Passive Online Rogue Access Point Detection Using Sequential Hypothesis Testing with TCP ACK-Pairs

Wei Wei†, Kyoungwon Suh†, Bing Wang‡, Yu Gu†, Jim Kurose†, Don Towsley†
†Department of Computer Science
University of Massachusetts, Amherst, MA 01003
‡Computer Science & Engineering Department
University of Connecticut, Storrs, CT 06269

UMass Computer Science Technical Report 2006-60
Nov-5-2006
Updated on August 21, 2007

Abstract

Rogue (unauthorized) wireless access points pose serious security threats to local networks. In this paper, we propose two online algorithms to detect rogue access points using sequential hypothesis tests applied to packet-header data collected passively at a monitoring point. One algorithm requires training sets, while the other does not. Both algorithms extend our earlier TCP ACK-pair technique to differentiate wired and wireless LAN TCP traffic, and exploit the fundamental properties of the 802.11 CSMA/CA MAC protocol and the half duplex nature of wireless channels. Our algorithms make prompt decisions as TCP ACK-pairs are observed, and only incur minimum computation and storage overhead. We have built a system for online rogue-access-point detection using these algorithms and deployed it at a university gateway router. Extensive experiments in various scenarios have demonstrated the excellent performance of our approach: the algorithm that requires training provides rapid detection and is extremely accurate (the detection is mostly within 10 seconds, with very low false positive and false negative ratios); the algorithm that does not require training detects 60%-76% of the wireless hosts without any false positives; both algorithms are light-weight (with computation and storage overhead well within the capability of commodity equipment).

I. Introduction

The deployment of IEEE 802.11 wireless networks (WLANs) has been growing at a remarkable rate during the past several years. The presence of a wireless infrastructure within a network, however, raises various network management and security issues. One of the most challenging issues is rogue access points (APs), i.e., wireless access points that are installed without explicit authorization from a local network management [9], [1], [3], [4]. Although usually installed by innocent users for convenience or higher productivity, rogue APs pose serious security threats to a secured network. They potentially open up the network to unauthorized parties, who may utilize the resources of the network, steal sensitive information or even launch attacks to the network. Furthermore, rogue APs may interfere with nearby well-planned APs and lead to performance problems inside the network.

Due to the above security and performance threats, detecting rouge APs is one of the most important tasks for a network manager. Broadly speaking, two approaches can be used to detect rogue APs. The first approach detects rogue APs by monitoring the RF airwaves, possibly exploiting additional information gathered at routers and switches [2], [8], [9], [1], [3], [4], [10], [11], [27]. The second approach monitors incoming traffic at a traffic aggregation point (e.g., a gateway router) and determines whether a host uses wired or wireless connection[1]. If

[1]A local network typically supports both Ethernet and WLAN technologies, and therefore the aggregation point observes a mixture of wired and wireless traffic. The scenario of purely wireless network is discussed in Section VIII-B.

a host is determined as using wireless connection while it is not authorized to do so (e.g., it is not contained in the authorization list), the AP attached by this host is detected as a rogue AP. The first approach can suffer from various drawbacks including scalability, deployment cost, effectiveness and accuracy (see Section I-A). The second approach does not have the above drawbacks: (1) since it is based on passive measurements at a *single* monitoring point, it is scalable, requiring little deployment cost and effort, and is easy to manage and maintain; (2) since the detection is by detecting wireless connections, it is equally applicable to detect layer-2 or layer-3 rogue devices while the first approach may need different schemes for rogues at different layers [10], [11]. The challenge in applying the second approach is: how to effectively detect wireless traffic from *passively* collected data in an *online* manner?

In this paper, we take the second approach and develop two *online* algorithms to meet the above challenges. Our main contributions are as follows:

- We extend the analysis in [25] and demonstrate that using TCP ACK-pairs can effectively differentiate Ethernet and wireless connections (including both 802.11b and 802.11g).
- We develop two online algorithms to detect rogue APs. Both algorithms use sequential hypothesis tests and make prompt decisions as TCP ACK-pairs are observed. One algorithm requires training data, while the other does not. To the best of our knowledge, ours are the first set of *passive online* techniques that detect rogue APs by differentiating connection types.
- We have built a system for online rogue-AP detection using the above algorithms and deployed it at the gateway router of the University of Massachusetts, Amherst (UMass). Extensive experiments in various scenarios have demonstrated the excellent performance of our algorithms: (1) The algorithm that requires training provides rapid detections and is extremely accurate (the detection is mostly within 10 seconds, with very low false positive and false negative ratios); (2) The algorithm that does not require training detects 60%-76% of the wireless hosts without any false positives; (3) Both algorithms are light-weight, with computation and storage overhead well within the capability of commodity equipment. We further conduct experiments to demonstrate that our scheme can detect connection-type switching and wireless networks behind a NAT box, and it is effective even when the hosts have high CPU, disk or network utilizations.

The rest of the paper is organized as follows. Section I-A describes related work. Section II presents the problem setting and a high-level description of our approach. Section III analyzes TCP ACK-pairs in Ethernet and WLAN. Sections IV and V present our online algorithms and online rogue-AP detection system, respectively. Sections VI and VII present experimental evaluation methodology and results, respectively. Section VIII discusses several practical issues related to rogue AP detection. Finally, Section IX concludes the paper.

A. Related work

As mentioned earlier, monitoring RF waves and IP traffic are two broad classes of approaches to detecting rogue APs. Most existing commercial products take the first approach — they either manually scan the RF waves using sniffers (e.g., AirMagnet [2], NetStumbler [8]) or automate the process using sensors (e.g.,[1], [9], [4]. Automatic scanning using sensors is less time consuming than manual scanning and provides a continuous vigilance to rogue APs. However, it may require a large number of sensors for good coverage, which leads to a high deployment cost. Furthermore, since it depends on signatures of APs (e.g., MAC address, SSID, etc.), it becomes ineffective when a rogue AP spoofs signatures. Three recent research efforts [10], [11], [27] also use RF sensing to detect rogue APs. In [10], wireless clients are instrumented to collect information about nearby APs and send the information to a centralized server for rogue AP detection. This approach is not resilient to spoofing. Secondly, it assumes that rogue APs use standard beacon messages in IEEE 802.11 and respond to probes from the clients, which may not

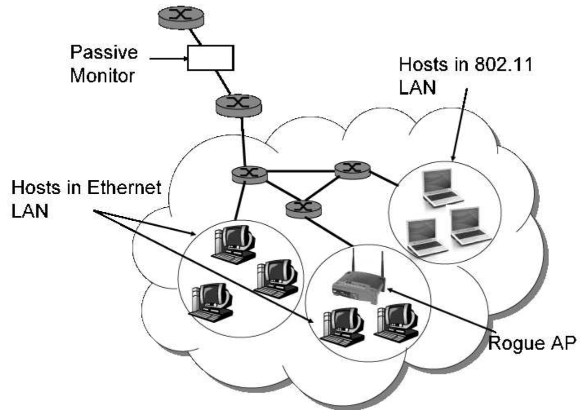

Fig. 1. Problem setting: a monitoring point at an aggregation point (e.g., the gateway router) captures incoming traffic and outgoing traffic to detect rogue APs.

hold in practice. Last, all unknown APs (including those in the vicinity networks) are flagged as rogue APs, which may lead to a large number of false positives. The main idea of [11] is to enable dense RF monitoring through wireless devices attached to desktop machines. This study improves upon [10] by providing more accurate and comprehensive rogue AP detection. However, it relies on proper operation of a large number of wireless devices, which can be difficult to manage. In contrast, our approach only requires a single monitoring point, and is easy to manage and maintain. The focus of [27] is on detecting protected layer-3 rogue APs. Our approach is equally applicable to detect layer-2 or layer-3 rogue devices.

The studies of [13], [19] detect rogue APs by monitoring IP traffic. The authors of [13] demonstrated from experiments in a local testbed that wired and wireless connections can be separated by visually inspecting the timing in the packet traces of traffic generated by the clients. The settings of their experiments are very restrictive. Furthermore, the visual inspection method cannot be carried out automatically. Our schemes are based on a *rigorous analysis* of Ethernet and wireless traffic characteristics in *realistic* settings. Furthermore, we provide two sequential hypothesis tests to *automatically* detect rogue APs *in real time*. The technique in [19] requires segmenting large packets into smaller ones, and hence is not a passive approach.

There are several prior studies on determining connection types. However, none of them provides a passive online technique, required for our scenario. Our previous work [25] proposes an iterative Bayesian inference technique to identify wireless traffic based on passive measurements. This iterative approach is not suitable for online deployment. The work of [12] uses entropies to detect wireless connection in an *offline* manner. In other studies, differentiating connection types is based on active measurements [26] or certain assumptions about wireless links (such as very low bandwidth and high loss rates) [15], which do not apply to our scenario.

Last, sequential hypothesis testing [24] provides an opportunity to make decisions as data come in, and thus is a suitable technique for our purpose. It is also used for prompt portscan detection in [18].

II. Problem Setting and Approach

Consider a local network (e.g., a university campus or an enterprise network), as illustrated in Fig. 1. A monitoring point is placed at an aggregation point (e.g., the gateway router) of this local network, capturing traffic coming in and going out of the network. End hosts within this network use either wired Ethernet or 802.11 WLAN to access the Internet. An end host not authorized to use WLAN may install a rogue AP to connect to the network. Our goal is to detect those rogue APs in *real time* based on *passive* measurements at the monitoring point. For this purpose,

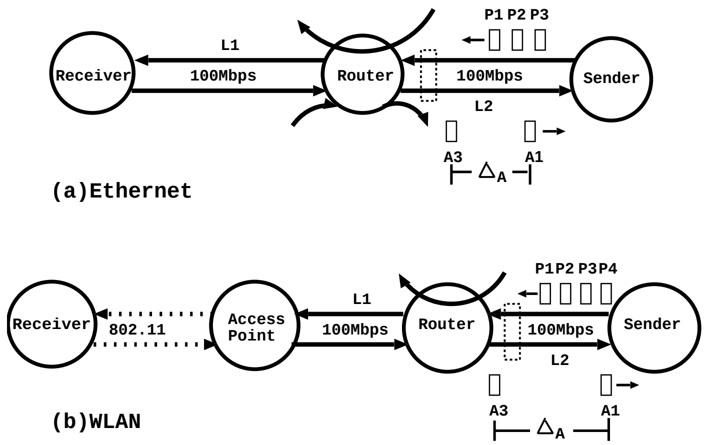

Fig. 2. Settings for the analysis: (a) Ethernet, (b) WLAN (802.11b or 802.11g). The dashed rectangle between the sender and the router represents the monitoring point. The pair of ACKs, A_1 and A_3, forms an ACK-pair.

we must answer the following two questions: (1) what statistics can be used to effectively detect wireless hosts? (2) how to detect wireless hosts in an online manner? We next provide a high-level description on how we address these two questions; a detailed description is deferred to Sections III and IV.

