Passive Online Detection of 802.11 Traffic Using Sequential Hypothesis Testing with TCP ACK-Pairs

Wei Wei, *Member*, *IEEE*, Kyoungwon Suh, *Member*, *IEEE*, Bing Wang, *Member*, *IEEE*, Yu Gu, *Member*, *IEEE*, Jim Kurose, *Fellow*, *IEEE*, Don Towsley, *Fellow*, *IEEE*, and Sharad Jaiswal

Abstract—In this paper, we propose two online algorithms to detect 802.11 traffic from packet-header data collected passively at a monitoring point. These algorithms have a number of applications in *real-time* wireless LAN management, for instance, in detecting unauthorized access points and detecting/predicting performance degradations. Both algorithms use sequential hypothesis tests and exploit fundamental properties of the 802.11 CSMA/CA MAC protocol and the half-duplex nature of wireless channels. They differ in that one requires training sets, while the other does not. We have built a system for online wireless traffic detection using these algorithms and deployed it at a university gateway router. Extensive experiments have demonstrated the effectiveness 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 percent to 76 percent of the wireless hosts without any false positives, and both algorithms are lightweight, with computation and storage overhead well within the capability of commodity equipment.

Index Terms—Wireless LAN management, wireless traffic detection, sequential hypothesis testing, TCP ACK-pairs.

1 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. Several recent studies address these issues [10], [11], [16], [40], [41], [19], [27], [32] (detailed in Section 2). These studies all adopt the approach of distributed monitoring of RF airwaves, which has also been adopted by most commercial products (e.g., [1], [3], [9], [4], [2], and [8]).

An alternative approach to managing a wireless network is through centralized monitoring at a *single* aggregation point. This single monitoring point is located at the edge of a local network (e.g., at a gateway router) and captures all traffic coming into and getting out of the local network. This centralized approach is scalable, requiring little deployment costs, and is easy to manage and maintain. However, a key challenge when using this approach for real-time network management is *online* detection of wireless traffic. This is because a local network typically supports both Ethernet and WLAN technologies, and hence, the aggregation point observes a mixture of wired and wireless traffic.

Online detection of wireless traffic at the aggregation point is not an easy task. It cannot be achieved based on IP addresses. This is because a network administrator may not allocate separate IP address pools for wired and wireless hosts. Even if there were separate pools, a host with an address from the wired address pool may act as a NAT box for a set of wireless hosts or install a wireless router and becomes a wireless host. In this paper, we develop two online algorithms to detect wireless traffic. Our algorithms take advantage of timing information at the aggregation point and can detect wireless traffic that is behind NAT boxes or user-installed wireless routers. Our main contributions are listed as follows:

- We extend the analysis in [37] and demonstrate that using TCP ACK-pairs can effectively differentiate Ethernet and wireless connections (including both 802.11b and 802.11g). Our analysis exploits fundamental properties of the 802.11 CSMA/CA MAC protocol and the half-duplex nature of wireless channels.
- We develop two online algorithms to detect wireless traffic. Both algorithms use sequential hypothesis tests and make prompt decisions as TCP ACK-pairs are observed at the monitoring point. 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 wireless traffic.
- We have built a system for online detection of wireless traffic using the above algorithms and

- *W. Wei and B. Wang are with the Computer Science and Engineering Department, University of Connecticut, 371 Fairfield Way, Unit 2155, Storrs, CT 06269-2155. E-mail: {weiwei, bing}@engr.uconn.edu.*
- *K. Suh is with the School of Information Technology, Illinois State University, Old Union Building, Campus Box# 5150, Normal, IL 61790-5150. E-mail: kwsuh@ilstu.edu.*
- *Y. Gu is with NEC Laboratories America, Princeton, NJ 08540. E-mail: yugu@nec-labs.com.*
- *J. Kurose and D. Towsley are with the Department of Computer Science, University of Massachusetts, Amherst, MA 01003. E-mail: {kurose, towsley}@cs.umass.edu.*
- *S. Jaiswal is with the Bell Labs Research India, Alcatel-Lucent Technologies, Salarpuria Ascent, 3rd Floor, 77, JNC Road, Koramangla Industrial Area, Bangalore 560095, India. E-mail: jsharad@alcatel-lucent.com.*

Manuscript received 18 Mar. 2008; revised 25 June 2008; accepted 18 Aug. 2008; published online 4 Sept. 2008.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2008-03-0090. Digital Object Identifier no. 10.1109/TMC.2008.126.

deployed it at the gateway router of the University of Massachusetts (UMass), Amherst. Extensive experiments in various scenarios have demonstrated the effectiveness of our algorithms: 1) the algorithm that requires training makes detections mostly within 10 seconds, and the false-positive and false-negative ratios are close to zero, 2) the algorithm that does not require training detects 60 percent to 76 percent of the wireless hosts without any false positives, and 3) both algorithms have computation and storage overhead well within the capability of commodity equipment. We further demonstrate that our scheme can detect connection-type switchings and wireless networks behind a NAT box, and it is effective even when end hosts have high CPU, disk, or network utilizations.

Our proposed algorithms have a number of important applications in *real-time* WLAN management. For instance, they are useful to detect rogue or unauthorized access points (APs). Suppose a host not authorized to use a wireless network installs a rogue AP for wireless connection. The traffic of this host is captured at the aggregation point. Using our online algorithms, a network administrator will detect that the host uses wireless while it is not authorized to do so and hence determines that it uses a rogue AP. Our proposed scheme can also help to monitor the performance of wireless hosts, which are more vulnerable to performance problems due to the unreliable nature of the wireless medium. More specifically, a network administrator may identify wireless hosts in real time using our algorithms, monitor their performance, and predict and/or detect performance degradations.

The rest of the paper is organized as follows: Section 2 describes related work. Section 3 presents the problem setting and a high-level description of our approach. Section 4 analyzes TCP ACK-pairs in Ethernet and WLAN. Sections 5 and 6 present our online algorithms and online detection system, respectively. Sections 7 and 8 present the experimental evaluation methodology and results, respectively. Finally, Section 9 discusses several issues related to using our algorithms, and Section 10 concludes the paper and presents future work.

2 Related Work

The study most closely related to ours is [37], which proposes an iterative Bayesian inference technique to detect wireless traffic based on passive measurement at an aggregation point in an *offline* manner. The focus there is on determining the extent of wireless usage and the belief that a flow is from a wireless host. Our focus here is on *online* detection of wireless traffic.

Detecting wireless traffic has also been studied in several other efforts. However, none of them adopts a passive online approach. Baiamonte et al. [12] use entropies to detect wireless connections in an *offline* manner. Beyah et al. [13] use visual inspection to detect wireless traffic, which cannot be carried out automatically. Mano et al. [28] propose a technique that requires segmenting large packets into smaller ones to detect wireless traffic. In other studies, differentiating wireless traffic and other types of traffics is based on active measurements [39] or certain assumptions about wireless links (such as very low bandwidth and high loss rates) [18].

