# Disambiguation of Residential Wired and Wireless Access in a Forensic Setting

Sookhyun Yang Jim Kurose Brian Neil Levine
Dept. of Computer Science, Univ. of Massachusetts, Amherst, MA, 01003
{shyang, kurose, brian}@cs.umass.edu

***Abstract*—Thousands of cases each year of child exploitation on P2P file sharing networks lead from an IP address to a home. A first step upon execution of a search warrant is to determine if the home's open Wi-Fi or the closed wired Ethernet was used for trafficking; in the latter case, a resident user is more likely to be the responsible party.**

***We propose methods that use remotely measured traffic to disambiguate wired and wireless residential medium access. Our practical techniques work across the Internet by estimating the per-flow distribution of inter-arrival times for different home access network types. We observe that the change of inter-arrival time distribution is subject to several residential factors, including wireless channel contention, differences between OS network stacks, and cable network mechanisms. We propose a model to explain the observed patterns of inter-arrival times, and we study the ability of supervised learning classifiers to differentiate between wired and wireless access based on these remote traffic measurements.***

## I. Introduction

Images of child sexual exploitation are common on BitTorrent, Gnutella, and other file-sharing networks [10], [19]. The end result of network-based investigations of these crimes is evidence that supports a court-issued warrant to enter and search the home associated with the observed IP address. A common alibi is that a third party used the home's open Wi-Fi. Thus, a useful first step during execution of the warrant would be to determine if contraband was distributed on the P2P network using the house's Ethernet network, therefore making the resident user more likely to be the responsible party, or if Wi-Fi was used and the alibi is justified. Indeed, drive-by abuse of open Wi-Fi by criminals has been a documented practice for years [9], [15], but methods to distinguish such access are not available.

In this paper, we investigate methods that use remotely measured traffic to disambiguate wired and wireless residential medium access. Importantly, we place our work in a practical forensic setting by *constraining our approaches to only use remotely gathered "plain view" data that can be gathered legally from p2p networks before a warrant or wiretap is required (in the US).* This constraint distinguishes our work from previous research on wired/wireless disambiguation, which has assumed that measurements are taken from the target's gateway router, which is not only a much less challenging problem but impractical from a forensic setting since it violates the Wiretap Act. Our goal is to provide information to investigators as they execute a search warrant inside a home. In addition to checking alibis and supplying information for a suspect's interview, our techniques are also useful for *forensic triage*. Backlogs of six months are typical for criminal forensics labs, and the easiest way to reduce the queue is to not add to it by eliminating computers from consideration (for example, those that have no wired interface) [13].

Our techniques work across the Internet by estimating the per-flow distribution of *inter-arrival times* of packets transmitted over different types of home access networks, as measured by an investigator at a remote Internet P2P client. Using a set of traces that we collected, we evaluate the ability of a number of classifiers to *remotely* distinguish wired from wireless access within the same house. We also develop a model of packet spacing for residential traffic sent via a cable modem through the Internet that illuminates and explains our classification results. We find that that our approach for classifying wired from wireless traffic can work well, but is subject to several residential factors, including differences between OS network stacks, cable modem mechanisms, and wireless channel contention. Our analysis reveals the following:

- We use a simple decision tree classifier that uses remotely measured traces and identify **25th percentiles** or **entropy** of inter-arrival times distribution of the traces as classification features, achieving a true positive rate (TPR) of 9.0 to 1.0 and false positive rate (FPR) of 0.0 to 0.1 in our studies. For Linux, we can precisely classify wired from wireless using 25th percentiles or entropy in accordance with a cable network's state. But for Windows, we can depend on only entropy as a good classification feature.
- High contention for a wireless channel locally at the target greatly affects classification accuracy, though this can be overcome.
- We evaluate the cases of single and multiple P2P flows from the source, but we find that this distinction does not affect our result. Only the individual throughput of each flow has an impact on the classifier.
- Our classifier must be trained separately for different throughputs from the target; fortunately, this throughput is easily observable at the receiver. Such training can be performed when the search warrant is executed from within the house; there is no reason to train a general classifier for all houses ahead of the warrant's execution.

In a separate technical report [23], we also show that measurements from points "near" the target (i.e., in the same cable network) do not guarantee better classification results. In sum, a second cable modem at the observer adds noise.

Overall, our findings suggest that it is difficult at best to find a foolproof classifier for remote identification in all scenarios. Our goal is to determine the scenarios in which network access type can be accurately determined, and to understand when and why these techniques cannot be reliably used in other scenarios.

## II. Investigative Method and Justification

The general criminal procedure for child pornography (CP) cases is as follows. Investigators search for content on P2P networks.

1) CP files offered in plain view by a peer, identified by IP address, are downloaded by investigators.
2) The download provides sufficient *probable cause* as part of an application for a magistrate-issued search warrant of the home associated with the IP address's billing records.
3) The warrant is issued, and once inside the home, a triage-style search begins for evidence associated with CP, which might not be the previously downloaded content. Users of the home's computers are interviewed.
4) Seized devices are sent to an off-site lab for detailed forensic examination.
5) Evidence found during search is then used to support a criminal trial for receipt, possession, or distribution of CP.