We have shown that *inter-ACK time* is a statistic that can be used to effectively detect wireless hosts in [25]. An inter-ACK time is the inter-arrival time of a *TCP ACK-pair*, i.e., a pair of ACKs corresponding to two data packets that arrive at the monitoring point close in time. In [25], we analyze the inter-ACK time in Ethernet and WLAN and demonstrate that it can be used to differentiate these two connection types. However, the analysis does not include 802.11g, since it was not widely deployed at that time. In Section III, we extend the analysis in [25] to 802.11g, and derive a new set of results for Ethernet and 802.11b. Our results demonstrate that inter-ACK times can effectively differentiate Ethernet and WLAN (including both 802.11b and 802.11g hosts).

For online detection of wireless hosts, we develop two light-weight algorithms (see Section IV), both using sequential hypothesis tests and taking the inter-ACK times as input. These two algorithms roughly work as follows. They calculate the likelihoods that a host uses WLAN and Ethernet as TCP ACK-pairs are observed. When the ratio of the WLAN likelihood against the Ethernet likelihood exceeds a certain threshold, they make a decision that the host uses WLAN.

III. ANALYSIS OF TCP ACK-PAIRS

In this section, we extend the analysis in [25] and demonstrate analytically that inter-ACK time can be used to effectively differentiate Ethernet and WLAN (including both 802.11b and 802.11g). In the following, we start from the assumptions and settings, and then present the analytical results. At the end, we briefly summarize the insights obtained from the analysis.

A. Assumptions and settings

The settings for our analysis are shown in Fig. 2, where an outside sender sends data to a receiver in the local network. In Fig. 2(a), the receiver uses Ethernet; in Fig. 2(b), the receiver uses 802.11b or 802.11g WLAN. We refer to the above settings as *Ethernet setting* and *WLAN setting*, respectively. In both settings, a router resides between the sender and the receiver, and is connected to the sender by link L_2 with 100 Mbps bandwidth. The monitoring point is between the sender and the router, tapping into link L_2. In the Ethernet setting, the router and the receiver are connected by link L_1 with 100 Mbps bandwidth. In the WLAN setting, an access point resides between the router and the receiver. The access point and the router are connected by link L_1 with 100 Mbps bandwidth; and the receiver is connected to the access point using 11 Mbps 802.11b or 54 Mbps 802.11g. In both the Ethernet and WLAN settings, the router's queues for incoming data packets and ACKs are modeled as $M/D/1$ queues. Let Q_D and Q_A denote the queues for data and ACKs respectively. The utilizations of Q_D and Q_A are ρ_D and ρ_A, respectively.

We assume that the receiver implements delayed ACK policy[2], since this policy is commonly used in practice [21], [7]. To accommodate the effects of delayed ACK, we consider four data packets P_1, P_2, P_3 and P_4, each of 1500 bytes, sent back-to-back from the sender. Without loss of generality, we assume that packet P_1 is acknowledged. Since we assume delayed ACK, packet P_3 is also acknowledged. Let A_1 and A_3 denote the ACKs corresponding to packets P_1 and P_3, respectively. Then A_1 and A_3 form an ACK-pair. Let Δ_A represent the inter-ACK time of A_1 and A_3 at the monitoring point. Let Δ denote the inter-arrival time of the data packets P_1 and P_3 at the monitoring point. Then $\Delta = 120 \times 2 = 240\ \mu s$ since each P_i $(i = 1, \ldots, 4)$ is 1500 bytes and the bandwidth of link L_2 is 100 Mbps.

Intuitively, the random backoff mechanism in 802.11 (i.e., a host must wait for a random backoff interval to transmit [17]) and the half duplex nature of wireless channels (i.e., data packets and ACKs contend for media access at a wireless host) may lead to larger inter-ACK times in WLAN than those in Ethernet. To demonstrate analytically that this is indeed the case, we consider the following worst-case scenarios (in terms of differentiating Ethernet and WLAN hosts). In the Ethernet setting, we assume cross traffic traversing both queues, Q_D and Q_A, at the router so that the Ethernet link may be heavily utilized. In the WLAN setting, the wireless link between the access point and the receiver is under *idealized conditions*, i.e., the channel is perfect, and is only used by the access point and the receiver. As we shall see, even in the above scenarios, the inter-ACK times of WLAN are generally larger than those of Ethernet, and hence can be used to differentiate WLAN and Ethernet connections.

B. Analysis of Ethernet

We next present two theorems on inter-ACK times in the Ethernet setting. Their proofs are found in Appendices I and II, respectively.

Theorem 1: **(Inter-ACK time distribution for Ethernet)** In the Ethernet setting, when $0 < \rho_D, \rho_A \leq 1$, $P(\Delta_A > 600\ \mu s) < 0.18$.

We next consider the sample median distribution of inter-ACK times, and calculate the probability that it exceeds $600\ \mu s$. Let $\{\Delta_i^A\}_{i=1}^n$ denote an i.i.d sequence of n inter-ACK times from a host (they can be from different TCP flows). Let $\xi_{.5}^n(\Delta_A)$ denote the sample median of $\{\Delta_i^A\}_{i=1}^n$. Then we have the following theorem on $\xi_{.5}^n(\Delta_A)$.

Theorem 2: **(Median inter-ACK time for Ethernet)** In the Ethernet setting, for a given i.i.d sequence of sample inter-ACK times $\{\Delta_i^A\}_{i=1}^n$, when $0 < \rho_D, \rho_A \leq 1$ and $43 \leq n \leq 100$, we have $P(\xi_{.5}^n(\Delta_A) \leq 600\ \mu s) \approx 1$. Furthermore, $\lim_{n\to\infty} P(\xi_{.5}^n(\Delta_A) \leq 600\ \mu s) = 1$.

Both of the above theorems will be used explicitly to construct a sequential hypothesis test in Section IV-B.

C. Analysis of 802.11b WLAN

We now analyze the inter-ACK time distribution in the 802.11b WLAN setting. As mentioned earlier, we assume idealized conditions, that is, the wireless channel between the access point and the receiver is perfect and there is no contention from other wireless nodes. For 11 Mbps 802.11b, the transmission overhead for a TCP packet with zero payload is $508\ \mu s$, which includes the overhead to transmit physical-layer, MAC-layer, IP and TCP headers,

[2]That is, a receiver releases an ACK after receiving two packets, or if the delayed-ACK timer is triggered after the arrival of a single packet.

the overhead for ACK transmission, and the durations of one SIFS and DIFS [16]. The slot time is $20\ \mu s$ and a wireless device waits for a random backoff time uniformly distributed in $[0, 31]$ time slots (i.e., $[0, 620]\ \mu s$) before transmitting a packet. Therefore, the *MAC service time* (i.e., the sum of the constant transmission overhead and the random backoff time) of a data packet of 1500 bytes is uniformly distributed in $[1570, 2190]\ \mu s$. The MAC service time of an ACK of 40 bytes is uniformly distributed in $[508, 1128]\ \mu s$. We have the following theorem for the 802.11b WLAN setting; the proof is found in Appendix III.

Theorem 3: **(Inter-ACK time distribution for 802.11b)** In the 802.11b WLAN setting, under idealized conditions, $P(\Delta_A > 600\ \mu s) > 0.96$.

D. Analysis of 802.11g WLAN

We next show that 54 Mbps 802.11g WLAN generally has larger inter-ACK times than 100 Mbps Ethernet although they have comparable bandwidths. We again assume ideal conditions. For 54 Mbps 802.11g, the transmission overhead for a TCP packet with zero payload is $103\ \mu s$. The slot time is $9\ \mu s$. The receiver waits for a random backoff time uniformly distributed in $[0, 15]$ time slots (i.e., $[0, 135]\ \mu s$) before transmitting a packet. Therefore, the MAC service time of a data packet (1500 bytes) is uniformly distributed in $[325, 460]\ \mu s$; the MAC service time of an ACK (40 bytes) is uniformly distributed in $[109, 244]\ \mu s$. We have the following theorem for the 802.11g WLAN setting; the proof is found in Appendix IV.

Theorem 4: **(Inter-ACK time distribution for 802.11g)** In the 802.11g WLAN setting, under idealized conditions, $P(\Delta_A > 600\ \mu s) > 0.45$.

E. Summary of Analysis

The above analysis demonstrates that, even when WLAN is under idealized conditions while Ethernet LAN is fully utilized, using TCP ACK-pairs can effectively differentiate Ethernet and WLAN connections: for Ethernet, less than 18% of the inter-ACK times exceed $600\ \mu s$, while for 802.11b and 802.11g, at least 96% and 45% of the inter-ACK times exceed $600\ \mu s$ (see Theorems 1, 3 and 4). Under more realistic conditions (e.g., noisy wireless channel and with contention), inter-ACK times in WLAN may be even higher than those in Ethernet. Last, our analysis is based on the fundamental properties of the 802.11 CSMA/CA MAC protocol and the half-duplex nature of wireless channels, thus indicating that using inter ACK-time is a robust technique and cannot be easily spoofed (e.g., it is robust against MAC-address spoofing).

IV. Online Detection Algorithms

In this section, we develop two online algorithms to detect wireless hosts based on our analysis in the previous section. Both algorithms use sequential hypothesis test technique and take the inter-ACK times as the input. The first algorithm requires knowing the inter-ACK time distributions for Ethernet and WLAN traffic *a priori*. The second algorithm does not have such a requirement. Instead, it is directly based on Theorems 1 and 2 (see Section III). We refer to these two algorithms as *sequential hypothesis test with training* and *sequential hypothesis test without training* respectively. The algorithm without training, although is not as powerful as the one with training (see Section VII), is suitable for scenarios where the inter-ACK time distributions are not available *a priori* (e.g., for organizations with no wireless networks).