Several recent studies focus on WLAN management. Adya et al. [10] present a client-based architecture to detect and diagnose faults. Bahl et al. [11] propose using USB devices that are attached to desktops to monitor an enterprise WLAN. This architecture has been recently extended to provide location-based management [16]. Yeo et al. propose a framework that merges link-level measurement from multiple distributed air monitors for WLAN management [40], [41]. This framework is substantially extended in Jigsaw [19] and Wit [27], where the authors provide formal and systematic techniques to construct a global view of the network by merging and synchronizing traces from multiple locations. Based on the global view, Cheng et al. infer all sources of delays due to media access and mobility for cross-layer WLAN diagnosis. In another recent study, Sheth et al. diagnose WLAN problems by detecting root causes at the physical layer [32]. All the above studies utilize distributed monitoring of RF airwaves. This is also true for most commercial wireless network management products [1], [3], [4], [2], [8]. The rationale is that RF airwave monitoring provides detailed low-level (i.e., PHY and MAC) information that is critical for analyzing the behavior of a network and pinpoints the exact causes of a fault. Our study takes the approach of centralized monitoring at a single aggregation point. The captured information is at higher layers (i.e., IP and transport layers) and hence may not provide sufficient insights into the root causes of a fault. However, as mentioned earlier, it has a number of applications in real-time WLAN management, e.g., in detecting rogue APs and detecting/predicting performance degradations.

Two recent studies [42], [26] focus on a specific WLAN management task—rogue AP detection. Yin et al. [42] use wireless sniffers and a verifier in the wired network to detect protected layer-3 rogue APs. Ma et al. [26] propose a framework that combines distributed detection through wireless sniffers and centralized detection at a gateway router. Both studies still heavily rely on airwave monitoring and target at detecting rogue APs only. We provide a scalable online approach to detecting wireless traffic, which has broader applications in real-time WLAN management.

Passive measurement at a single aggregation point falls broadly into "measurement-in-the-middle," i.e., measurements are taken at a single point in the "middle" of the end-to-end connections. The studies in [22] and [23] infer end-to-end properties of a TCP connection through measurement-in-the-middle. Our study differs in that we focus on differentiating wired and wireless traffic.

Last, sequential hypothesis testing [35] provides an opportunity to make decisions as data streams come in and thus is a suitable technique for our purpose. It has also been used for detecting portscans [24], jamming attacks [25], and misbehavior in WLANs [30], [14].

3 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. Therefore, the aggregation point captures a

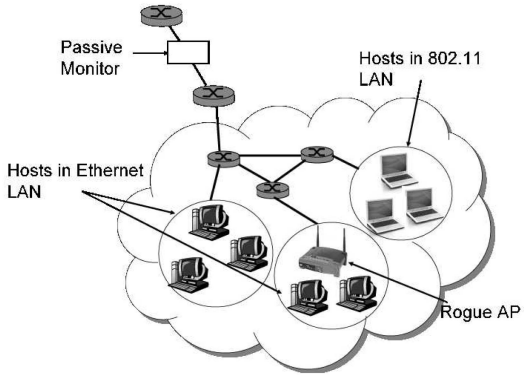


Fig. 1. Problem setting: a monitoring point at an aggregation point (e.g., the gateway router) captures incoming traffic and outgoing traffic. Our goal is to detect wireless hosts in *real-time* based on *passive* measurements at the monitoring point. This figure also illustrates a scenario where a host not authorized to use the WLAN installs a rogue AP.

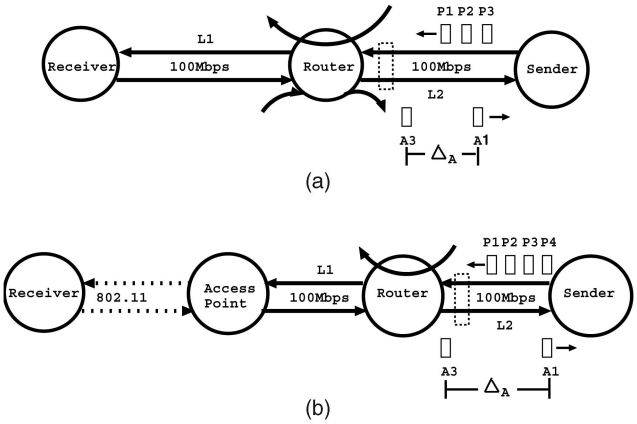


Fig. 2. Settings for the analysis. (a) Ethernet. (b) WLAN (802.11b or 802.11g). The dashed rectangle between the sender and the router represents the monitoring point. The pair of ACKs, A_1 and A_3, forms an ACK-pair.

mixture of wired and wireless traffic. Our goal is to detect what hosts use WLAN inside the local network in *real time*. For this purpose, 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 4 and 5.

We have shown that *inter-ACK time* is a statistic that can be used to effectively detect wireless hosts in [37]. An inter-ACK time is the interarrival 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 [37], 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 4, we extend the analysis in [37] 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 lightweight algorithms (see Section 5), 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.

4 Analysis of TCP ACK-Pairs

In this section, we extend the analysis in [37] 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.

4.1 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. 2a, the receiver uses Ethernet; in Fig. 2b, the receiver uses 802.11b or 802.11g WLAN. We refer to the above settings as the *Ethernet setting* and the *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 AP resides between the router and the receiver. The AP and the router are connected by link L_1 with 100-Mbps bandwidth, and the receiver is connected to the AP 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 the TCP delayed-ACK policy,[1] since this policy is commonly used in practice [31], [7]. To accommodate the effects of delayed ACK, we consider four data packets P_1, P_2, P_3, and P_4, each of 1,500 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 interarrival 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 1,500 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 [21]) 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

1. 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.

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 AP and the receiver is under *idealized conditions*, i.e., the channel is perfect, and is only used by the AP 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.

4.2 Analysis of Ethernet

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

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

Theorem 2 (Median Inter-ACK time for Ethernet). *Let* $\{\Delta_i^A\}_{i=1}^n$ *denote an independent and identically distributed (i.i.d.) sequence of* n *inter-ACK times from a host (they can be from different TCP flows) in the Ethernet setting. Let* $\xi_{.5}^n(\Delta_A)$ *denote the sample median of* $\{\Delta_i^A\}_{i=1}^n$. *Then, when* $0 < \rho_D$, $\rho_A \leq 1$ *and* $43 \leq n \leq 100$, *we have* $P(\xi_{0.5}^n(\Delta_A) \leq 600\ \mu\text{s}) \approx 1$. *Furthermore,* $\lim_{n\to\infty} P(\xi_{.5}^n(\Delta_A) \leq 600\ \mu\text{s}) = 1$.

Both of the above theorems will be used explicitly to construct a sequential hypothesis test in Section 5.2.

4.3 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 AP 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 μs, which includes the overhead to transmit physical-layer, MAC-layer, IP, and TCP headers, the overhead for ACK transmission, and the durations of one SIFS and DIFS [20]. The slot time is 20 μs, and a wireless device waits for a random backoff time uniformly distributed in $[0, 31]$ time slots (i.e., $[0, 620]$ μs) 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 1,500 bytes is uniformly distributed in [1,570, 2,190] μs. The MAC service time of an ACK of 40 bytes is uniformly distributed in [508, 1,128] μs. We have the following theorem for the 802.11b WLAN setting; the proof is found in Appendix C.