The step of searching a home is time consuming. Homes have an increasing number of devices that can contain evidence, including Xboxes and ebook readers with Web browsers, smart phones, desktops, and laptops. Investigators have three main *triage* aims: *(a)* reducing the numbers of devices that must be examined on-scene since warrants are time-limited; *(b)* reducing the number of devices that must be sent to an off-site central forensics lab for in-depth examination since work queues are months-long; and *(c)* quickly locating a subset of evidence, if it exists, so as to obtain an admission of guilt by a suspect via an interview. All of these practical goals are met more efficiently by knowing whether a computer used over the Internet is likely wired or wireless.

Our goal is to examine whether it is possible to remotely infer the target's access type. Our technique would be used as follows. During Step (1) above, investigators would keep a packet-level trace of the file download, which is already common practice. Using the packet-level trace, investigators identify a criminal's computer setting (such as operating systems, TCP parameters and P2P applications), and characterize the download throughput and *concatenation* rate (as we see in Section V). This information is not a part of the warrant application. During Step (3), the classifier is trained, which can be completed in minutes with a pre-configured program and a laptop with both wired and wireless interfaces. The pre-configured program regenerates the flow, having equivalent throughput and concatenation rate observed in Step (1) via wired and wireless interfaces. The results of classification tests are used on scene to inform triage and user interviews. We note that it would only reduce accuracy to pre-train a classifier from general Internet scenarios.

Importantly, our collection takes place at the investigator's end host. This measurement is possible without warrant or wiretap since the investigator is a party to the communication. In contrast, previous work proposes to collect packets at a network gateway, which is illegal in our forensics context. It is also impractical as investigators cannot know which gateway until they have a suspect; going back to the gateway after the suspect has uploaded the CP to the investigator is too late.

A number of legal issues restrict the initial process of gathering the data we use to infer a target's medium access type [2], [8]. First, US law prohibits government search and seizure of evidence without a warrant if and only if the source of the data has a *reasonable expectation of privacy* (REP) [8]. US courts have found consistently that users of P2P file sharing networks have no REP when investigators are peers in the network; see *U.S. v. Breese, 2008 WL 1376269* and *U.S. v. Gabel, 2010 WL 3927697*. Collecting information at a user's gateway without a warrant is certainly illegal.

Second, prior to obtaining a warrant, law enforcement cannot use technology that is not in "general public use" to obtain information that would otherwise be unavailable. This restriction is a result of *Kyllo v. U.S., 533 U.S. 27 (2001)*. For example, recently the court ruled that software designed for law enforcement to monitor activity on P2P networks does not violate 4th Amendment protections since if it follows the protocol as any peer on the network does. Similarly, in *Massachusetts v. Karch* (2011), the court ruled that law enforcement programs that do not search the remote computer, but "merely gather and evaluate publicly available information with greater efficiency and with an eye toward obtaining evidence of criminal activity" do not violate *Kyllo*, even if the software itself is unavailable for general public use.

Related work, in Section VII, that has been motivated by network monitoring and measurement is also governed by several US federal laws. Sicker et al. [16] provide an excellent overview and discussion of these laws and their consequences for the network traffic measurement research community. Criminal investigations are not included in that analysis since they lack the *provider protection* motive, which is measurement with the aim of protecting the network infrastructure, e.g., detecting or characterizing network attacks. In monitoring settings, clients typically consent to monitoring by the provider as part of an acceptable use policy.

Information gathered in a criminal investigation ideally meets the standards of criminal trials (beyond a reasonable doubt). However, information that meets the *probable cause* (PC) standard used to issue search warrants is still useful. There is no quantification of PC by courts; often it is defined qualitatively as a "fair probability"; see *U.S. v. Sokolow, 490 U.S. 1 (1989)*. We evaluate our work with these standards in mind by quantifying true and false positive rates. Finally, we note that we expect that the techniques we introduce in this paper are most useful as simple, practical information to inform the process of search and triage, as noted above, rather than as evidence.

## III. Problem Statement

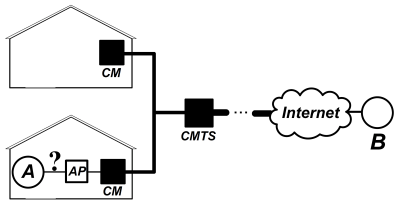


Fig. 1. An illustration of our expected network topology.

Our problem setting is illustrated in Fig. 1. As described in Section II, we begin by assuming that investigators have already identified a peer, denoted as $A$ in the figure, who is a *target* that uploads illegal content to the investigator. *Our challenge is to determine whether $A$ is connected to the home AP via a wireless 802.11 network or via a wired Ethernet.* Investigators, denoted as $B$ in the figure, can make this determination using only traces measured from a remote location.

We assume the AP used by $A$ is connected to the Internet via a cable modem (CM). The coordination system of a regional head-end, known as a Cable Modem Termination System (CMTS), regulates the use of upstream and downstream bandwidth based on $A$'s level of contracted service with the cable network service provider. The CM communicates with the CMTS using the Data Over Cable Service Interface Specification (DOCSIS) [1], [4] protocol stack. In the downstream direction, the CMTS broadcasts data and control frames to a set of CMs. The upstream channel consists of a stream of time slots shared among CMs. Using the DOCSIS, the CMTS replants to CM time-slots requests and grants time-slots to the CM using *MAP messages* every 2ms. Once the CM has acquired time slots from the CMTS, it usually transmits a TCP segment per DOCSIS frame. In the case of congestion, the CM can buffer multiple TCP segments and concatenate the segments in a DOCSIS frame after waiting for a longer time-slot-granting delay. One manifestation of buffering at the CM has been recently noted in *bufferbloat* [6].