We now describe these two algorithms in detail. Both algorithms use at most $N = 100$ ACK-pairs to make a decision (i.e., whether the connection is Ethernet or WLAN) to accommodate the scenarios where a host switches between Ethernet and WLAN connections.

```
n = 0, l_E = l_W = 0.
do {
  Identify an ACK-pair
  n = n + 1
  p_n = P(Δ_n^A = δ_n^A | E), q_n = P(Δ_n^A = δ_n^A | W)
  l_E = l_E + log p_n, l_W = l_W + log q_n

  if l_W − l_E > log K
    Report WLAN, n = 0, l_E = l_W = 0.

  else if l_W − l_E < − log K
    Report Ethernet, n = 0, l_E = l_W = 0.

  else if n = N
    Report undetermined, n = 0, l_E = l_W = 0.
}
```

Fig. 3. Sequential hypothesis test with training, $N = 100$.

A. *Sequential Hypothesis Test with Training*

We have demonstrated that the inter-ACK time distributions for Ethernet and WLAN differ significantly (see Section III). When these distributions are known, we can calculate the likelihoods that a host uses Ethernet and WLAN respectively given a sequence of observed inter-ACK times. If the likelihood of using WLAN is much higher than that of using Ethernet, we conclude that the host uses WLAN (and vice versa).

We now describe the test in more detail. Let $\{\delta_i^A\}_{i=1}^n$ represent a sequence of inter-ACK time observations from a host, and $\{\Delta_i^A\}_{i=1}^n$ represent their corresponding random variables. Let E and W represent respectively the events that a host uses Ethernet and WLAN. Let $L_E = P(\Delta_1^A = \delta_1^A, \Delta_2^A = \delta_2^A, \ldots, \Delta_n^A = \delta_n^A \mid E)$ be the likelihood that this observation sequence is from an Ethernet host. Similarly, let $L_W = P(\Delta_1^A = \delta_1^A, \Delta_2^A = \delta_2^A, \ldots, \Delta_n^A = \delta_n^A \mid W)$ be the likelihood that the observation sequence is from a WLAN host. Let $p_i = P(\Delta_i^A = \delta_i^A \mid E)$ be the probability that the i-th inter-ACK time has value δ_i^A given that it is from an Ethernet host. Similarly, let $q_i = P(\Delta_i^A = \delta_i^A \mid W)$ be the probability that the i-th inter-ACK time has value δ_i^A given that it is from a WLAN host. Both p_i and q_i are known, obtained from the inter-ACK time distributions for Ethernet and WLAN traffic respectively. Assuming that the inter-ACK times are independent and identically distributed, we have

$$L_E = P(\Delta_1^A = \delta_1^A, \ldots, \Delta_n^A = \delta_n^A \mid E) = \prod_{i=1}^n p_i,$$

$$L_W = P(\Delta_1^A = \delta_1^A, \ldots, \Delta_n^A = \delta_n^A \mid W) = \prod_{i=1}^n q_i.$$

This test updates L_W and L_E as an ACK-pair is observed. Let $K > 1$ be a threshold. If after the n-th ACK-pair, the ratio of L_W and L_E is over the threshold, i.e., $L_W/L_E > K$, then the host is classified as a WLAN host. If $L_W/L_E < 1/K$, then the host is classified as an Ethernet host. If neither decision is made after N ACK-pairs, the connection type is classified as undetermined. In the implementation, for convenience, we use log-likelihood function $l_w = \log(L_W)$ and $l_E = \log(L_E)$ instead of the likelihood function.

This test is summarized in Fig. 3. As we can see, it has very little computation and storage overhead (it only stores the current likelihoods for Ethernet and WLAN for each IP address being monitored).

```
m = n = 0.
do {
  Identify an ACK-pair
   n = n + 1
   m = m + 1(δ_n^A ≥ 600 μs)
   p̂ = m/n

   if p̂ = 1 and n > −(log K)/(log θ)
     Report WLAN. m = n = 0.

   else if n < (m(log p̂ − log θ + log(1−θ) − log(1−p̂)) − log K)/(log(1−θ) − log(1−p̂))
     Report WLAN. m = n = 0.

   else if n ≥ 43 and p̂ ≥ 0.5
     Report WLAN. m = n = 0.

   else if n = N
     Report undetermined. m = n = 0.
}
```

Fig. 4. Sequential hypothesis test without training, , where $\mathbf{1}(\cdot)$ is the indicator function, $N = 100$.

B. Sequential Hypothesis Test without Training

This test does not require knowing the inter-ACK time distributions for Ethernet and WLAN hosts *a priori*. Instead, it leverages the analytical results that the probability of an inter-ACK time exceeding $600\ \mu s$ is small for Ethernet hosts, while it is much larger for WLAN hosts (see Section III). In the following, we first construct a likelihood ratio test [14], and then derive from it a sequential hypothesis test.

The likelihood ratio test is as follows. Let p be the probability that an inter-ACK time exceeds $600\ \mu s$, that is, $p = P(\Delta_A > 600\ \mu s)$. By Theorem 1, we have $p < \theta = 0.18$ for Ethernet host. Therefore, if the hypothesis $p < \theta$ is rejected by the inter-ACK time observation sequence, we conclude that this host does not use Ethernet and hence uses WLAN. More specifically, consider two hypotheses, H_0 and H_a, representing respectively the null hypothesis that a host uses Ethernet and the alternative hypothesis that the host uses WLAN. For a sequence of inter-ACK time observations $\{\delta_i^A\}_{i=1}^n$, let m be the number of observations that exceed $600\ \mu s$. Let $K > 1$ be a threshold. Then the likelihood ratio test rejects the null hypothesis H_0 when

$$\lambda = \frac{\sup_{0 \le p \le \theta} p^m (1-p)^{n-m}}{\sup_{0 \le p \le 1} p^m (1-p)^{n-m}} < \frac{1}{K}$$

In the middle term above, the numerator is the maximum probability of having the observed sequence (which has m inter-ACK times exceeding $600\ \mu s$) computed over parameters in the null hypothesis (i.e., $0 \le p \le \theta$). The denominator of λ is the maximum probability of having the observed sequence over all possible parameters (i.e., $0 \le p \le 1$). If $\lambda < 1/K$, that is, there are parameter points in the alternative hypothesis for which the observed sample is much more likely than for any parameter points in the null hypothesis, the likelihood ratio test concludes that H_0 should be rejected. In other words, if $\lambda < 1/K$, the likelihood ratio test concludes that the host uses WLAN.

We now derive a sequential hypothesis test from the above likelihood ratio test. Let $\hat{p} = m/n$, where m is the number of inter-ACK times exceeding $600\ \mu s$ and n is the total number of inter-ACK times. It is straightforward to show that $\hat{p}$ is the maximum likelihood estimator of p, i.e., $\sup_{0\le p\le 1} p^m(1-p)^{n-m}$ is achieved when $p = \hat{p}$. When $\hat{p} \le \theta$, we have $\sup_{0\le p\le \theta} p^m(1-p)^{n-m} = \sup_{0\le p\le 1} p^m(1-p)^{n-m}$, and hence $\lambda = 1 > 1/K$. In this case, the null hypothesis H_0 is not rejected. Therefore, we only consider the case where $\theta < \hat{p}$, which can be classified into two cases:

Case 1: $\theta < \hat{p} < 1$**.** In this case, to reject the null hypothesis H_0, we need

$$\frac{\hat{p}^m(1-\hat{p})^{n-m}}{\theta^m(1-\theta)^{n-m}} \quad > \quad K$$

which is equivalent to

$$n < \frac{m(\log \hat{p} - \log \theta + \log(1-\theta) - \log(1-\hat{p})) - \log K}{\log(1-\theta) - \log(1-\hat{p})}. \tag{1}$$

Case 2: $\hat{p} = 1$**.** In this case, to reject the null hypothesis H_0, we need

$$\frac{1}{\theta^n} > K$$

which is equivalent to

$$n > -\frac{\log K}{\log \theta}. \tag{2}$$

When $K = 10^6$ and $\theta = 0.18$, from (2), we have $n \ge 8$. This implies that we need at least 8 ACK-pairs to detect a WLAN host for the above setting.

In addition to conditions (1) and (2), we also derive a complementary condition to reject the null hypothesis H_0 directly from Theorem 2. Theorem 2 states that, when the number of inter-ACK observations n is between 43 and 100, we have $P(\xi^n_{.5}(\Delta_A) \le 600\ \mu s) \approx 1$ for Ethernet hosts. Therefore, an additional condition to reject H_0 is when $43 \le n \le 100$ and $\hat{p} > 0.5$ (because this condition implies that at least half of the inter-ACK observations exceed $600\ \mu s$, that is, $\xi^n_{.5}(\Delta_A) > 600\ \mu s$, which contradicts Theorem 2).

We combine the above three conditions to construct a sequential hypothesis test as shown in Fig. 4. As we can see, this test has very little computational and storage overhead (it only stores the total number of inter-ACK times and the number of inter-ACK times exceeding 600 μs for each IP address being monitored). Last, note that it only reports WLAN hosts, while the sequential hypothesis test with training reports both WLAN and Ethernet hosts.

V. Online Rogue-AP Detection System

We design a system for online detection of rogue APs. This system consists of three major components as illustrated in Fig. 5. The data capturing component collects incoming and outgoing packet headers. These packet headers are then passed on to the *online detection engine*, where WLAN hosts are detected using the algorithms described in the previous sections. Once a WLAN host is detected, its IP address is looked up from an authorization list for rogue-AP detection. We next describe the online detection engine, the core component in the system, in more detail. Afterwards, we describe how to identify ACK-pairs in real time and obtain inter-ACK time distributions *a priori* (required by the sequential hypothesis test with training).

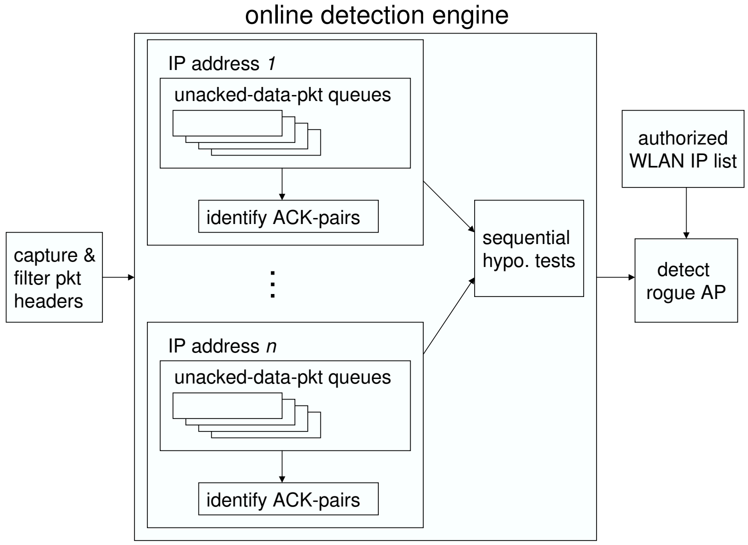

Fig. 5. Online rogue-AP detection system.