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

4.4 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 μs. The slot time is 9 μs. The receiver waits for a random backoff time uniformly distributed in [0, 15] time slots (i.e., [0, 135] μs) before transmitting a packet. Therefore, the MAC service time of a data packet (1,500 bytes) is uniformly distributed in [325, 460] μs; the MAC service time of an ACK (40 bytes) is uniformly distributed in [109, 244] μs. We have the following theorem for the 802.11g WLAN setting; the proof is found in Appendix D.

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

4.5 Summary of Analysis

The above analysis demonstrates that even when a WLAN is under idealized conditions while an Ethernet LAN is fully utilized, using TCP ACK-pairs can effectively differentiate Ethernet and WLAN connections: for Ethernet, less than 18 percent of the inter-ACK times exceed 600 μs, while for 802.11b and 802.11g, at least 96 percent and 45 percent of the inter-ACK times exceed 600 μs, respectively (see Theorems 1, 3, and 4). Under more realistic conditions (e.g., noisy wireless channel and with contention), the inter-ACK times in WLAN may be even larger than those in Ethernet.

5 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 a 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 4). We refer to these two algorithms as *the sequential hypothesis test with training* and *the sequential hypothesis test without training*, respectively. The algorithm without training, although not as powerful as the one with training (see Section 8), is suitable for scenarios where the inter-ACK time distributions are not available a priori (e.g., in organizations with no authorized wireless networks, where detecting wireless traffic is crucial since the presence of wireless traffic implies rogue APs and, hence, severe security threats).

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.

5.1 Sequential Hypothesis Test with Training

Algorithm 1. Sequential hypothesis test with training

$n = 0$, $l_E = l_W = 0$
loop
  Identify an ACK-pair
  $n = n + 1$
  $p_n = P(\Delta_n^A = \delta_n^A \mid E)$, $q_n = P(\Delta_n^A = \delta_n^A \mid W)$
  $l_E = l_E + \log p_n$, $l_W = l_W + \log q_n$
  if $l_W - l_E > \log K$ **then**
    Report WLAN, $n = 0$, $l_E = l_W = 0$.
  else if $l_W - l_E < -\log K$ **then**
    Report Ethernet, $n = 0$, $l_E = l_W = 0$.
  else if $n = N$ **then**
    Report undetermined, $n = 0$, $l_E = l_W = 0$.
  end if
end loop

We have demonstrated that the inter-ACK time distributions for Ethernet and WLAN differ significantly (see Section 4). 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 larger 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 ith 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 ith 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 i.i.d., we have

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

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

This test updates L_W and L_E as an ACK-pair is observed. Let $K > 1$ be a threshold. If after the nth 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 our implementation of the algorithm, 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 Algorithm 1, where $N = 100$. 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.

5.2 Sequential Hypothesis Test without Training

Algorithm 2. Sequential hypothesis test without training

$m = n = 0$
loop
  Identify an ACK-pair
  $n = n + 1$
  $m = m + \mathbf{1}(\delta_n^A \geq 600\ \mu\text{s})$
  $\hat{p} = m/n$
  if $\hat{p} = 1$ **and** $n > -\frac{\log K}{\log \theta}$ **then**
    Report WLAN, $m = n = 0$.
  else if $n < \frac{m(\log \hat{p} - \log \theta + \log(1-\theta) - \log(1-\hat{p})) - \log K}{\log(1-\theta) - \log(1-\hat{p})}$ **then**
    Report WLAN, $m = n = 0$.
  else if $n \geq 43$ **and** $\hat{p} \geq 0.5$ **then**
    Report WLAN, $m = n = 0$.
  else if $n = N$ **then**
    Report undetermined. $m = n = 0$.
  end if
end loop

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 μs is small for Ethernet hosts, while it is much larger for WLAN hosts (see Section 4). In the following, we first construct a likelihood ratio test [15] and then derive from it a sequential hypothesis test.

The likelihood ratio test is described as follows: Let p be the probability that an inter-ACK time exceeds 600 μs, that is, $p = P(\Delta_A > 600\ \mu\text{s})$. By Theorem 1, we have $p < \theta = 0.18$ for the 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 μs. Let $K > 1$ be a threshold. Then, the likelihood ratio test rejects the null hypothesis H_0 when

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

In the middle term above, the numerator is the maximum probability of having the observed sequence (which has m inter-ACK times exceeding 600 μs) computed over parameters in the null hypothesis (i.e., $0 \leq p \leq \theta$); the denominator is the maximum probability of having the observed sequence over all possible parameters (i.e., $0 \leq p \leq 1$). 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 μs, 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., $\sup_{0 \leq p \leq 1} p^m (1-p)^{n-m}$ is achieved when $p = \hat{p}$. When $\hat{p} \leq \theta$, we have $\sup_{0 \leq p \leq \theta} p^m (1-p)^{n-m} = \sup_{0 \leq p \leq 1} p^m (1-p)^{n-m}$, 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

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

which is equivalent to

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

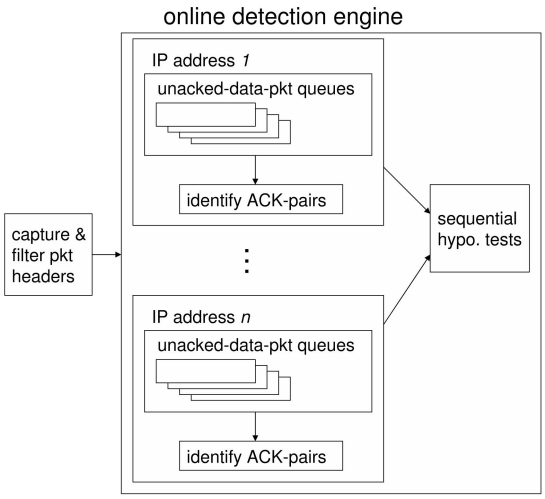


Fig. 3. Online wireless-traffic detection system.

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 $n \geq 8$. This implies that we need at least eight 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 $P(\xi^n_{0.5}(\Delta_A) \leq 600\ \mu\text{s}) \approx 1$ for Ethernet hosts. Therefore, an additional condition to reject H_0 is when $43 \leq n \leq 100$ and $\hat{p} > 0.5$ (because this condition implies that at least half of the inter-ACK observations exceed 600 μs, that is, $\xi^n_{0.5}(\Delta_A) > 600\ \mu\text{s}$, which contradicts Theorem 2).

We combine the above three conditions to construct a sequential hypothesis test as shown in Algorithm 2, where $\mathbf{1}(\cdot)$ is the indicator function, and $N = 100$. 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.