We consider measurements taken at $B$, where an investigator can legally make such measurements. $B$ records the inter-arrival times of TCP data packets sent from $A$ during file-sharing uploads to $B$. To provide the most general solution, we assume measurement is from a typical Internet end-point, and not a gateway router or other specialized device. In this case, the investigator is located outside of the cable network. $A$'s traffic will be transmitted through the cable network and then through a number of additional networks before arriving at $B$. In our evaluation scenarios, we assume $B$ has rich, high-speed connectivity to the Internet.

### A. Factors Affecting TCP segment spacing and burst size

Since we will use the inter-arrival times between segments to distinguish between wired and wireless access in the sender's home, let us next consider how the TCP and DOCSIS protocols shape the time between transmission of $A$'s TCP segments. TCP's sliding window algorithm typically results in *bursts* of packets that are sent back-to-back. i.e., with only short inter-departure times between back-to-back segments. These bursts are then separated by a relatively longer interval of time, while the sender waits for the receiver's ACK.

When the CM transmits segments, the inter-departure time between two segments can be different from those segments' inter-arrival time to the CM. These changes can be small or significant, and can depend on the level of congestion in the cable network. Since the segments' inter-arrival times to the CM follow their departure from the (wired or wireless) access network to the CM, and since our goal is to distinguish between wired and wireless access times based on these inter-arrival times, we will focus on segments whose inter-departure time from the CM closely matches their inter-arrival time to the CM. In Section VI.A we present key insights that allow us to identify segments whose inter-departure time is relatively unchanged from their inter-arrival time.

Several other factors found in a typical setting also affect TCP burst sizes and segment inter-arrival times at the CM.

**P2P application rate limit.** As a file sender, peer $A$ is assumed to always have file data to send. In some cases, a P2P application may use a rate-limiting algorithm [17] to purposefully limit TCP throughput. In this case, TCP might not send segments fast enough to fill the congestion window. Such a rate-limited flow may have a larger number of *smaller* bursts (i.e., fewer segments with short inter-departure times) and a larger number of *longer* inter-departure times than an unconstrained TCP sender. But TCP's burst-followed-by-an-inter-burst-delay behavior - a feature we exploit in our classification - is still observed.

**Multiple flows from A.** A P2P peer often exchanges data with multiple peers simultaneously. Since upstream bandwidth is shared among these multiple flows, each individual flow will experience a lower throughput than in a single flow scenario. This decreased throughput is evidenced in a decreased burst size and increased inter-burst spacing. We find, however, that for accurate classification, we only need determine (by measurement) the throughput of a target flow; the number of competing flows need not be known.

**TCP send buffer size and TCP algorithms.** The TCP send buffer size and TCP algorithm play an important role in determing the burst size. Linux has a large maximum send buffer size and disables Nagle's algorithm by default. Consequently, Linux's burst sizes adaptively change over congestion window or P2P application rate limit algorithms; conversely, Windows's burst size is often equal to its *very small* default send buffer size of 8 KB [11]. Windows buffers and transmits send-buffer-sized data. This buffering of data larger than the MSS (Maximum Segment Size) bypasses Nagle's algorithm, which is enabled by default for Windows. These TCP send buffer sizes can be overriden by P2P applications, including *eMule* and *ktorrent*, and cause a flow to have different burst sizes. (We verified these applications' behavior by examining their source code.)

**Wireless channel contention.** Packets ready for departure from $A$ must gain access to a wired or wireless medium in order to reach the local CM. Significant differences between PHY and MAC protocols of wired and wireless access networks

result in distinguishable distributions of inter-frame arrival (and therefore inter-packet arrival) at the CM; these differences can survive through the Internet as we show in Section V. These differences are easier to detect when local contention for the medium increases, as wireless MACs introduce much greater delays between frames during contention than Ethernet MACs.

## IV. Experimental environment and methodology

This section describes the experimental setting in which we obtained measurement traces. Our experiments do not include results from law enforcement trials. It would violate IRB protocols to experiment on Internet users without consent.

### A. Experimental setting

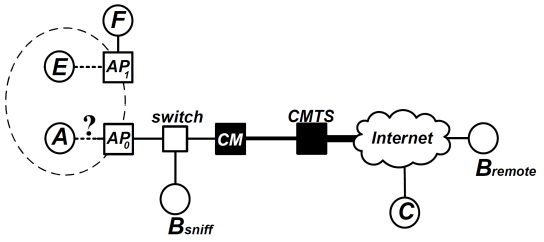


Fig. 2. Our measurement network topology for experiments.

Fig. 2 illustrates the experimental setting for our packet measurements. We have two monitoring points: $B_{sniff}$, and $B_{remote}$. Node $A$ generates TCP traffic to $B_{remote}$ using *iPerf* to transmit TCP data at the maximum rate possible; hence the TCP transmit queue is never starved for data. At $B_{sniff}$ we place a sniffer that captures frames before they are transmitted via the CM. We stress that $B_{sniff}$ is used here *only for experimental research purposes here; as discussed above, our practical forensic setting would preclude making measurements at this point in practice.* Our classification results are performed using *only* traces gathered at $B_{remote}$.