A. Online Detection Engine

The online detection engine makes a detection on a per host (or IP address) basis. Since TCP data packets and ACKs come in on a per flow basis and a host may have multiple simultaneous active TCP flows[3], the online detection engine maintains a set of data structures in memory, each corresponding to an active TCP flow. We name the data structure as an *unacked-data-packet queue* since it stores the information on all the data packets that have not been acknowledged by the receiver. Each item in a queue represents a data packet in the corresponding active flow. It records the sequence number (4 bytes), the timestamp (8 bytes) and size (2 bytes) of the packet. In addition, the online detection engine also records the latest ACK for each TCP flow in memory. These information is used to identify ACK-pairs as follows. For each incoming ACK, the online detection engine finds its corresponding unacked-data-packet queue (using a hash function for quick lookup) and then matches it with the items in the queue to identify ACK-pairs. Once an ACK-pair is identified, depending on whether training data is available, it is fed into the sequential hypothesis test with or without training to determine whether the host uses WLAN.

The memory requirement of the online detection system mainly comes from storing the unacked-data-packet queues. Each queue contains no more than M items, where M is the maximum TCP window size (since an item is removed from the queue once its corresponding ACK arrives). In our experiments, we find that most queues contain a very small number of items (see Section VII-C), indicating that the memory usage of this online detection system is low.

B. Online Identification of TCP ACK-pairs

As described earlier, two successive ACKs form an ACK-pair if the inter-arrival time of their corresponding data packets at the monitoring point is less than a threshold T (chosen as $240\ \mu s$ or $400\ \mu s$ in our system, see Section VII). In addition to the above condition, we also take account of several practical issues when identifying ACK-pairs. First, we exclude all ACKs whose corresponding data packets have been retransmitted or reordered. We also exclude ACKs due to expiration of delayed-ACK timers if delayed ACK is implemented (inferred using techniques in [25]). This is because, if an ACK is triggered by a delayed-ACK timer, it is not released immediately after a data packet.

[3]We define a flow that has not terminated and has data transmission during the last minute as an active flow.

Therefore, the inter-arrival time of this ACK and its previous ACK does not reflect the characteristics of the access link. Furthermore, to ensure that two ACKs are successive, we require that the difference of their IPIDs to be no more than 1. We also restrict that the ACKs are for relatively large data packets (of size at least 1000 bytes), to be consistent with the assumption of our analysis (in Section III). Last, we require that the inter-ACK time of an ACK-pair to be below 200ms. This is due to the following reasons. Consider three ACKs, the second and third ones being triggered by delayed-ACK timer. If the second ACK is lost, the measurement point will only observe a pair of ACKs (the first and third ACK), which is not a valid ACK-pair (since the third ACK is triggered by delayed-ACK timer). Requiring the inter-ACK time of an ACK-pair to be below 200 ms can exclude this pair of ACKs because their inter-arrival time is at least 200 ms (it takes at least 100 ms for a delayed-ACK timer to go off).

A user may purposely violate the above criteria for ACK-pairs (e.g., by never using TCP, using smaller MTUs or tampering with the IPID field) so that the measurement point does not capture any valid ACK-pair from this user. However, all the above cases are easy to detect and can raise an alarm that this user may attempt to hide a rogue AP.

C. Obtaining Inter-ACK Time Distributions Beforehand

To apply the sequential hypothesis test with training, we need to know the inter-ACK time distributions for Ethernet and WLAN beforehand. In general, the inter-ACK time distribution for a connection type can be acquired from a *training set*, which contains TCP flows known to use this connection type. We detail how we construct training sets for our experimental evaluation in Section VI-B; training sets for other networks can be constructed in a similar manner.

VI. Evaluation Methodology

We evaluate the performance of our rogue-AP detection algorithms through extensive experiments. In this section, we describe our evaluation methodology, including the measurement equipment, training sets, test sets, and offline and online evaluation.

A. Measurement Equipment

Our measurement equipment is a commodity PC, installed with a DAG card [6] to capture packet headers. It is placed at the gateway router of UMass, Amherst, connected via an optical splitter to the access link connecting the campus network to the commercial network. The TCP and IP headers of all the packets that traverse this link are captured by the DAG card, along with the current timestamp. The captured data are streamed to our online detection algorithms, which are running on the commodity PC. The PC has three Intel Xeon Y 2.80 GHz CPUs (cache size 512 KB), 2 Gbytes memory, and SCSI hard disks.

B. Training Sets

Training sets are required to obtain inter-ACK time distributions (see Section V-C). We construct training sets for our experimental evaluation as follows. First, based on our knowledge on the UMass campus network, we identify $\mathcal{E}$ and $\mathcal{W}$, denoting the set of IP addresses known to use Ethernet and WLAN respectively. The set $\mathcal{E}$ consists of IP addresses for hosts using 100 Mbps Ethernet in the Computer Science department. The set $\mathcal{W}$ consists of IP addresses that are reserved for the campus public WLAN (an 802.11 network providing wireless access to campus users at public places such as the libraries, campus eateries, etc.). The numbers of IP addresses in $\mathcal{E}$ and $\mathcal{W}$ are 648 and 1177 respectively. The training set for Ethernet (or WLAN) is constructed by extracting TCP flows destined to hosts in $\mathcal{E}$ (or $\mathcal{W}$) from a trace collected at the monitoring point. The trace for Ethernet was collected between

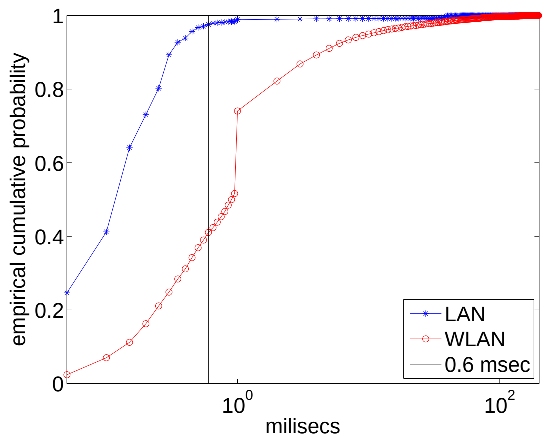

Fig. 6. Ethernet and WLAN inter-ACK time distributions obtained from training sets ($T = 240\ \mu s$).

February and April, 2005. In early 2006, 802.11g APs were deployed on UMass campus and more users start to use 802.11g. Therefore, we collected a new set of traces on $9/29/2006$ for WLAN. Note that the training set for WLAN contains a mixture of 802.11b and 802.11g traffic since a host can use either 802.11b or 802.11g depending on whether its wireless card and its associated AP support 802.11g.

From the training set (for Ethernet or WLAN), we identify a sequence of ACK-pairs, and discretize the inter-ACK times to obtain the inter-ACK time distribution. The discretization is as follows. We divide the range from 0 to 1 ms into $50\mu s$-bins, and divide the range from 1 ms to 200 ms (which is the maximum value for inter-ACK times) into 1ms-bins. Fig. 6 plots the CDFs (Cumulative Distribution Function) of the inter-ACK times for Ethernet and WLAN, where the threshold $T = 240\ \mu s$. We observe that 2.5% of the inter-ACK times for Ethernet hosts are above $600\ \mu s$, while 59.0% of the inter-ACK times for WLAN hosts are above $600\ \mu s$, confirming our analytical results in Section III (for Ethernet, the observed value is lower than the analytical result because our analysis is very conservative; for WLAN, the samples contain a mixture of 802.11b and 802.11g traffic).

C. Test Sets

To validate that our algorithms can detect WLAN hosts while does not misclassify Ethernet hosts, we construct a WLAN and an Ethernet test set, containing IP addresses known to use WLAN and Ethernet respectively. The WLAN test set contains the IP addresses (of 1177 addresses) reserved for the campus public WLAN. The Ethernet test set contains the IP addresses of a subset of Dell desktops that use Ethernet in the Computer Science building. It contains 258 desktops, each with documented IP address, MAC address, operating system, and location information for ease of validation. Among these desktops, 35% of them use different versions of Windows operating system (e.g., Windows 2000, Windows ME, Windows XP, etc.); the rest use different variants of Linux and Unix operating systems (e.g., RedHat, Solaris, CentOS, Fedora Core, etc.). These hosts are three hops away from the university gateway router (and the monitoring point).

In addition to these two test sets, we further investigate whether our schemes can detect connection switchings and other types of rogue APs by conducting additional experiments in the Computer Science Department. The total IP space monitored in our experimental evaluation consists of the WLAN test set (1177 addresses) and all the IP addresses in the Computer Science Department (2540 addresses), totally 3217 addresses.

TABLE I

OFFLINE EVALUATION OF SEQUENTIAL HYPOTHESIS TEST WITH TRAINING: RESULTS ON WLANS (10/20/2006).

	T=240 μs			T=400 μs		
	$K = 10^4$	$K = 10^5$	$K = 10^6$	$K = 10^4$	$K = 10^5$	$K = 10^6$
Avg. # of ACK-pairs for a detection	5	6	7	5	6	7
Avg. # of data pkts for a detection	250	288	347	204	235	283
Median detection time (sec)	8	10	13	6	8	11
Number of detections	$12,607$	$10,882$	$8,969$	$15,724$	$13,567$	$11,169$
Correct detection ratio	99.43%	99.59%	99.61%	99.38%	99.53%	99.61%
ACK-pair ratio	2%			2%		

TABLE II

OFFLINE EVALUATION OF SEQUENTIAL HYPOTHESIS TEST WITH TRAINING: RESULTS ON ETHERNET (10/20/2006).