6 Online Detection System

We now describe the design of a system for online detection of wireless traffic. This system consists of two major components, as illustrated in Fig. 3. 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 Section 5. We next describe the online detection engine, the core component in the system, in more detail. Afterward, we describe how to identify ACK-pairs in real time and obtain inter-ACK time distributions beforehand.

6.1 Online Detection Engine

The online detection engine makes detections 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,[2] 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 time stamp (8 bytes), and the size (2 bytes) of the packet. In addition, the online detection engine also records the latest ACK for each TCP flow in memory. This 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 our algorithm (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 90 percent of the queues contain less than three items (see Section 8.3), indicating that the memory usage of this online detection system is low.

6.2 Online Identification of TCP ACK-Pairs

As described earlier, two successive ACKs form an ACK-pair if the interarrival time of their corresponding data packets at the monitoring point is less than a threshold T (chosen as 240 μs or 400 μs in our system; see Section 8). Taking account of several practical issues, we further impose the following additional restrictions when identifying ACK-pairs. First, we exclude all ACKs whose corresponding data packets have been retransmitted or reordered. Second, to ensure that two ACKs are successive, we require that the difference of their IPIDs to be no more than one.[3] Third, we require that the ACKs are for relatively large data packets (of size at least 1,000 bytes) to be consistent with the assumption of our analysis (in Section 4). We also exclude ACKs due to expiration of delayed-ACK timers (if delayed ACK is implemented) since such an ACK is not released immediately after its corresponding data packet, and hence, the

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

3. The IPID field carries a copy of the current value of an IPID counter in a host's IP stack. Many commercial operating systems maintain a single IPID counter that is incremented whenever a new IP packet is generated; other systems implement an IPID counter as a per-flow counter, a random number, or a constant [17]. Our IPID restriction is most effective when a host uses a single IPID counter. It is also helpful when a host uses per-flow counters. We do not impose this restriction when IPID is not monotonically increasing.

interarrival time of this ACK and its previous ACK does not reflect the characteristics of the access link. We use the technique in [36] to infer whether delayed ACK is implemented, which further requires that the inter-ACK time of an ACK-pair to be below 200 ms.

6.3 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 7.2; training sets for other networks can be constructed in a similar manner.

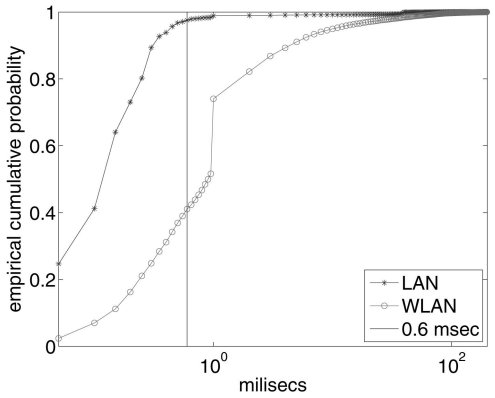


Fig. 4. Ethernet and WLAN inter-ACK time distributions obtained from training sets ($T = 240\ \mu$s).

7 Evaluation Methodology

We evaluate the performance of our online 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.

7.1 Measurement Equipment

Our measurement equipment consists of 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 that connects 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 time stamp. 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 of 512 Kbytes), 2-Gbyte memory, and SCSI hard disks.

7.2 Training Sets

Training sets are required to obtain inter-ACK time distributions (see Section 6.3). 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 1,177, 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 February and April 2005. In early 2006, 802.11g APs were deployed on the UMass campus, and more users started 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 described as follows: We divide the range from 0 to 1 ms into 50-μs bins and divide the range from 1 ms to 200 ms (which is the maximum value for inter-ACK times) into 1-ms bins. Fig. 4 plots the Cumulative Distribution Functions (CDFs) of the inter-ACK times for Ethernet and WLAN, where the threshold $T = 240\ \mu$s. We observe that 2.5 percent of the inter-ACK times for Ethernet hosts are above 600 μs, while 59.0 percent of the inter-ACK times for WLAN hosts are above 600 μs, confirming our analytical results in Section 4 (for Ethernet, the observed value is lower than the analytical result because our analysis is very conservative; for WLAN, the observed value is between the analytical results for 802.11b and 802.11g since the training set contains both types of wireless traffic).

7.3 Test Sets

To validate that our algorithms can detect WLAN hosts and do 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 1,177 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 percent of them use different versions of the Windows operating system (e.g., Windows 2000, Windows ME, and Windows XP); the rest use different variants of Linux and Unix operating systems (e.g., RedHat, Solaris, CentOS, and Fedora Core). 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-type switchings and wireless traffic behind a NAT box by conducting additional experiments in the Computer Science Department. The total IP space monitored in our experimental evaluation has 3,217 addresses: 1,177 addresses in the WLAN test set and 2,540 addresses in the Computer Science Department.

7.4 Offline and Online Evaluation

We evaluate both the offline and online performance of our algorithms. In the offline evaluation, we first store the traffic measurements (to the hard disk) and then apply our

TABLE 1
Offline Evaluation of the Sequential Hypothesis Test with Training: Results on WLANs (10/20/2006)

	$T = 240\ \mu s$			$T = 400\ \mu s$		
	$K = 10^4$	$K = 10^5$	$K = 10^6$	$K = 10^4$	$K = 10^5$	$K = 10^6$
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 2
Offline Evaluation of the Sequential Hypothesis Test with Training: Results on Ethernet (10/20/2006)

	$T = 240\ \mu s$			$T = 400\ \mu s$		
	$K = 10^4$	$K = 10^5$	$K = 10^6$	$K = 10^4$	$K = 10^5$	$K = 10^6$
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%		

algorithms on the collected trace. In the online evaluation, we run our algorithms while capturing the data at the measurement point. The offline evaluation, although not capturing the targeted operation mode of our algorithms, allows us to investigate the impact of various parameters (e.g., T, the threshold to identify ACK-pairs, and K, the threshold in the sequential hypothesis tests). The online evaluation investigates the performance of our algorithms in their targeted operation mode.

8 Experimental Evaluation

We now describe our experimental results. In our experiments, the online detection algorithms make a decision (i.e., detecting WLAN, Ethernet, or undetermined) using 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 detection is referred to as the *detection time*. The *correct detection ratio* is the total number of correct detections over the total number of detections.

In the following, we first evaluate the performance (in accuracy and promptness) of our online detection algorithms (Sections 8.1 and 8.2). We then investigate the scalability of our approach (Section 8.3). Afterward, we demonstrate that our approach is effective in detecting wireless traffic behind a NAT box (Section 8.4). Last, we show that our approach can quickly detect connection-type switchings (Section 8.5) and is robust to high CPU, disk, or network utilizations at end hosts (Section 8.6).

8.1 Performance of the Sequential Hypothesis Test with Training

We now investigate the performance of the sequential hypothesis test with training. The Ethernet and WLAN inter-ACK time distributions required by this algorithm are obtained as described in Section 7.2. We describe results from both offline and online evaluations.