$B_{remote}$ is located at the Univ. of Massachusetts Amherst. $A$ is located in a house in a town near UMass Amherst, using Comcast's residential cable network. The number of hops between $A$ and $B_{remote}$ varies, as determined via *traceroute*. We used a node $C$ as the sink for TCP flows originating at $A$ that compete with traffic to $B$. Node $C$ was located at Purdue University. $AP_0$ is the link type we seek to classify. $A$'s connection to $AP_0$ was either IEEE 802.11g with 54Mbps or 1 Gbps Ethernet in our study. For 802.11g measurement, we located $A$ less than 1m away from the $AP_0$ to obtain the wireless traces least distinguishable from wired traces. For emulating contention traffic for the wireless network, we set up an independent subnet near $A$ as shown in Fig. 2. The subnet consisted of nodes $E$, $F$, and $\text{AP}_1$, all using the same wireless channel as $\text{AP}_0$. The Comcast cable network supports DOCSIS v2.0 with 4 ticks per time-slot as an upstream modulation and 8,160 bytes as a maximum burst frame size. Thus, we can instantaneously see an upstream throughput up to 10 Mbps, although the upstream capacity of a contract is 3 Mbps.

We varied the experimental environment as follows.

**1) P2P application rate limiting algorithm**. For some experiments, instead of *iPerf*, we ran our own emulator that generates TCP data with different inter-departure delays between application chunks and with different chunk sizes.

**2) Competing flows**. We separately evaluated cases of single and multiple competing TCP flows. In the single flow case, $A$ generated a single TCP flow sent to $B_{remote}$. In the multiple flow case, $A$ generated one TCP flow sent to $B_{remote}$ and four competing TCP flows in parallel that were sent to $C$. These five flows generated from $A$ were delivered to a single CM via either wired or wireless access.

**3) Linux vs. Windows**. Our traffic source was either Ubuntu with Linux Kernel 2.6.22 or Windows Vista at node $A$. Linux Kernel 2.6.22 uses CUBIC [7] and Windows Vista uses CTCP [18]. In operating systems, the TCP send buffer size can be globally assigned by the kernel, or applications can individually override its maximum size using *SO_SNDBUF* socket option. Since Kernel 2.4, Linux takes 4 KB as minimum, 16 KB as default, and 4 MB as maximum, and TCP dynamically adjusts the size of the send buffer based on these three values. However, Windows Vista has a TCP send buffer size of 8 KB by default; the same 8 KB default is also found in Windows XP and Windows 7. We observed different inter-arrival time distributions for default or overriden send buffer sizes.

**4) Wireless channel contention (0 Mbps or 10 Mbps)**. $F$ continuously received 0 Mbps or 10 Mbps TCP traffic from $E$ on the same channel as $A$ and aritificially generated 0 Mbps or 10 Mbps contention. In both cases, we characterized non-aritificial wireless channel contention caused by actual in-range wireless transmitters as background traffic near $A$ during our measurement. We used *monitor mode* that captures all the traffic generated with the same or overlap wireless channels across different SSIDs [12]. The measured background traffic was commonly less than 1 Mbps.

Each above experiment setting was performed ten times for 10 sec, 1 min, or 10 mins. We captured a target flow via *tcpdump* at $B_{sniff}$ and $B_{remote}$ and then derived inter-arrival time datasets. We calculated the inter-arrival time as the time interval between the first bit of a first packet and the first bit of a second packet of two back-to-back TCP segments and considered only segments that experienced neither retransmission nor loss. The datasets mainly consisted of inter-arrival times generated during the congestion avoidance phase.

## V. Properties of Classification Features

This section identifies the properties of the sender's TCP and CM access protocol that influence segment inter-departure times as they leave the CM and enter the cable network, in some cases allowing the differences in inter-segment spacing introduced by a wired versus wireless access network at the sending host to be preserved and manifested at the receiver. We then introduce percentiles and entropy of the inter-arrival time distribution for a TCP flow observed at the receiver ($B_{remote}$) as classification features, and discuss how to select inter-arrival times that have preserved the difference between wired versus wireless access networks as a classification feature.

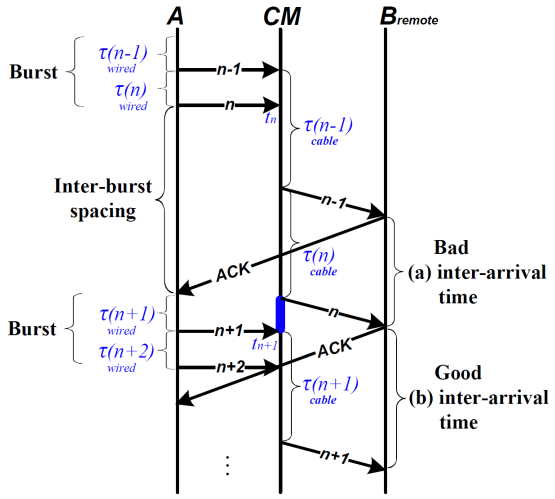


Fig. 3. An example on a TCP flow's time-sequence diagram during the congestion avoidance phase.

### A. Model

In order to preserve the differences in access delays between a wired versus wireless access network at the sender (which will be central to our classification), a segment's inter-departure time from the sending host into the CM should not be greatly changed by the CM. By carefully analyzing the change of the spacing between two successive TCP segments, measured on their arrival to the CM at $A$ and at $B_{remote}$, we will see that a long segment inter-departure time from the sending host into the wired or wireless channel at A, reflecting the characteristics of the PHY and MAC layers at A, can be preserved when the segment arrives at $B_{remote}$. For simplicity, we assume the following about the flow (which we will see generally hold in practice in our experimental evaluation):

- All TCP segment sizes are equal.
- An Internet path traversed by the flow is stable, i.e., that there is little delay variation from the CM to $B_{remote}$.
- The propagation delay at a wired or wireless link is negligible.