	T=240 μs			T=400 μs		
	$K = 10^4$	$K = 10^5$	$K = 10^6$	$K = 10^4$	$K = 10^5$	$K = 10^6$
Avg. # of ACK-pairs for a detection	11	13	16	13	16	19
Avg. # of data pkts for a detection	87	106	124	73	89	106
Median detection time (sec)	0.6	1.0	1.2	0.3	0.6	0.9
Number of detections	$4,896$	$3,990$	$3,363$	$5,860$	$4,747$	$4,002$
Correct detection ratio	99.88%	100.00%	99.97%	99.61%	99.79%	99.78%
ACK-pair ratio	13%			17%		

D. Offline and Online Evaluation

We evaluate the performance of our algorithms in both offline and online manners. In offline evaluation, we first collect measurements (to the hard disk) and then apply the sequential hypothesis test to the collected trace. In online evaluation, we run the sequential hypothesis test online while capturing the data at the measurement point. The offline evaluation, although does not represent the normal operation mode of our algorithms, allows us to investigate the impact of various parameters (e.g., T, the threshold to identify ACK-pairs, K, the threshold in the sequential hypothesis tests). The online evaluation investigates the performance of our algorithms in their normal operation mode.

VII. EXPERIMENTAL EVALUATION

We now describe our experimental results. In our experiments, the online detection algorithms make a decision (detecting WLAN, Ethernet or undetermined) with at most N ACK-pairs, $N = 100$. A decision of WLAN or Ethernet is referred to as a *detection*. The time it takes to make a decision is referred to as *detection time*.

In the following, we first evaluate the performance (in terms of accuracy and promptness) of our online detection algorithms (Sections VII-A and VII-B). We then investigate the scalability of our approach (Section VII-C). Afterwards, we demonstrate that our approach is effective to detect other types of rogues (Section VII-D). Last, we show that our approach can quickly detect connection-type switchings (Section VII-E) and is robust to high CPU, disk or network utilization at end hosts (Section VII-F).

A. Performance of Sequential Hypothesis Test with Training

We now investigate the performance of our sequential hypothesis test with training. The Ethernet and WLAN inter-ACK time distributions required by this algorithm are obtained as described in Section VI-B. We next describe results from offline and online evaluation.

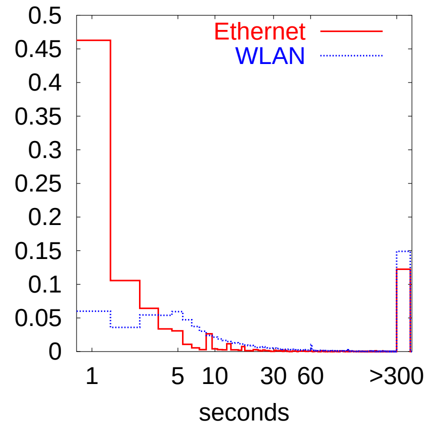

Fig. 7. Detection-time distributions for the trace collected on $10/20/2006$ ($T = 240\ \mu s$, $K = 10^6, N = 100$).

1) Offline Evaluation: In offline evaluation, we collect measurements on three consecutive days, from $10/18/2006$ to $10/20/2006$. The trace on each day lasts for 6 to 7 hours. The threshold to identify ACK-pairs, T, is set to $240\mu s$ or $400\ \mu s$. The threshold to decide a host's connection type, K, is set to $10^4, 10^5$ or 10^6. We next describe the results for the trace collected on $10/20/2006$; the results for the other two days are similar.

Tables I and II present the detection results for the campus public WLAN and the Ethernet test set respectively. In both cases we observe that the detection results are similar under different values of T and K, indicating that our algorithm is insensitive to the choice of parameters. For all values of T and K, the detection results are extremely accurate with a correct detection ratio above 99.38%. On average, it takes less than 10 ACK-pairs (corresponding to 250 to 347 data packets) to make a detection for WLAN and less than 20 ACK-pairs (corresponding to 87 to 124 data packets) for Ethernet. The relatively larger number of data packets for a detection of WLAN compared to that of Ethernet can be explained as follows. The inter-ACK times in WLAN tend to be large (compared to those in Ethernet), leading to large inter-arrival times between newly triggered data packets due to TCP's self-clocking. When the inter-arrival time of the data packets is larger than the threshold T, the corresponding ACKs are not qualified as an ACK-pair. This is confirmed by the lower ACK-pair ratio (i.e., the number of ACK-pairs divided by the total number of packets) in WLAN traffic shown in Tables I and II.

The detection-time distributions for both WLAN and LAN when $K = 10^6$ is shown in Fig. 7. The median detection times for Ethernet and WLAN are around 1 second and 10 seconds respectively. The much shorter detection time in Ethernet is due to higher ACK-pair ratios, as explained earlier. We also observe long detection times (over 5 minutes) in the figure. They might be caused by users' change of activities (e.g., a user stops using the computer to think or talk and then resume using it).

Finally, around 84% of ACK-pairs used in WLAN detection and 89% of ACK-pairs used in LAN detection are generated by web traffic, indicating that our approach is effective even for short flows.

2) Online Evaluation: In online evaluation, we run our detection algorithm online on three consecutive days, from $10/25/2006$ to $10/27/2006$, lasting for 6 to 7 hours on each day. We set $T = 240\ \mu s, K = 10^6$, representing a conservative selection of parameters. Table III presents the detection results for both test sets. We observe consistent results as those in offline evaluation. That is, the detection is highly accurate and prompt. The average numbers of ACK-pairs and data packets required for a detection are consistent with those in the offline evaluation. The above demonstrates the efficiency of our online detection algorithm.

TABLE III

ONLINE EVALUATION OF SEQUENTIAL HYPOTHESIS TEST WITH TRAINING ($10/25/2006$ - $10/27/2006$).

	$10/25/2006$		$10/26/2006$		$10/27/2006$	
	WLAN	Ethernet	WLAN	Ethernet	WLAN	Ethernet
Avg. # of ACK-pairs for a detection	7	16	8	21	7	16
Avg. # of data pkts for a detection	310	145	351	153	336	135
Median detection time (sec)	9.7	1.2	15.0	0.1	11.4	1.2
Number of detections	$23,266$	$5,798$	$15,977$	$15,654$	$10,628$	$2,948$
Correct detection ratio	99.58%	99.93%	98.44%	99.92%	99.72%	99.76%
ACK-pair ratio	2%	11%	2%	13%	2%	12%

TABLE IV

EVALUATION OF SEQUENTIAL HYPOTHESIS TEST WITHOUT TRAINING ON WLANS.

Date	10/18/2006	10/19/2006	10/20/2006
Detection ratio	68%	76%	60%
Avg. # of ACK-pairs for a detection	22	21	19
Avg. # of data pkts for a detection	997	858	903
Median detection time (sec)	105	59	52
Number of detections	$3,259$	$6,539$	$2,722$

B. Performance of Sequential Hypothesis Test without Training

We now examine the performance of our sequential hypothesis test without training. Recall that this algorithm does not require training sets. It takes at most N ACK-pairs to make a decision (i.e., detecting WLAN or undetermined). We apply this algorithm to traces collected between $10/18/2006$ and $10/20/2006$ using $T = 240\ \mu s$, $K = 10^6$, and $N = 100$. For the Ethernet test set, this algorithm detects no WLAN host for all the traces, indicating that it has no false positives. Note that although this algorithm is derived using analytical results in Section III (in a setting where the receiver is one hop away from the router), our experimental results indicate that it is accurate in more relaxed settings (the Ethernet hosts in the Computer Science building are three hops away from the gateway router). This is not surprising since our algorithm is based on an extremely conservative analysis (assuming that the single Ethernet link is full utilized). For the WLAN test set, of all the hosts with at least one ACK-pair, this algorithm detects 60% to 76% of them as WLAN hosts. Table IV presents the experimental results for the WLAN test set. In general, this algorithm requires more ACK-pairs and longer time to make a detection than the algorithm with training.

C. Scalability Study

We investigate the scalability of our approach by looking at its CPU and memory usages of the PC that runs the detection algorithms (the configuration of the PC is described in Section VI-A). During online evaluation, we sample the CPU usage at the measurement PC every 30 seconds. The maximum CPU usage is 9.1% (without optimizing our implementation), indicating that the measurement task is well within the capability of the measurement PC. For memory usage, we investigate the space taken by the unacked-data-packet queues since the memory usage mainly comes from storing these queues (see Section V). Fig. 8 plots the CDF of the maximum number of items in each queue for the trace collected on $10/20/2006$ (results for other traces are similar). This trace was collected over 7 hours and captures 1.8 million TCP flows for the IP addresses being monitored (the maximum number of concurrent flows is 8244). We observe that most of the queues are very short: 90% of them have less than 3 items, indicating that the memory usage is very low (each data item only keeps 14 bytes of data; see Section V-A). However, we also observe very long queues. We conjecture that these long queues are due to routing changes or abnormal behaviors

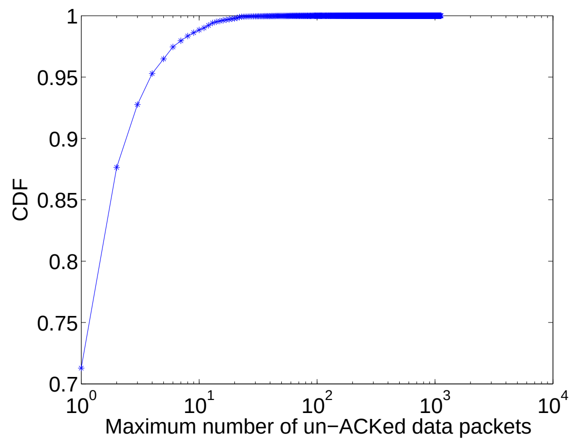

Fig. 8. CDF of the number of items in the unacked-data-packet queues.

in the routes. As an optimization to our online detection system, we can discard unacked-data-packet queues longer than a certain threshold.

D. Detection of Wireless Networks behind NAT

We now demonstrate that our approach is equally applicable to detect other types of rouges, in particular, wireless networks behind a NAT box. Note that, schemes using MAC address (e.g., [9], [4], [10]) fail to detect this type of rogue, since all traffic going through a NAT box have the same MAC address (i.e., the MAC address of the NAT box). We look at NAT boxes in two settings, one configured by ourselves and the other being used in the Computer Science Department.

1) Self-configured NAT: We configure a Linux host A as a NAT box. Host A has two network interfaces, an Ethernet card and a ZCOMAX AirRunner/XI-300 802.11b wireless card. The Ethernet interface connects directly to the Internet. The wireless card is configured to the master mode using Host AP [5] so that it acts as an AP. We then set up two laptops B and C to access the Internet through the wireless card of A. When host B or C accesses the Internet, its packets reach host A. Host A then translates the addresses of the packets and forwards the packets to the Internet through its Ethernet card.