8.1.1 Offline Evaluation

We collect measurements on three consecutive days, from 10/18/2006 to 10/20/2006. For each day, the duration of the trace is 6 to 7 hours. The threshold to identify ACK-pairs, T, is either 240 μs or 400 μs. The threshold to decide a host's connection type, K, is 10^4, 10^5, or 10^6. We only describe the results for the trace collected on 10/20/2006; the results for the other two days are similar.

Tables 1 and 2 present the detection results for the WLAN and Ethernet test sets, 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 percent. On the average, it takes less than 10 ACK-pairs (corresponding to 250-347 data packets) to make a detection for WLAN and less than 20 ACK-pairs (corresponding to 87-124 data packets) for Ethernet. The larger number of data packets for detecting a WLAN host can be explained as follows: Inter-ACK times in WLAN tend to be large (compared to those in Ethernet), thus leading to large interarrival times between newly triggered data packets (due to TCP's self-clocking mechanism). When interarrival times of data packets exceed the threshold T, the corresponding ACKs do not qualify as ACK-pairs. 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 1 and 2.

The detection-time distributions for both Ethernet and WLAN are shown in Fig. 5, where $T = 240\ \mu$s, and $K = 10^6$. The median detection times are around 1 second and 10 seconds for Ethernet and WLAN, respectively. The much shorter detection time in Ethernet is due to higher ACK-pair

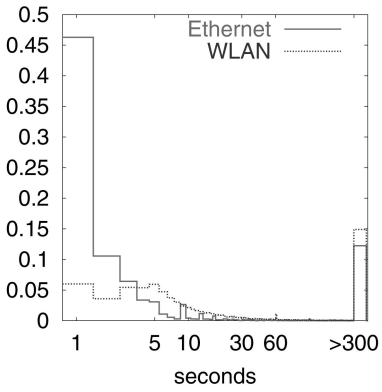


Fig. 5. Detection-time distributions for the trace collected on 10/20/2006 ($T = 240\ \mu$s, $K = 10^6$, and $N = 100$).

ratios. We also observe some cases of long detection times (more than 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 percent of ACK-pairs used in WLAN detection and 89 percent of ACK-pairs used in Ethernet detection are generated by Web traffic, indicating that our approach is effective even for short flows.

8.1.2 Online Evaluation

We run our detection algorithm online on three consecutive days, from 10/25/2006 to 10/27/2006. The evaluation on each day lasts for 6 to 7 hours. We set $T = 240\ \mu$s and $K = 10^6$, representing a conservative selection of parameters. Table 3 presents the detection results for both test sets. We observe consistent results as those in the offline evaluation—the detection is highly accurate and prompt. The average numbers of ACK-pairs and data packets required for a detection are also consistent with those in the offline evaluation.

8.2 Performance of the Sequential Hypothesis Test without Training

We now examine the performance of the sequential hypothesis test without training. 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. This demonstrates that although this algorithm is derived using analytical results in Section 4 (in a setting where the receiver is one hop away from the router), it is accurate in more general settings (the Ethernet hosts in the Computer Science building are three hops away from the gateway router). This is not surprising since this algorithm is based on an extremely conservative analysis (assuming that the single Ethernet link is fully utilized). For the WLAN test set, of all the hosts with at least one ACK-pair, this algorithm detects 60 percent to 76 percent of them as WLAN hosts. Table 4 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 that requires training.

8.3 Scalability Study

We investigate the scalability of our approach by examining the CPU and memory usage on the PC that runs the detection algorithms (the configuration of the PC is described in Section 7.1). During online evaluation, we sample the CPU usage at the measurement PC every 30 seconds. The maximum CPU usage is 9.1 percent (we have made no special efforts to optimize our implementation), indicating that the measurement task is well within the capability of a commodity 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 6). Fig. 6 plots the CDF of the

TABLE 3
Online Evaluation of the Sequential Hypothesis Test with Training (10/25/2006 to 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 4
Evaluation of the 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

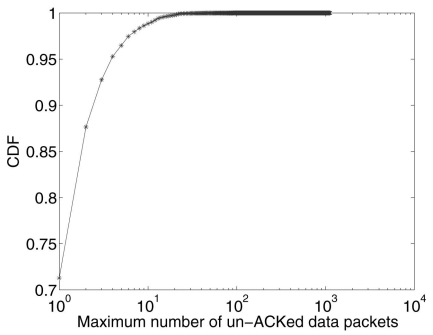


Fig. 6. CDF of the number of items in the unacked-data-packet queues.

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 8,244). We observe that most of the queues are very short: 90 percent of them have less than three items, indicating a very low memory usage (each data item only keeps 14 bytes of data; see Section 6.1). However, we also observe some long queues. We conjecture that these long queues are due to routing changes or abnormal behaviors in the routes. As an optimization to our online detection system, we can discard unacked-data-packet queues longer than a certain threshold.

8.4 Detecting Wireless Networks behind NAT

We now demonstrate that our approach is equally applicable to detect wireless networks behind a NAT box. Note that schemes based on MAC addresses (e.g., [9], [4], and [10]) fail to detect this type of wireless traffic, 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.

8.4.1 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 first reach host A. Host A then translates the addresses of these packets and forwards them to the Internet through its Ethernet card.

We conduct an experiment with the above setup on 10/26/2006. The experiment lasts for about 2 minutes. We observe 163 ACK-pairs. Among them, 92 percent of the ACK-pairs are from Web traffic via port 80. The remaining ACK-pairs are from port 1,935, which is used by Macromedia Flash Communication Server MX for the Real-Time Messaging Protocol (RTMP). The sequential hypothesis test with training makes 37 online detections, all as WLAN host. On the average, one detection is made for every four ACK-pairs. The above results demonstrate that our test can effectively detect wireless networks behind NAT boxes.

8.4.2 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.

8.5 Detecting Connection-Type Switchings

We next explore a scenario where an end host switches between wired and wireless connections. Our goal is to examine whether our detection approach can accurately report the connection-type 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 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 percent. This demonstrates that our approach is still effective when a host switches between using Ethernet and WLAN.

8.6 Detection under High CPU, Disk, or Network Utilizations

We now investigate the performance of our approach when an end host has very high CPU, disk, or network utilizations. The reason for considering these three factors is that they may directly and/or indirectly affect packet arrival times at the monitoring point (e.g., a high CPU or disk utilization affects packet generation times at the end host, which further affects packet arrival times at the monitoring point) and hence may affect the performance of our approach.

For the above 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 the 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 percent) by running an infinite loop. For the Ethernet connection, we observe 1,077 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 1,158 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, which is 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 and 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 and 487 ACK-pairs and 46 detections for the WLAN connection. All the detections are correct. When stressing either the downlink or uplink, we observe significantly less ACK-pairs than those when stressing CPU or disk. This is due to the cross traffic generated by the local downloading or uploading activities. We also observe less ACK-pairs when stressing the uplink than those 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 end hosts have high CPU, hard disk, or network utilizations.

9 Discussion

We next discuss several issues related to using our algorithms in practice.