Fig. 3 shows a TCP flow's time-sequence diagram observed at $A$, at the CM, and at $B_{remote}$. $A$ generates a series of short segment inter-departure times as a burst (sometimes referred to as a "flight" of segments) followed by a relatively longer inter-departure time. Let $\tau_{wired}(n)$ be segment $n$'s access and transmission delays at a wired link. ($\tau_{wired}$ reflects the characteristics of the PHY and MAC protocols of wired access.) Let $\tau_{cable}(n)$ be the sum of time-slot-granting and transmission delays at the CM for segment $n$.

A **bad inter-arrival time**, shown in Fig. 3(a), occurs when the inter-segment spacing at the receiver has been completely re-shaped (from the original inter-segment spacing on arrival to the CM at A) by the CM; in this case, any difference in access times due to the wired or wireless nature of the access network at A would be lost. A bad inter-arrival time between two segments happens if *(i)* the latter segment is queued before being served at the CM, or *(ii)* the CM concatenates these two segments in a single DOCSIS frame. In Fig. 3(a), segments $n$-1 and $n$ show an example of case *(i)*. When segment $n$ (the latter segment associated with an inter-arrival time) arrives at time $t_n$, the CM is serving segment $n$-1 and hence enqueues segment $n$. After segment $n$-1 has been transmitted, segment $n$ is transmitted and experiences $\tau_{cable}(n)$ delay at the CM. Thus, the (bad) inter-arrival time (a) at $B_{remote}$ equals $\tau_{cable}(n)$.

A **good inter-arrival time**, as shown in Fig. 3(b), occurs when the inter-segment spacing is long enough to be less affected by the CM, thus preserving the difference between wired versus wireless access delay at A when the segments arrive at $B_{remote}$. A good inter-arrival time occurs if no segments are being transmitted (or are queued for transmission) at the CM when the second segment arrives. For instance, suppose that segment $n+1$ (as the latter segment associated with an inter-arrival time) arrives at $t_{n+1}$ when the CM is empty, and is instantly transmitted without being queued. Then, a part of the inter-segment spacing (marked as a *"blue"* bar in Fig. 3) is unchanged by the CM and the difference between wired versus wireless access networks would be preserved in the segment inter-arrival time at $B_{remote}$.

A consequence of the above observations is that the segment inter-arrival time distribution observed at $B_{remote}$ will consist of more good inter-arrival times when the flow sends smaller bursts after longer inter-burst delays and when flow segments rarely experience congestion or concatenation at the CM. A flow having smaller bursts and longer inter-burst delays results in a lower throughput at $B_{remote}$ than a flow with long bursts separated by short inter-arrival times. Thus, in the following discussion, we will characterize a target TCP flow using burst size, throughput, and concatenation rate.

**Burst size observed at $B_{sniff}$.** $\alpha$ denotes the burstiness of a segment arrival process to the CM after leaving the host computer but before reaching the CM. Using the dataset measured at $B_{sniff}$, we calculate $\alpha$ as

$$\alpha = \left( \frac{\text{no. of inter-arrival times below 1ms at } B_{sniff}}{\text{total number of inter-arrival times at } B_{sniff}} \right).$$

In our setting, 802.11g uses neither the RTS/CTS option nor CTS protection but supports frame-burst. Since 802.11g spends 322$\mu$s on transmitting a full-sized TCP segment without random-backoff and frame-burst [5], most short inter-depature times in a burst are less than 1ms at $B_{sniff}$. *We stress $\alpha$ would not be used during a forensic investigation, nor do we employ it in our classification procedure. However, we will find it useful to use $\alpha$ to explain our classifier results, as $\alpha$ characterizes the burstiness of the (unobservable) source.*

**Throughput observed at $B_{remote}$.** We calculate an average $A$-to-$B_{remote}$ throughput observed at $B_{remote}$ and denote it by $T$. We will see that a flow with a lower throughput is more likely to have more good inter-arrival times than a flow with a higher throughput, and thus is more likely to result in more accurate classification. Throughput is an important flow attribute to be considered in assessing the classification accuracy in Step (1) (described in Section II). During Step (3), $T$ would be used to generate a flow observed in step (1).

**Concatenation rate observed at $B_{remote}$.** A flow's concatenation rate (denoted by $\beta$) is the fraction of segment inter-arrival times at $B_{remote}$ that indicate that these two segments were concatenated in a single DOCSIS frame by the CM. We calculate $\beta$ using the dataset measured at $B_{remote}$ as

$$\beta = \left( \frac{\text{no. of inter-arrival times below 1ms at } B_{remote}}{\text{total number of inter-arrival times at } B_{remote}} \right).$$

A receiver can easily identify concatenated TCP segments as those segments having an inter-arrival time of less than 1ms since the CM must wait for at least 2ms to be granted a time slot from the CMTS. As in the case of $T$, $\beta$ would be used to generate a flow having equivalent values of $T$ and $\beta$ as observed in step (1). As we will see, the value of $\beta$ will be an important factor in deciding whether to use percentiles or entropy values of the inter-arrival time distribution observed at $B_{remote}$ as classification features.

### *B. Percentiles and entropy of a distribution observed at* $B_{\text{remote}}$

This subsection discusses and motivates the use of percentiles and entropy of the segment inter-arrival time distribution as classifiers at $B_{remote}$ and the dependence of this use on observed values of $T$ and $\beta$. We will also use the (unobserved) values of $\alpha$ to provide additional insight into the discussion.