We conduct an experiment on $10/26/2006$. The experiment lasts for about two minutes. We observed 163 ACK-pairs. Among them, 92% of the ACK-pairs are from web traffic via port 80. The remaining ACK-pairs are from port 1935, which is used by Macromedia Flash Communication Server MX for the RTMP (Real-Time Messaging Protocol). The sequential hypothesis test with training makes 37 online detections, all as WLAN host. On average, one detection is made in every 4 ACK-pairs. The above results demonstrate that our test can effectively detect wireless networks behind NAT boxes.

2) NATs in the Computer Science Department: Two NAT boxes in the Computer Science Department provide a free local network to users in the department. A host may use either Ethernet or WLAN to connect to a NAT box. All traffic through a NAT box will have the IP address of the NAT box. We monitor the IP addresses of these two NAT boxes. Our offline detection (from $10/18/2006$ to $10/20/2006$) and online detection (from $10/25/2006$ to $10/27/2006$) both indicate a mixture of WLAN and Ethernet connections. The ACK-pair ratios are higher than that of WLAN and lower than that of Ethernet hosts, which are consistent with the setting that these two NAT boxes provide both WLAN and Ethernet connections.

E. Detection of Connection-type Switchings

Next we explore a scenario where an end host may switch between a wired and wireless connection. Our goal is to examine whether our detection approach can accurately report the connection switchings. We use an IBM laptop with both 100 Mbps Ethernet and 54 Mbps 802.11g WLAN connections. This laptop uses a web crawler to download the first 200 web files from cnn.com (8.3 Mbytes of data) using Ethernet, and then switches to WLAN to download the first 200 web files from nytimes.com (6.5 Mbytes of data). This process is repeated for three times. We run the sequential hypothesis test with training using $T = 240\mu s, K = 10^6$ and $N = 100$. Our algorithm makes 284 detections, 283 correct and one incorrect. The correct detection ratio is 99.65%. This demonstrates that our approach is effective in detecting connection-type switchings. Therefore, if a host switches between using Ethernet and WLAN provided by its rogue AP, our approach can effectively detect this rogue AP.

F. Detection under High CPU, Disk or Network Utilization

We now investigate whether the performance of our approach will be affected when an end host has very high CPU, disk or network utilization. For this purpose, we stress either the CPU, disk or network connection of an end host, while downloading the first 200 web files from cnn.com using a web crawler at the host. For each scenario, we conduct experiments for both Ethernet and WLAN connections and detect the connection type using sequential hypothesis test with training. All experiments are conducted on an IBM laptop with both a 100 Mbps Ethernet and a 54 Mbps 802.11g WLAN connection card.

We stress the CPU (utilization reaching 100%) by running an infinite loop. For the Ethernet connection, we observe 1077 ACK-pairs and 53 detections. For the WLAN connection, we observe 921 ACK-pairs and 123 detections. All the detections are correct. We stress the hard disk by running a virus scanning program that scans the disk. For the Ethernet connection, we observe 1158 ACK-pairs and 57 detections. For the WLAN connection, we observe 872 ACK-pairs and 84 detections. Again, all the detections are correct.

To stress the network connection, we conduct two sets of experiments, one stressing the downlink direction by downloading a large file from the local network; the other stressing the uplink direction by uploading a large file to the local network. Note that both cases only generate traffic in the local network, not captured at the monitoring point, and hence does not interfere with data monitoring. When stressing the downlink, we observe 848 ACK-pairs and 42 detections for the Ethernet connection; 660 ACK-pairs and 72 detections for the WLAN connection. When stressing the uplink, we observe 438 ACK-pairs and 21 detections for the Ethernet connection; 487 ACK-pairs and 46 detections for the WLAN connection. All the detections are correct. We observe that while stressing the downlink or the uplink, the number of ACK-pairs is significantly smaller than that when stressing CPU or disk. This is due to cross traffic generated by the local downloading or uploading activities. We also observe that the number of ACK-pairs is less when stressing the uplink than that when stressing the downlink. This is because the uploading data packets may be inserted between ACKs and lead to less ACK-pairs.

In summary, the above results indicate that our detection approach is effective even when hosts have high CPU, hard disk or network utilization.

VIII. Discussions

We next discuss several issues related to rogue AP detection.

A. Locating Rogue APs

Our approach to detecting a rogue AP also helps to locate the rogue AP. Let us consider a common scenario in which a WLAN host is connected to a rogue AP, which is connected to an access router via one or multiple switches. In this scenario, the rogue AP can be located using the following steps. First, a network manager detects the IP address of the WLAN host at the monitoring point, and then locates the access router of this host based on the host's IP address and the subnet addressing structure. From the ARP table at the access router (which stores the mapping between an IP address and its corresponding MAC address), the network manager further determines the MAC address of the WLAN host. Afterwards, the network manager uses the identified MAC address to obtain its corresponding switch port by SNMP querying the first downstream switch connected to the access router (this is through the switch table at the switch, which stores the mapping between a MAC address and a switch port). Last, the network manager sequentially queries downstream switches (if any) to locate the switch port (and hence the physical location) of the rogue AP.

B. Rogues by Authorized Users

Our scheme can easily detect rogue APs installed by hosts not authorized to use WLAN. We next discuss the case that rouges are installed by hosts authorized to use WLAN. We consider two types of local networks: purely wireless networks (i.e., all IP addresses are allowed to use wireless connections) and mixed networks (i.e., networks supporting both Ethernet and wireless connections).

Purely wireless networks. In such a network, a wireless host A may set up another wireless card as a rogue AP for an illegitimate host B (as described in Section VII-D). In this case, packets from B will have the IP address of A, which is an authorized WLAN address. Therefore, our scheme does not detect this type of rogue directly. However, since host B connects to the Internet through two wireless hops, its traffic characteristics will differ from those through a single wireless hop and those through Ethernet, and hence can be detected through traffic analysis. An accurate detection scheme for this type of rogue is left as future work.

Mixed networks. In such a network, we consider two scenarios. In the first scenario, the IP address blocks for Ethernet and WLAN connections do not overlap. Then a host will have different IP addresses for its Ethernet and WLAN connections. In this scenario, if a host authorized to use both Ethernet and WLAN installs a rogue AP on its Ethernet connection, the host obtains an IP address in the Ethernet block and the associated rogue AP will be easily detected by our scheme. If the host uses its authorized WLAN connection to connect to the Internet and sets up another wireless card as a rogue AP for an illegitimate host, then this illegitimate host connects to the Internet via two wireless hops, and can be detected through traffic analysis (as described for purely wireless networks.)

In the second scenario, the IP address blocks for Ethernet and WLAN connections overlap. Then a host may maintain the same IP address for both Ethernet and WLAN connections. Similar to the first scenario, we can detect rogue APs that provide hosts Internet connection using two wireless hops through traffic analysis. However, a host authorized to use WLAN may also set up a rogue AP on its Ethernet connection for itself to connect to the Internet. This type of rogue cannot be detected by our scheme or traffic analysis (since this host only use a single wireless hop). However, in this case, our scheme can be combined with RF-sensing schemes so that only hosts in the authorization list need to be closely monitored by RF sensing.

The above discussions imply that, to achieve tighter security, it is better to use separate IP blocks for Ethernet and WLAN connections.

C. Possible attacks to our approach

Our approach is based on inter-ACK times. It is effective for the common scenario where a rogue AP is installed by an innocent user (for convenience or flexibility). It is also robust against MAC-address spoofing attacks. However, a rogue AP may change the inter-ACK times to elude being detected by our algorithms. For instance, it may reduce the inter-ACK times by buffering ACKs first and then releasing them in a batch in order to disguise the traffic as Ethernet traffic. Such a camouflage, however, will inevitably increase local RTTs (i.e., the portion of RTT inside the WLAN). Therefore, we may combine inter-ACK time and local RTT measurement to detect such a camouflage. An effective scheme is left as future work.

IX. Conclusions

In this paper, we have proposed two online algorithms to detect rogue access points, based on real time passive measurements collected at a gateway router. Both algorithms exploit the fundamental properties of the 802.11 CSMA/CA MAC protocol and the half duplex nature of wireless channels to differentiate Ethernet and WLAN TCP traffic. Central to both algorithms are sequential hypothesis tests that determine a host's connection type in real time by extending our earlier TCP ACK-pair techniques [25]. One algorithm requires training sets, while the other does not. Extensive experiments in various scenarios and over hosts with various operating systems have demonstrated the excellent performance of our approach: the algorithm that requires training provides rapid detection and is extremely accurate; the algorithm that does not require training detects 60%-76% of the wireless hosts without any false positives; both algorithms require computation and storage well within the capability of commodity equipment. Furthermore, our scheme can detect connection switchings and wireless networks behind a NAT box. Last, our scheme remains effective for hosts with high CPU, hard disk or network utilizations.

Acknowledgement

This research was supported in part by the National Science Foundation under NSF grants ANI-0325868, ANI-0240487, ANI-0085848, CNS-0519998, CNS-0519922, CNS-0524323, and EIA-0080119. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies. We would like to thank Prof. Richard S. Ellis (UMass, Amherst), Prof. Guanling Chen (UMass, Lowell), and Dr. Sharad Jaiswal (Bell lab, India) for helpful discussions. Last, we wish to thank Rick Tuthill from the Office of Information Technology at UMass Amherst, for helping us understand the UMass network architecture, and in the installation and management of the monitoring equipment.

References

[1] *AirDefense, Wireless LAN Security*. http://airdefense.net.

[2] *AirMagnet*. http://www.airmagnet.com.

[3] *AirWave, AirWave Management Platform*. http://airwave.com.

[4] *Cisco Wireless LAN Solution Engine (WLSE)*. http://www.cisco.com/en/US/products/sw/cscowork /ps3915/.

[5] *Host AP*. http://hostap.epitest.fi.

[6] http://www.endace.com.

[7] Microsoft Windows 2000 TCP/IP implementation details, http://www.microsoft.com/technet/itsolu tions/network/deploy/depovg/tcpip2k .mspx.

[8] *NetStumbler*. http://www.netstumbler.com.

[9] *Rogue Access Point Detection: Automatically Detect and Manage Wireless Threats to Your Network*. http://www.proxim.com.

[10] A. Adya, V. Bahl, R. Chandra, and L. Qiu. Architecture and techniques for diagnosing faults in ieee 802.11 infrastructure networks. In *Proc. ACM MOBICOM*, September 2004.

