symposium on Network Computing and Applications*, 2004.

[16] D. Kotz and K. Essien. Analysis of a Campus-wide Wireless Network. In *Proceedings of ACM Mobicom*, 2002.

[17] B.-J. Kwak, N.-O. Song, and L. E. Miller. Performance analysis of exponential backoff. *IEEE/ACM Transactions on Networking*, 13(2), Apr. 2005.

[18] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Analyzing the MAC-level Behavior of Wireless Networks in the Wild. In *Proceedings of ACM SIGCOMM*, 2006.

[19] A. Mishra, M. Shin, and W. Arbaugh. An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process. *ACM Computer Communications Review*, 33(2), 2003.

[20] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling tcp reno performance: A simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, Apr. 2000.

[21] A. Sheth, C. Doerr, D. Grunwald, R. Han, and D. Sicker. Mojo: a distributed physical layer anomaly detection system for 802.11 wlans. In *Proceedings of MobiSys*, pages 191–204, June 2006.

[22] J. Yeo, M. Youssef, and A. Agrawala. A Framework for Wireless LAN Monitoring and its Applications. In *Proceedings of ACM WiSe*, 2004.