9.1 Heterogeneous Ethernet Backbone

The algorithm that requires training can be easily extended to a network that supports heterogeneous Ethernet (e.g., 10-Mpbs, 100-Mbps, and 1-Gbps Ethernet) as follows: Consider a network that supports both 10-Mbps and 100-Mbps Ethernet. In this case, we obtain the inter-ACK time distributions for both 10-Mbps and 100-Mbps Ethernet through their corresponding training sets. The algorithm calculates two likelihood ratios, for 10-Mbps and 100-Mbps Ethernet over WLAN, respectively. The decision is that a host uses 10-Mbps Ethernet, 100-Mbps Ethernet, or WLAN.

9.2 Location of Monitoring Point

If an institution provides separate Ethernet and WLAN networks (e.g., for security purposes), we need to place two monitoring points, at the gateway routers for the Ethernet and WLAN, respectively. If a local network is multihomed and the incoming and outgoing traffic do not traverse the same access link, we need to monitor multiple access links simultaneously (recent monitoring equipment has this capability).

9.3 Using Our Algorithms in Future Networks

A natural question is whether our algorithms will still be effective in the future, when the bandwidths of Ethernet and WLAN increase and WLAN might provide similar or even higher bandwidth than Ethernet. Our algorithm that requires training will still be effective as long as the inter-ACK time distributions of Ethernet and WLAN are sufficiently different. For instance, it is likely to be effective in differentiating 100-Mbps Ethernet and the emerging 802.11n, since 802.11n still uses CSMA-CA and the wireless channel is half duplex. Our algorithm that requires no training is based on the analytical results of 100-Mbps Ethernet and uses median inter-ACK time. It might become less effective when the bandwidth of WLAN technology approaches or exceeds the bandwidth of Ethernet.

10 Conclusions and Future Work

In this paper, we have proposed two online algorithms, one requires training while the other does not, to detect wireless traffic based on real-time passive measurements collected at a monitoring point. Extensive experiments demonstrated that the algorithm that requires training provides rapid detection and is extremely accurate, the algorithm that does not require training detects 60 percent to 76 percent of the wireless hosts without any false positives, and both algorithms have low computation and storage overhead. Furthermore, our scheme can detect connection-type switchings and wireless networks behind a NAT box and remains effective for end hosts with high CPU, disk, or network utilizations.

As future work, we are exploring in the following three directions: 1) optimize the implementation of our algorithms for speed and even lower memory usage, 2) evaluate the performance of our algorithms when a router shapes the traffic (e.g., it may prioritize incoming traffic for QoS considerations), and 3) explore the impact of mobility on differentiating WLAN and Ethernet traffic.

Appendix A
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 [34], [29]. 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 interdeparture 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. I_D contains two components. One component is the transmission time of packets P_1 and P_2 at queue Q_D, which is $2 \times 120/30 = 8$ time units. The other component is the (discretized) transmission time of the cross-traffic packets that arrive between P_1 and P_3 at queue 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 interarrival time of P_1 and P_3 at queue Q_D is $2 \times 120/30 = 8$ time units, on the average, eight cross-traffic packets arrive between P_1 and P_3 at queue Q_D. This is because given $\rho_D = 1$, the arrival rate of cross-traffic packets at queue Q_D is one 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. 2a, I_A is the same as the interdeparture 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 interarrival time of ACKs 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* $\rho_D = \rho_A = 1$,

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

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 $P(I_A > 20) = P(\Delta_A > 600\ \mu\text{s}) < 0.18$.

We next prove that the theorem also holds when $0 < \rho_D < 1$ or $0 < \rho_A < 1$. When $0 < \rho_D < 1$, the interdeparture 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 interdeparture time of ACKs A_1 and A_3 at queue Q_A is no more than that when $\rho_A = 1$. Therefore, $P(\Delta_A > 600\ \mu\text{s}) < 0.18$ also holds when $0 < \rho_D < 1$ or $0 < \rho_A < 1$. □

APPENDIX B
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* $0 \le q \le 1$. *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:

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

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

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

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

We have

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

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 [33], we have

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

which is equivalent to

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

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

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

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 $\xi_{0.5}^n(\Delta_A) = \Delta_A^{(\lfloor (n+1)/2 \rfloor)}$ regardless of n being even or odd.

Proof. Let $u = P(\Delta_A \le 600\ \mu\text{s})$:

$$\begin{aligned}
P\left(\xi_{0.5}^n(\Delta_A) \le 600\ \mu\text{s}\right) &= \sum_{i=\lfloor (n+1)/2 \rfloor}^{n} \binom{n}{i} u^i (1-u)^{n-i} \\
&= g(n,u),
\end{aligned}$$

where $g(n,u)$ is as defined in Lemma 4. By Lemma 4, $g(n,q)$ is an increasing function of q for $0 \le q \le 1$. By Theorem 1, we know that $u > 1 - 0.18 = 0.82$. Therefore, we have $g(n,u) \ge g(n,0.82)$. Hence, $P(\xi_{0.5}^n(\Delta_A) \le 600\ \mu\text{s}) \ge g(n,0.82)$. By direct calculation, we have $P(\xi_{0.5}^n(\Delta_A) \le 600\ \mu\text{s}) \approx 1$ for $43 \le n \le 100$. Furthermore, since $0.82 > 1/2$, by Lemma 4, we have $\lim_{n\to\infty} P(\xi_{0.5}^n(\Delta_A) \le 600\ \mu\text{s}) = 1$. □

APPENDIX C
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 interarrival time of data packets P_i and P_{i+1} at the AP, $i = 1, 2, 3$. Then, $P(\Delta^D_{i,i+1} < 1{,}570\ \mu s) \approx 1$, $P(\Delta^D_{i,i+1} < 325\ \mu s) \geq 0.89$.*

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, $P(\Delta^D_{i,i+1} < 1{,}570\ \mu s) > P(I^D_{i,i+1} = 52) = \sum_{x=4}^{52} \frac{4^{x-4}e^{-4}}{(x-4)!} \approx 1$. When $\rho_D < 1$, the value of $\Delta^D_{i,i+1}$ is less than that when $\rho_D = 1$, and hence, $P(\Delta^D_{i,i+1} < 1{,}570\ \mu s) \approx 1$ 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 $\Delta^D_{i,i+1} \leq 1{,}570\ \mu s$, $i = 1, 2, 3$. 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 1,570 μs in 11-Mbps 802.11b. Assuming independence, we have

$$P(C) = \prod_{i=1}^{3} P\left(\Delta^D_{i,i+1} \leq 1{,}570\ \mu s\right).$$

From Lemma 5, $P(C) \approx 1$. Let $\bar{C}$ denote the complementary condition of C. Then,

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

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 ACKs A_1 and A_3, since the transmission time of a data packet is at least 1,570 μs. 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 that 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, $P(\Delta_A > 600\,\mu s) \geq P(\Delta_A > 600\,\mu s | C) > 1 - 0.04 > 0.96$. □