[11] P. Bahl, R. Chandra, J. Padhye, L. Ravindranath, M. Singh, A. Wolman, and B. Zill. Enhancing the security of corporate Wi-Fi networks using DAIR. In *Proc. ACM MOBISYS*, 2006.

[12] V. Baiamonte, K. Papagiannaki, and G. Iannaccone. Detecting 802.11 wireless hosts from remote passive observations. In *Proc. IFIP/TC6 Networking*, Atlanta, GE, May 2007.

[13] R. Beyah, S. Kangude, G. Yu, B. Strickland, and J. Copeland. Rogue access point detection using temporal traffic characteristics. In *Proc. IEEE GLOBECOM*, Dec 2004.

[14] G. Casella and R. L. Berger. *Statistical Inference*. Duxbury Thomson Learning, 2002.

[15] L. Cheng and I. Marsic. Fuzzy reasoning for wireless awareness. *International Journal of Wireless Information Networks*, 8(1), 2001.

[16] S. Garg, M. Kappes, and A. S. Krishnakumar. On the effect of contention-window sizes in IEEE 802.11b networks. Technical Report ALR-2002-024, Avaya Labs Research, 2002.

[17] IEEE 802.11, 802.11a, 802.11b standards for wireless local area networks. http://standards.ieee.org/getieee802/802.11.html.

[18] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Proc. IEEE Symposium on Security and Privacy*, May 2004.

[19] C. Mano, A. Blaich, Q. Liao, Y. Jiang, D. Salyers, D. Cieslak, and A. Striegel. RIPPS: Rogue identifying packet payload slicer detecting unauthorized wireless hosts through network traffic conditioning. *ACM Transactions on Information Systems and Security*, to appear.

[20] Packet trace analysis.
http://ipmon.sprintlabs.com/packstat/packetoverview.php.

[21] P. Sarolahti and A. Kuznetsov. Congestion control in Linux TCP. In *Proc. USENIX02*, June 2002.

[22] A. N. Shiryaev. *Probability*. Springer, 2nd edition.

[23] K. Thompson, G. Miller, and R. Wilder. Wide-area Internet traffic patterns and characteristics. *IEEE Network*, 11(6):10–23, Nov./Dec. 1997.

[24] A. Wald. *Sequential Analysis*. J. Wiley & Sons, 1947.

[25] W. Wei, S. Jaiswal, J. Kurose, and D. Towsley. Identifying 802.11 traffic from passive measurements using iterative Bayesian inference. In *Proc. IEEE INFOCOM*, 2006.

[26] W. Wei, B. Wang, C. Zhang, J. Kurose, and D. Towsley. Classification of access network types: Ethernet, wireless LAN, ADSL, cable modem or dialup? In *Proc. IEEE INFOCOM*, March 2005.

[27] H. Yin, G. Chen, and J. Wang. Detecting Protected Layer-3 Rogue APs. In *Proceedings of the Fourth IEEE International Conference on Broadband Communications, Networks, and Systems (BROADNETS)*, Raleigh, NC, September 2007.

APPENDIX I
PROOF OF THEOREM 1

In the Ethernet setting, we ignore the transmission time of an ACK since it is negligible. For convenience, we introduce a *time unit* of 30 μs. Measurement studies show that the average packet size on the Internet is between 300 and 400 bytes [23], [20]. For ease of calculation, we assume that all cross-traffic packets are 375 bytes. Then the transmission time of a cross-traffic packet on a 100 Mbps link is 1 time unit.

Recall that Δ_A denotes the inter-ACK time of ACKs A_1 and A_3. We discretize Δ_A using the time unit and denote the discretized value as I_A, that is, $I_A = \lfloor \Delta_A/30 \rfloor$. Let Δ_D denote the inter-departure time of packets P_1 and P_3 at queue Q_D (i.e., the queue at the router in the direction of data packets). Similarly, we discretize Δ_D and denote the discretized value as I_D, that is, $I_D = \lfloor \Delta_D/30 \rfloor$. We next state three lemmas that are used to prove Theorem 1.

Lemma 1: Let $Z = I_D - 8$. When $\rho_D = 1$, Z follows a Poisson distribution with the mean of 8 time units.

Proof: One component of I_D is the transmission time of packets P_1 and P_2 at queue Q_D, which is $2\times 120/30 = 8$ time units (since P_i is 1500 bytes and the bandwidth of the link between the router and the receiver is 100 Mbps). The other component of I_D is the (discretized) transmission time of the cross-traffic packets that arrive between P_1 and P_3 at Q_D, denoted as Z. Then $Z = I_D - 8$. By the $M/D/1$ queue assumption, Z follows a Poisson distribution. Furthermore, since the inter-arrival time of P_1 and P_3 at Q_D is $2\times 120/30 = 8$ time units (since P_i is 1500 bytes and the bandwidth of the link between the source and the router is 100 Mbps), on average, 8 cross-traffic packets arrive between P_1 and P_3 at Q_D. This is because, given $\rho_D = 1$, the arrival rate of cross-traffic packets at Q_D is 1 packet per time unit, equal to the processing rate. Therefore, the mean of Z is 8 time units. ■

Lemma 2: Suppose $I_D = x$ time units. When $\rho_A = 1$, the conditional distribution of I_A given I_D follows a Poisson distribution with the mean of x time units.

Proof: From Fig. 2(a), I_A is the same as the inter-departure time of ACKs A_1 and A_3 at queue Q_A. Since we assume no other traffic between the router and the receiver, the inter-arrival time of A_1 and A_3 at queue Q_A is the same as I_D. Therefore, given that $I_D = x$ time units, the number of cross-traffic packets arriving between A_1 and A_3 at queue Q_A follows a Poisson distribution with the mean of x time units (following a reasoning similar to the proof for Lemma 1). Therefore, the conditional distribution of I_A given $I_D = x$ follows a Poisson distribution with the mean of x time units. ■

Lemma 3: When $\rho_D = \rho_A = 1$,

$$P(I_A \leq x) = \sum_{y=8}^{\infty} \frac{8^{y-8} e^{-8}}{(y-8)!} \sum_{i=0}^{x} \frac{y^i e^{-y}}{i!}$$

Proof: This follows directly from Lemmas 1 and 2. ■

We now proceed to prove Theorem 1.

Proof: We first prove the theorem when $\rho_D = \rho_A = 1$. Under this condition, from Lemma 3, by direct calculation, we have $P(I_A > 20) = P(\Delta_A > 600\ \mu s) < 0.18$.

We next prove that the theorem also holds when $0 < \rho_D < 1$ or $0 < \rho_A < 1$. When $0 < \rho_D < 1$, the inter-departure time of data packets P_1 and P_3 at queue Q_D is no more than that when $\rho_D = 1$. Similarly, when $0 < \rho_A < 1$, the inter-departure time of ACKs A_1 and A_3 at queue Q_A is no more than that when $\rho_A = 1$. Therefore, $P(\Delta_A > 600\ \mu s) < 0.18$ also holds when $0 < \rho_D < 1$ or $0 < \rho_A < 1$. ■

APPENDIX II
PROOF OF THEOREM 2

We first present a lemma that is used to prove Theorem 2.

Lemma 4: Let $g(n,q) = \sum_{i=\lfloor (n+1)/2 \rfloor}^{n} \binom{n}{i} q^i (1-q)^{n-i}$. Then $g(n,q)$ is an increasing function of q, where $0 \leq q \leq 1$. Furthermore, $\lim_{n\to\infty} g_q(n) = 1$ for $q > 1/2$.

Proof: We first prove the monotonicity of the function $g(n,q)$ with respect to q.

$$\begin{aligned}
\frac{\partial g(n,q)}{\partial q} &= \sum_{i=\lfloor (n+1)/2 \rfloor}^{n} \frac{n!}{i!(n-i)!} i q^{i-1} (1-q)^{n-i} \\
&\quad - \sum_{i=\lfloor (n+1)/2 \rfloor}^{n-1} \frac{n!}{i!(n-i)!} (n-i) q^i (1-q)^{n-i-1} \\
&= \sum_{i=\lfloor (n+1)/2 \rfloor}^{n} \frac{n!}{(i-1)!(n-i)!} q^{i-1} (1-q)^{n-i} \\
&\quad - \sum_{i=\lfloor (n+1)/2 \rfloor}^{n-1} \frac{n!}{i!(n-i-1)!} q^i (1-q)^{n-i-1} \\
&= \sum_{j=\lfloor (n+1)/2 \rfloor - 1}^{n-1} \frac{n!}{j!(n-j-1)!} q^j (1-q)^{n-j-1} \\
&\quad - \sum_{i=\lfloor (n+1)/2 \rfloor}^{n-1} \frac{n!}{i!(n-i-1)!} q^i (1-q)^{n-i-1} \\
&= \frac{n! q^{\lfloor (n+1)/2 \rfloor - 1} (1-q)^{n - \lfloor (n+1)/2 \rfloor}}{(\lfloor (n+1)/2 \rfloor - 1)!(n - \lfloor (n+1)/2 \rfloor)!} \geq 0
\end{aligned}$$

Hence $g(n,q)$ is an increasing function of q, $0 \leq q \leq 1$.

We now prove the second part of the lemma. Assume that $\{X_i\}$ is a set of i.i.d Bernoulli random variables with $P(X_i = 1) = q$. By the definition of a binomial distribution,

$$g_q(n) = P\Big(\frac{\sum_{i=1}^n X_i}{\lfloor (n+1)/2 \rfloor} \geq 1\Big).$$

We have

$$\frac{\sum_{i=1}^n X_i}{(n/2)+1} \leq \frac{\sum_{i=1}^n X_i}{\lfloor (n+1)/2 \rfloor} \leq \frac{\sum_{i=1}^n X_i}{n/2} \quad \forall n.$$

By the strong law of large numbers, we also have

$$\lim_{n\to\infty} \frac{\sum_{i=1}^n X_i}{(n/2)+1} = \lim_{n\to\infty} \frac{\sum_{i=1}^n X_i}{n/2} = 2q \quad a.e.$$

Therefore,

$$\lim_{n\to\infty} \frac{\sum_{i=1}^n X_i}{\lfloor (n+1)/2 \rfloor} = 2q \quad a.e.$$

Since almost sure convergence implies convergence in probability [22], we have

$$\lim_{n\to\infty} P\Big(\Big|\frac{\sum_{i=1}^n X_i}{\lfloor (n+1)/2 \rfloor} - 2q\Big| \geq \epsilon\Big) = 0 \quad \forall \epsilon > 0,$$

which is equivalent to

$$\lim_{n\to\infty} P\Big(\frac{\sum_{i=1}^n X_i}{\lfloor (n+1)/2 \rfloor} \in (2q-\epsilon, 2q+\epsilon)\Big) = 1 \quad \forall \epsilon > 0.$$

Since for $q > 1/2$ and $0 < \epsilon < 2q - 1$, we have

$$\begin{aligned} 1 \geq g_q(n) &= P\Big(\frac{\sum_{i=1}^n X_i}{\lfloor (n+1)/2 \rfloor} \geq 1\Big) \\ &\geq P\Big(\frac{\sum_{i=1}^n X_i}{\lfloor (n+1)/2 \rfloor} \in (2q-\epsilon, 2q+\epsilon)\Big). \end{aligned}$$

It follows that $\lim_{n\to\infty} g_q(n) = 1$ for $q > 1/2$. ■

We now prove Theorem 2. Let $\Delta_A^{(1)}, \ldots, \Delta_A^{(n)}$ be the ordered statistic of $\Delta_1^A, \ldots, \Delta_n^A$ in the ascending order. For simplicity, we use $\xi_{.5}^n(\Delta_A) = \Delta_A^{(\lfloor (n+1)/2 \rfloor)}$ regardless n being even or odd.