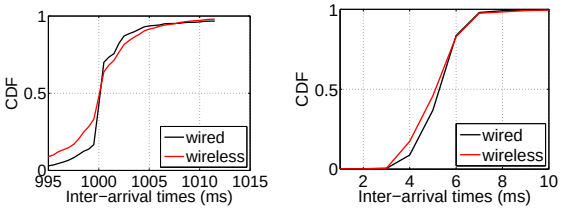


Fig. 4. CDF of segment inter-arrival times at $B_{remote}$, wired versus wireless access at A. (a): $T = 80$ bps ($\alpha = 0.00$); (b): $T = 2.3$ Mbps ($\alpha = 0.50$). No concatenation ($\beta = 0.00$).

**Percentiles.** Fig. 4(a) and (b) plot the CDF of segment inter-arrival times at $B_{remote}$, for the case of wired versus wireless access at A. Fig. 4(a) is for the case of low throughput and low burstiness; these curves are for the case that the sender sends only one 10-byte segment per second. Since 1 sec is much longer than the 100ms RTT in our setting, the CM handles only a single segment at a time ($\alpha = 0$). Fig. 4(b) is for the case of higher throughput and higher burstiness ($\alpha = 0.5$). Here, the sender transmits at the fastest possible rate using *iPerf*, but we observed that no concatenation occurred.

In Fig. 4(a) and (b), the differences between the curves for wired and wireless access suggest those features of the segment inter-arrival time distribution that might best be used as a classifier to distinguish wired from wireless access. Specifically, we note that *the difference between wired versus wireless access networks is manifested most obviously in segment inter-arrival time percentiles lower than median.*

In the case of concatenation (not shown), we observed only a small number of "good" inter-arrival times, since most of segments are buffered and concatenated at the CM. In this case, it is difficult to guarantee that those good inter-arrival times are located at the lower percentiles and (as we will see), percentiles are not a reliable classifier. This motivates our use of entropy (below) as a classifier in these cases.

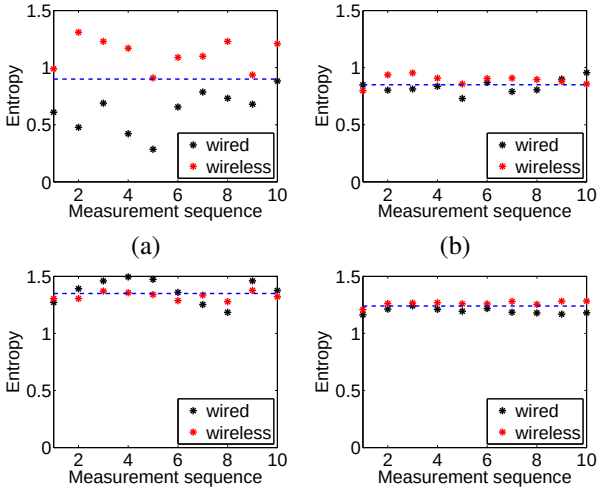


Fig. 5. Entropy of segment inter-arrival times at $B_{remote}$, wired versus wireless access at A. (a): $T = 80$ bps, $\beta = 0.00$, $\alpha = 0.00$; (b): $T = 2.3$ Mbps,$\beta = 0.00$, $\alpha = 0.50$; (c): $T = 2.5$ Mbps, $\beta = 0.17$, $\alpha = 0.83$; (d): $T = 3.5$ Mbps, $\beta = 0.78$, $\alpha = 0.83$.

**Entropy.** Fig. 5 shows the entropy of the inter-arrrival time distribution at the receiver for ten wired and wireless access datasets with minimal and with high concatenation rates, respectively. These datasets were obtained as described in the previous section. Fig. 5(a) and 5(b) were generated in the same manner as Fig. 4(a) and 4(b), respectively. Fig. 5(c) and 5(d) were generated via *iPerf* when a cable network experienced different amount of congestion.

A classifier using entropy would conceptually construct a horizontal line (representing a given entropy value) separating the wired points from wireless points in Fig. 5. In an ideal case, all the entropy values for wireless access would fall above the horizontal line and all the entropy values for wired access would fall below the horizontal line. The blue dotted lines in Fig. 5 are the result of running the entropy-based classifier algorithms that we will discuss in the following section. Fig. 5(a) and 5(d) show that entropy is a good classifier, but Fig. 5(b) and 5(c) show that entropy is not a good classifier.

Fig. 5(a) has the small values of throughput, $\alpha$ and $\beta$ and indicates that there are few bad inter-arrival times. Thus, we conjecture that the inter-arrival time distributions are different enough that inter-arrival time entropy (which considers the entire distribution) is also sufficient (as is the lower percentile of inter-arrival time distributions) to distinguish wired from wireless access.

In Fig. 5(b), the increased throughput (see earlier discussion) results in larger bursts of arrivals (and likely queueing) at the sender, and consequently has fewer good inter-arrival times. Although entropy is not a good classifier in this case, we saw earlier that the lower percentile of the inter-arrival time distribution is a good classifier in this case.

Fig. 5(c) is a scenario similar to Fig. 5(b), except with a non-negligible amount of concatenation, and thus the explanation is similar to that of Fig. 5(b).