APPENDIX D
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 μs 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 μs. 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 μs. Let X and Y denote respectively the MAC service times 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 4.4, X and Y are uniformly distributed in [325, 460] μs and [109, 244] μs, respectively. Then, by a standard technique, we have $\alpha = 0.70$. Hence,

$$\begin{aligned} P(\Delta_A \leq 600\ \mu s \mid C) &= \phi(1-\phi)/4 + (1-\phi)/4 + \alpha\phi(1-\phi)/2 \\ &\quad + \alpha(1-\phi)/8 \\ &= (-87{,}718\phi^2 + 38{,}451\phi \\ &\quad + 49{,}267)/145{,}800 \\ &\leq 0.37. \end{aligned}$$

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

ACKNOWLEDGMENTS

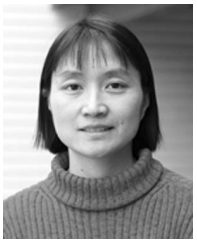
A preliminary version of this paper appeared in IMC 2007 [38]. This research was supported in part by the National Science Foundation under NSF Grants ANI-0325868, ANI-0240487, ANI-0085848, CNS-0519998, CNS-0519922, CNS-0524323, and EIA-0080119, and NSF CAREER Award 0746841. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies. The authors would like to thank Prof. Richard S. Ellis (UMass, Amherst) and Prof. Guanling Chen (UMass, Lowell) for the helpful discussion. They also wish to thank Rick Tuthill from the Office of Information Technology, UMass, Amherst, for helping them understand the UMass network architecture and for installing and managing the monitoring equipment.

REFERENCES

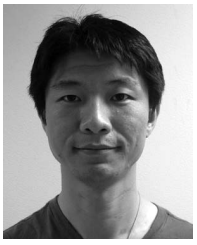
[1] *AirDefense, Wireless LAN Security,* http://airdefense.net, 2008.
[2] *AirMagnet,* http://www.airmagnet.com, 2008.
[3] *AirWave, AirWave Management Platform,* http://airwave.com, 2008.
[4] *Cisco Wireless LAN Solution Engine (WLSE),* http://www.cisco.com/en/US/products/sw/cscowork/ps3915/, 2008.
[5] *Host AP,* http://hostap.epitest.fi, 2008.
[6] http://www.endace.com, 2008.
[7] *Microsoft Windows 2000 TCP/IP Implementation Details,* http://www.microsoft.com/technet/itsolutions/network/deploy/depovg/tcpip2k.mspx, 2008.
[8] *NetStumbler,* http://www.netstumbler.com, 2008.
[9] *Rogue Access Point Detection: Automatically Detect and Manage Wireless Threats to Your Network,* http://www.proxim.com, 2008.
[10] A. Adya, V. Bahl, R. Chandra, and L. Qiu, "Architecture and Techniques for Diagnosing Faults in IEEE 802.11 Infrastructure Networks," *Proc. ACM MobiCom '04,* Sept. 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," *Proc. Fourth ACM Int'l Conf. Mobile Systems, Applications, and Services (MobiSys '06),* June 2006.
[12] V. Baiamonte, K. Papagiannaki, and G. Iannaccone, "Detecting 802.11 Wireless Hosts from Remote Passive Observations," *Proc. IFIP/TC6 Networking,* May 2007.
[13] R. Beyah, S. Kangude, G. Yu, B. Strickland, and J. Copeland, "Rogue Access Point Detection Using Temporal Traffic Characteristics," *Proc. 47th Ann. IEEE Global Telecomm. Conf. (GLOBECOM '04),* Dec. 2004.
[14] A.A. Cardenas, S. Radosavac, and J.S. Baras, "An Analytical Evaluation of MAC Layer Misbehavior Detection Schemes," *Proc. IEEE INFOCOM '07,* May 2007.
[15] G. Casella and R.L. Berger, *Statistical Inference.* Duxbury Thomson Learning, 2002.
[16] R. Chandra, J. Padhye, A. Wolman, and B. Zill, "A Location-Based Management System for Enterprise Wireless LANs," *Proc. Fourth Usenix Symp. Networked Systems Design and Implementation (NSDI '07),* Apr. 2007.
[17] W. Chen, Y. Huang, B.F. Ribeiro, K. Suh, H. Zhang, E. de Souza e Silva, J. Kurose, and D. Towsley, "Exploiting the IPID Field to Infer Network Path and End-System Characteristics," *Proc. Sixth Passive and Active Measurement Workshop (PAM),* 2005.
[18] L. Cheng and I. Marsic, "Fuzzy Reasoning for Wireless Awareness," *Int'l J. Wireless Information Networks,* vol. 8, no. 1, 2001.
[19] 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," *Proc. ACM SIGCOMM '06,* Sept. 2006.
[20] 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.
[21] *IEEE 802.11, 802.11a, 802.11b Standards for Wireless Local Area Networks,* http://standards.ieee.org/getieee802/802.11.html, 2008.
[22] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone," *Proc. IEEE INFOCOM '03,* Mar. 2003.
[23] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Inferring TCP Connection Characteristics through Passive Measurements," *Proc. IEEE INFOCOM '04,* Mar. 2004.
[24] J. Jung, V. Paxson, A.W. Berger, and H. Balakrishnan, "Fast Portscan Detection Using Sequential Hypothesis Testing," *Proc. IEEE Symp. Security and Privacy,* May 2004.
[25] M. Li, I. Koutsopoulos, and R. Poovendran, "Optimal Jamming Attacks and Network Defense Policies in Wireless Sensor Networks," *Proc. IEEE INFOCOM '07,* May 2007.
[26] L. Ma, A.Y. Teymorian, and X. Cheng, "A Hybrid Rogue Access Point Protection Framework for Commodity Wi-Fi Networks," *Proc. IEEE INFOCOM '08,* Apr. 2008.
[27] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Analyzing the MAC-Level Behavior of Wireless Networks in the Wild," *Proc. ACM SIGCOMM '06,* Sept. 2006.
[28] 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 Trans. Information Systems and Security,* vol. 11, no. 2, Mar. 2008.
[29] *Packet Trace Analysis,* http://ipmon.sprintlabs.com/packstat/packetoverview.php, 2008.
[30] S. Radosavac, J. Baras, and I. Koutsopoulos, "A Framework for MAC Protocol Misbehavior Detection in Wireless Networks," *Proc. ACM Workshop Wireless Security (WiSe '05),* Sept. 2005.
[31] P. Sarolahti and A. Kuznetsov, "Congestion Control in Linux TCP," *Proc. Usenix,* June 2002.
[32] A. Sheth, C. Doerr, D. Grunwald, R. Han, and D.C. Sicker, "MOJO: A Distributed Physical Layer Anomaly Detection System for 802.11 WLANs," *Proc. Fourth ACM Int'l Conf. Mobile Systems, Applications, and Services (MobiSys '06),* June 2006.
[33] A.N. Shiryaev, *Probability,* second ed. Springer, 1995.
[34] K. Thompson, G. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics," *IEEE Network,* vol. 11, no. 6, pp. 10-23, Nov./Dec. 1997.
[35] A. Wald, *Sequential Analysis.* John Wiley & Sons, 1947.
[36] W. Wei, S. Jaiswal, J. Kurose, and D. Towsley, "Identifying 802.11 Traffic from Passive Measurements Using Iterative Bayesian Inference," Technical Report 05-47, Dept. of Computer Science, Univ. of Mass., Amherst, 2005.
[37] W. Wei, S. Jaiswal, J. Kurose, and D. Towsley, "Identifying 802.11 Traffic from Passive Measurements Using Iterative Bayesian Inference," *Proc. IEEE INFOCOM '06,* Apr. 2006.
[38] W. Wei, K. Suh, B. Wang, Y. Gu, J. Kurose, and D. Towsley, "Passive Online Rogue Access Point Detection Using Sequential Hypothesis Testing with TCP ACK-Pairs," *Proc. ACM SIGCOMM Internet Measurement Conf. (IMC '07),* Oct. 2007.
[39] W. Wei, B. Wang, C. Zhang, J. Kurose, and D. Towsley, "Classification of Access Network Types: Ethernet, Wireless LAN, ADSL, Cable Modem or Dialup," *Proc. IEEE INFOCOM '05,* Mar. 2005.
[40] J. Yeo, M. Youssef, and A. Agrawala, "A Framework for Wireless LAN Monitoring and Its Applications," *Proc. ACM Workshop Wireless Security (WiSe '04),* Oct. 2004.
[41] J. Yeo, M. Youssef, T. Henderson, and A. Agrawala, "An Accurate Technique for Measuring the Wireless Side of Wireless Networks," *Proc. Usenix/ACM Workshop Wireless Traffic Measurements and Modeling (WiTMeMo '05),* June 2005.
[42] H. Yin, G. Chen, and J. Wang, "Detecting Protected Layer-3 Rogue APs," *Proc. IEEE Int'l Conf. Broadband Comm., Networks, and Systems (BROADNETS '07),* Sept. 2007.