Proof: Let $u = P(\Delta_A \leq 600\ \mu s)$.

$$\begin{aligned} P(\xi_{.5}^n(\Delta_A) \leq 600\ \mu s) &= \sum_{i=\lfloor (n+1)/2 \rfloor}^{n} \binom{n}{i} u^i (1-u)^{n-i} \\ &= g(n,u), \end{aligned}$$

where $g(n,u)$ is as defined in Lemma 4. By Lemma 4, $g(n,q)$ is an increasing function of q for $0 \leq q \leq 1$. By Theorem 1, we know $u > 1 - 0.18 = 0.82$. Therefore, we have $g(n,u) \geq g(n, 0.82)$. Hence, $P(\xi_{.5}^n(\Delta_A) \leq 600\ \mu s) \geq g(n, 0.82)$. By direct calculation, we have $P(\xi_{.5}^n(\Delta_A) \leq 600\ \mu s) \approx 1$ for $43 \leq n \leq 100$. Furthermore, since $0.82 > 1/2$, by Lemma 4, we have $\lim_{n\to\infty} P(\xi_{.5}^n(\Delta_A) \leq 600\ \mu s) = 1$. ■

APPENDIX III
PROOF OF THEOREM 3

Before proving Theorem 3, we first state a lemma that is used in the proof.

Lemma 5: Let $\Delta^D_{i,i+1}$ represent the inter-arrival time of data packets P_i and P_{i+1} at the AP, $i = 1, 2, 3$. Then $P(\Delta^D_{i,i+1} < 1570\ \mu s) \approx 1$, $P(\Delta^D_{i,i+1} < 325\ \mu s) \geq 0.89$.

Proof: Let $I^D_{i,i+1}$ be the discretized value of $\Delta^D_{i,i+1}$, i.e., $I^D_{i,i+1} = \lfloor \Delta^D_{i,i+1}/30 \rfloor$. When $\rho_D = 1$, similar to the proof of Lemma 1, we can show that $I^D_{i,i+1}$ follows a Poisson distribution with the parameter of 4 time units. Then $P(\Delta^D_{i,i+1} < 1570\mu s) > P(I^D_{i,i+1} = 52) = \sum_{x=4}^{52} \frac{4^{x-4}e^{-4}}{(x-4)!} \approx 1$. When $\rho_D < 1$, the value of $\Delta^D_{i,i+1}$ is less than that when $\rho_D = 1$, and hence $P(\Delta^D_{i,i+1} < 1570\mu s) \approx 1$ also holds. Similarly, we obtain $P(\Delta^D_{i,i+1} < 325\ \mu s) \geq 0.89$. ■

We now prove Theorem 3.

Proof: Let C denote the condition that $\Delta^D_{i,i+1} \leq 1570\ \mu s$, $i = 1, 2, 3$. Under this condition, P_{i+1} arrives at the AP before the AP finishes transmitting P_i, since the MAC service time of a data packet is at least $1570\ \mu s$ in 11 Mbps 802.11b. Assuming independence, we have
$P(C) = \prod_{i=1}^{3} P(\Delta^D_{i,i+1} \leq 1570\ \mu s)$. From Lemma 5, $P(C) \approx 1$. Let $\bar{C}$ denote the complementary condition of C. Then

$$\begin{aligned} & P(\Delta_A > 600\ \mu s) \\ = \ & P(\Delta_A > 600\ \mu s \mid C)P(C) + P(\Delta_A > 600\ \mu s \mid \bar{C})P(\bar{C}) \\ \geq \ & P(\Delta_A > 600\ \mu s \mid C)P(C) \approx P(\Delta_A > 600\ \mu s \mid C) \end{aligned}$$

We now derive $P(\Delta_A \leq 600\ \mu s \mid C)$. To satisfy $\Delta_A \leq 600\ \mu s$, no data packet can be transmitted between A_1 and A_3, since the transmission time of a data packet is at least $1570\ \mu s$. Therefore, only the following two sequences are possible: $P_2P_3A_1A_3P_4$ and $P_2P_3P_4A_1A_3$.

We first derive the probability that the first sequence occurs given condition C. Since P_2 arrives at the AP before the AP finishes transmitting P_1, the receiver and the AP contend for the wireless channel: the receiver needs to transmit ACK A_1 (which is generated corresponding to packet P_1) while the AP needs to transmit packet P_2. Let ϕ denote the probability that A_1 obtains the channel earlier than P_2. Since this probability can be affected by many factors (e.g., the timing when A_1 reaches the MAC layer, when packet P_2 can be transmitted), we assume ϕ can take any value in $[0, 1]$. When P_2 transmits earlier than A_1, A_1 will contend with packet P_3 for the wireless channel. In this case, we assume that A_1 and P_3 are equally likely to win the contention, since they can both be transmitted immediately. To summarize, the probability that P_2 and P_3 are earlier than A_1 is $(1-\phi) \times 1/2$, the probability that A_1 and A_3 are earlier than P_4 is $1/2 \times \phi$ (for similar reasons as described earlier). Therefore, the probability that the first sequence occurs given C is $(1-\phi) \times 1/2 \times 1/2 \times \phi = \phi(1-\phi)/4$.

For the second sequence, the probability of having P_2 and P_3 earlier than A_1 is $(1-\phi) \times 1/2$; the probability that P_4 is earlier than A_1 is $1/2$. Therefore, the probability that the second sequence occurs is $(1-\phi) \times 1/2 \times 1/2 = (1-\phi)/4$.

In both sequences, to satisfy $\Delta_A \leq 600\ \mu s$, we also require the MAC service time of A_3 to be less than 600 μs. The probability of this condition being satisfied is $(600 - 508)/620 = 92/620$. Therefore,

$$\begin{aligned} P(\Delta_A \leq 600\ \mu s \mid C) &= [\phi(1-\phi)/4 + (1-\phi)/4]92/620 \\ &= \frac{1}{4}(1-\phi^2)\frac{92}{620} < 0.04. \end{aligned}$$

Hence, $P(\Delta_A > 600\ \mu s) \geq P(\Delta_A > 600\ \mu s \mid C) > 1 - 0.04 > 0.96$. ■

APPENDIX IV
PROOF OF THEOREM 4

Proof: The proof is similar to that of Theorem 3. Let C denote the condition that $\Delta^D_{i,i+1} \leq 325\ \mu s$, $i = 1, 2, 3$. Under this condition, P_{i+1} arrives at the AP before the AP finishes transmitting P_i, since the MAC service time of a data packet is at least $325\ \mu s$ in 54 Mbps 802.11g. Then assuming independence and from Lemma 5, $P(C) \geq 0.89^3$.

We now obtain $P(\Delta_A \leq 600\ \mu s \mid C)$. To satisfy $\Delta_A \leq 600\ \mu s$, there can be at most one data packet transmitted between ACKs A_1 and A_3, since the minimum transmission time of two data packets and one ACK exceeds 600 μs. This constraint leads to the following four possible sequences: $P_2P_3A_1A_3P_4$, $P_2P_3P_4A_1A_3$, $P_2A_1P_3A_3P_4$, and $P_2P_3A_1P_4A_3$. The first two sequences are the same as those in the proof of Theorem 3. They occur with respectively the probabilities of $\phi(1-\phi)/4$ and $(1-\phi)/4$, where ϕ is the probability that ACK A_1 transmits earlier than P_2. Following a similar reasoning as that in the proof of Theorem 3, the probability that the third sequence occurs is $(1-\phi) \times 1/2 \times 1 \times \phi = \phi(1-\phi)/2$, and the probability that the last sequence occurs is $(1-\phi) \times 1/2 \times 1/2 \times 1/2 = (1-\phi)/8$.

For the first two sequences, we have $\Delta_A \leq 600\ \mu s$. For the third sequence, to satisfy $\Delta_A \leq 600\ \mu s$, we need the total MAC service time of P_3 and A_3 to be below $600\ \mu s$. Similarly, for the fourth sequence, to satisfy $\Delta_A \leq 600\ \mu s$, we need the total MAC service time of P_4 and A_3 to be below $600\ \mu s$. Let X and Y denote respectively the MAC service time of a data packet and an ACK. Then for both the third and the fourth sequences, we need $X + Y \leq 600\ \mu s$. Let $\alpha = P(X + Y \leq 600\ \mu s)$. As described in Section III-D, X and Y are uniformly distributed in $[325, 460]\ \mu s$ and $[109, 244]\ \mu s$, respectively. Then, by a standard technique, we have

$$\alpha = 1 - \frac{(244-140) \times (460-356)/2}{(460-325) \times (244-109)} = 0.70.$$

Hence,

$$\begin{aligned} & P(\Delta_A \leq 600\ \mu s \mid C) \\ = \ & \phi(1-\phi)/4 + (1-\phi)/4 + \alpha\phi(1-\phi)/2 + \alpha(1-\phi)/8 \\ = \ & (-87718\phi^2 + 38451\phi + 49267)/145800 \leq 0.37. \end{aligned}$$

Therefore, $P(\Delta_A > 600\ \mu s) \geq P(\Delta_A > 600\ \mu s \mid C)P(C) > (1-0.37) \times 0.89^3 = 0.45$. ■