In Fig. 5(d), there are even fewer good inter-arrival times than in Fig. 5(b). However, we conjecture that with a higher concatenation rate, many of the bad inter-arrival times are due to concatenation. Recall from our earlier discussion that concatenation leads to nearly constant inter-arrival times between segments at the receiver. Thus, when considering the *difference* in the entropy of the inter-arrival time distribution for the wired and wireless access cases, these identical, deterministic delays due to concatenation cancel each other out. This then leaves a proportionally larger fraction of good inter-arrival times that then contribute to the entropy difference of the wired and wireless access cases.

## VI. Evaluation of Classifier Performance

In this section, we describe the classification algorithms that we use to distinguish wireless from wired access at the sending host $A$ and the experimental procedure for evaluating these classifiers using the traces described in Section IV. We present our empirical evaluation of *decision tree* (DT) classification and verify our conjectures above regarding the circumstances in which different classifiers would work well. We also discuss the relationship of the classification results with other factors discussed in Section III.

### A. Classification algorithms

The decision tree (DT) classification algorithm [14] builds a tree that predicts the value of output based on several features as an input. Each interior node represents a feature, each branch descending from a feature node is one of the possible values for that feature, and each leaf is an output value. During a training phase, a DT is built by selecting the feature making the most difference to the classification at the root and testing a path to a leaf.

In the course of our research, we also ran logistic regression (LR), and support vector machine (SVM) classifiers. The LR classification produces linear decision boundaries between data using a logistic function when the predicted output has only two possible values. SVM projects data into a new space using a kernel function that seeks to create a clear gap between two possible output values and builds a hyperplane to classify data. We found that LR and SVM typically provided similar classification accuracy as DT and thus due to space limitations, we do not report the results for LR and SVM here; see [23] for these details.

### B. Experimental procedure

For each experimental trace (consisting of ten wired and ten wireless datasets taken in the same house using the methodology discussed in section IV), we evaluated DT classification as follows. We trained and cross-validated the classifier using datasets of wired and wireless traffic *"without channel contention"* and investigated various features such as the 25th, 50th, 75th percentiles of the inter-arrival time distribution at the receiver, and the entropy of this distribution. We did not use cross-validation for the following two cases. *(i)* The 10 Mbps wireless access cases were evaluated using the trained model based on wired and wireless traffic without channel contention. We expect to see that the classifier trained with less wireless contention traffic performs well for classification when there is more contending traffic, since the gap between wireless and wired access features would only increase as the amount of interfering wireless traffic increases. *(ii)* A flow generated with competing flows was evaluated using the trained model based on a single rate-limited flow. During Step (3) in Section II, it is not easy for an investigator to generate multiple flow cases (for training purposes). Moreover, an investigator cannot remotely distinguish a single rate-limited flow from a flow generated with multiple competing flows. We will see that a single rate-limited flow's training dataset can indeed be applied to evaluating the multiple flow case.

We quantify classification accuracy using the *true positive rate* (TPR) and *false positive rate* (FPR). TPR denotes the fraction of cases where the access network type is classified as wired given that it is wired. FPR denotes the fraction of cases when the access network type is classified as wired given that it is actually wireless. If the TPR were to be low, the classifier would wrongly argue for accepting the false alibi of a wired user. If the FPR were to be high, the classifier would wrongly argue for not accepting a valid alibi (i.e., that the CP distributor actually did use a wireless network). For our purposes here, we consider it as an *acceptable result* when the TPR is located between 0.9 and 1 and the FPR is located between 0 and 0.1.

The tables in the following subsections show an average of ten classification results for each experimental setting. All the traces shown in the tables were generated from a single house. Each dataset contained at least one thousand inter-arrival times. The inter-arrival times were produced by two successive full-sized (1,460 bytes) TCP segments. But approximately 30% of the inter-arrival times in the Windows with an 8 KB send buffer traces were observed to be transmissions of a burst of five full-sized segments followed by a 892-byte TCP segment. The tables show averages of $\alpha$, $\beta$ and $T$ values of ten datasets for wired and wireless access networks. The rows in the tables show the TPR and FPR when we used the 25th-percentiles and the entropy of inter-arrival time distributions as features. We do not show results for the use of the 50th and 75th percentiles as classifiers, as we found that that they do not work well as classification features. We mark acceptable results using a bold-faced font marked with a star (e.g., **1.0***) in the tables.

### C. Evaluating classifiers for single full-rate flow cases

Let us begin our discussion of our DT classification results by considering what we found to the most difficult classification scenario: classifying a single full-rate flow (i.e., a flow whose sending rate is not constrained by the application, in this case, *iPerf*), when there is no artificially generated wireless channel traffic. Intuitively, we might expect this to be the most challenging case, since in this high throughput scenario, there are a large number of long bursts and minimal inter-burst-delay. In our discussions, we distinguish between the OSes used at A (Linux, Windows), since sometimes our classification results will differ based on the OS type. Also, for the same

| Linux | Trace 1 | | Trace 2 | | Trace 3 | |
|---|---|---|---|---|---|---|
| | wired | wireless | wired | wireless | wired | wireless |
| $\alpha$ | 0.50 | 0.48 | 0.74 | 0.72 | 0.73 | 0.73 |
| $\beta$ | 0.00 | 0.00 | 0.80 | 0.80 | 0.79 | 0.79 |
| **Features** | **TPR** | **FPR** | **TPR** | **FPR** | **TPR** | **FPR** |
| 25th-percentile | **1.0*** | **0.1*** | 0.7 | 0.1 | 0.7 | 0.1 |
| entropy | 0.6 | 0.1 | **0.9*** | **0.0*** | 0.5 | 0.1 |