Wei Wei received the BS degree in applied mathematics from Beijing University, China, in 1992, the MS degree in statistics from Texas A&M University in 2000, and the MS degree in computer science, the MS degree in applied mathematics, and the PhD degree in computer science from the University of Massachusetts, Amherst, in 2004, 2004, and 2006, respectively. He is currently a visiting assistant professor in the Computer Science and Engineering Department, University of Connecticut, Storrs. His research interests are in the areas of computer networks, distributed embedded systems, and performance modeling. He is a member of the ACM and the IEEE.

Kyoungwon Suh received the BS and MS degrees in computer engineering from Seoul National University, Korea, in 1991 and 1993, respectively, the MS degree from the Department of Computer Science, Rutgers University, New Jersey, in 2000, and the PhD degree in computer science from the University of Massachusetts, Amherst, in 2007. In 2008, he served as a technical consultant to NHN Corp. in the area of network security and content distribution network. He is currently an assistant professor in the School of Information Technology, Illinois State University, Normal. His research interests include peer-to-peer and overlay networks, network measurement and inference, network security, and multimedia content distribution. He is a member of the ACM and the IEEE.



Bing Wang received the BS degree in computer science from the Nanjing University of Science and Technology, China, in 1994, the MS degree in computer engineering from the Institute of Computing Technology, Chinese Academy of Sciences, in 1997, and the MS degree in computer science, the MS degree in applied mathematics, and the PhD degree in computer science from the University of Massachusetts, Amherst, in 2000, 2004, and 2005, respectively. Afterward, she joined the Computer Science and Engineering Department, University of Connecticut, Storrs, as an assistant professor. Her research interests are in computer networks, multimedia, and distributed systems. More specifically, she is interested in topics on Internet technologies and applications, wireless and sensor networks, overlay networks, content distribution, network management and measurement, network modeling and performance evaluation. She received a US National Science Foundation CAREER award in 2008. She is a member of the ACM, the ACM SIGCOMM, the IEEE, the IEEE Computer Society, and the IEEE Communications Society.



Yu Gu received the BS and MS degrees in computer science from the Beijing University of Aeronautics and Astronautics in 1998 and 2001, respectively, and the PhD degree in computer science from the computer network research group in the University of Massachusetts, Amherst, in 2008. He is now with NEC Laboratories America, Princeton, New Jersey. His research interests include network measurement, congestion control, anomaly detection, network simulation, and multimedia networks. He is a member of the IEEE.

Jim Kurose received the PhD degree in computer science from Columbia University. He is currently a distinguished university professor in the Department of Computer Science, University of Massachusetts, Amherst. He has been a visiting scientist at IBM Research, INRIA, Institut EURECOM, the University of Paris, LIP6, and Thomson Research Labs. His research interests include network protocols and architecture, network measurement, sensor networks, multimedia communication, and modeling and performance evaluation. He has served as the editor in chief of the *IEEE Transactions on Communications* and was the founding editor in chief of the *IEEE/ACM Transactions on Networking*. He has been active in the program committees for IEEE INFOCOM, ACM SIGCOMM, and ACM SIGMETRICS for a number of years and has served as a technical program cochair for these conferences. He has received a number of awards for his educational activities, including the IEEE Taylor Booth Education Medal. With Keith Ross, he a coauthor of the textbook *Computer Networking, a Top-Down Approach* (fourth ed.), published by Addison-Wesley Longman. He is a fellow of the IEEE.

Don Towsley received the BA degree in physics and the PhD degree in computer science from University of Texas in 1971 and 1975, respectively. From 1976 to 1985, he was a member of the faculty of the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst. He is currently a distinguished professor in the Department of Computer Science, University of Massachusetts. He has held visiting positions at IBM T.J. Watson Research Center, Yorktown Heights, New York; Laboratoire MASI, Paris, France; INRIA, Sophia-Antipolis, France; AT&T Labs—Research, Florham Park, New Jersey; and Microsoft Research Laboratory, Cambridge, United Kingdom. His research interests include networks and performance evaluation. He currently serves as the editor in chief of the *IEEE/ACM Transactions on Networking*, is on the editorial boards of the *Journal of the ACM* and *IEEE Journal on Selected Areas in Communications*, and has previously served on numerous other editorial boards. He was a program cochair of the joint ACM SIGMETRICS and PERFORMANCE 1992 conference and the Performance 2002 conference. He is the recipient of the 2007 IEEE Koji Kobayashi Award, the 2007 ACM SIGMETRICS Achievement Award, the 2008 ACM SIGCOMM Achievement Award, the 1998 IEEE Communications Society William Bennett Best Paper Award, and numerous best conference/workshop paper awards. He is a member of the Operations Research Society of America (ORSA) and the chair of IFIP Working Group 7.3. Last, he has been elected as a fellow of both the ACM and the IEEE.

Sharad Jaiswal received the BE degree in computer science and engineering from the Regional Engineering College, Trichy, India, the MS degree in computer systems engineering from Boston University, and the PhD degree in computer Science from the University of Massachusetts, Amherst, in 2005. He is a member of technical staff at Bell Labs Research India, Alcatel-Lucent Technologies, Bangalore, India. His current interests include network traffic monitoring and modeling and wireless network architecture and design.