| Windows | Trace 4 | | Trace 5 | | Trace 6 | |
|---|---|---|---|---|---|---|
| | wired | wireless | wired | wireless | wired | wireless |
| $\alpha$ | 0.83 | 0.81 | 0.83 | 0.81 | 0.83 | 0.81 |
| $\beta$ | 0.17 | 0.17 | 0.78 | 0.78 | 0.78 | 0.78 |
| **Features** | **TPR** | **FPR** | **TPR** | **FPR** | **TPR** | **FPR** |
| 25th-percentile | 0.5 | 1.0 | 0.1 | 0.0 | 0.5 | 0.3 |
| entropy | 0.9 | 0.3 | **0.9*** | **0.1*** | 0.2 | 0.4 |

TABLE I
DT CLASSIFICATION RESULTS FOR A SINGLE FLOW CASE.

send buffer size, we find that Windows and Linux can generate quite different values of $\alpha$ and $\beta$ and that Windows consistently generates traffic with a non-negligible amount of concatenation. This result requires that an investigator identify the OS type and P2P application's send buffer configuration during Step (1). For a detailed discussion of OS-related issues, see [23].

Table I shows the classification results for a single full-rate flow with no concatenation (very low values of $\beta$) and higher concatenation, using the default send buffer size. For the six traces in the table, throughput saturated at a value less than the contracted upstream service rate of 3 Mbps.

**Linux.** Trace 1, a case with no concatenation, shows that the 25th-percentile classifier worked well but that the entropy classifier does not work well. Traces 2 and 3, cases with concatenation, show that the 25th-percentile classifier has a lower TPR (.7) and that neither the 25th-percentile classifier nor the entropy classifier work well in both traces 2 and 3 (although entropy works well as a classifier for trace 2).

In additional traces (not shown here) taken at nine other houses in western MA, we again found that neither classifier worked consistently well for single full-rate flows with high concatenation rates. Also, for cases with low concatenation, the 25th-percentile classifier generally worked well, but did not *always* perform well in densely populated areas, where we might expect more cable network congestion.

**Windows.** Windows (with an 8 KB send buffer) consistently generated large bursts ($\alpha \approx 0.8$) as a result of Winsock buffering, resulting in the CM performing a mild degree of concatenation ($0.17 \leq \beta$), regardless of a cable network's congestion state. Consistent with our discussion in the previous section, we thus see that 25th-percentile classification does not work well for Windows. Additionally, we find that as with Linux, the entropy classifier does not work consistently well for single full-rate flows with high concatenation rates. Our observations for these three Windows traces are consistent with what we observed in experiments run at other locations.

In summary, we found that accurate classification of a single full-rate flow is difficult, with either classifier. We will see shortly that we find much better classification results in other circumstances (e.g., non-full-rate single flows, or the case of multiple flows). Accurate and reliable classification of single full-rate flows remains an open challenge.

### *D. Other Scenarios*

In this subsection, we examine the classification results for the following three scenarios.

**D1) Wireless channel contention.** We consider a scenario when a host transmits a *single full-rate flow* via a wireless access network, but in the presence of other wireless hosts (not attached to the access point under study) that transmit interfering traffic.

**D2) Application-limited rate.** In this case, a flow is limited by the application. Here, we would expect that the inter-burst-delay at a sending host is often greater than the RTT, resulting in a proportionally larger number of good inter-arrival times.

**D3) Multiple competing flows at the sending host, $A$.** The sender $A$ transmits *full-rate multiple flows* to the CM via either the wired or wireless access network.

We use the default send buffer size for Windows and Linux for above three cases. As we will see, these cases show classification performance that is more consistent than that of the full-rate single flow scenario.

| | Linux (Trace 1) | | Windows (Trace 5) | |
|---|---|---|---|---|
| | 10 Mbps wireless | | 10 Mbps wireless | |
| $\alpha$, $\beta$ | 0.46, 0.00 | | 0.81, 0.77 | |
| **Features** | **TPR** | **FPR** | **TPR** | **FPR** |
| 25th-percentile | **1.0*** | **0.0*** | 0.0 | 0.0 |
| entropy | 1.0 | 1.0 | **1.0*** | **0.0*** |

TABLE II
DT CLASSIFICATION RESULTS WITH **10 Mbps wireless cases**

*1) 10 Mbps wireless channel contention:* Table II shows the classification results for scenario D1 (10 Mbps wireless contention) using the classifiers trained by wired and wireless (without contention) access datatsets for Linux and Windows. The values of $\alpha$, and $\beta$ for 10 Mbps wireless datasets are shown in the table. The traces used for training purposes are indicated at the top of columns two and three. For Linux, the 25th-percentile classifier shows perfect performance in classifying 10 Mbps wireless access, a trace with no concatenation. Similarly, for Windows the entropy classifier also shows perfect performance for identifying 10 Mbps wireless datasets. We note that, as we expected, the 25th-percentile classifier does not perform well in this high concatenation case, and the entropy classifier does not perform well for the non-concatenation case.

| | Linux (Trace 7) | | Windows (Trace 8) | |
|---|---|---|---|---|
| | wired | wireless | wired | wireless |
| $\alpha$, $\beta$ | 0.00, 0.00 | 0.00, 0.00 | 0.81, 0.69 | 0.79, 0.68 |
| $T$ (Mbps) | 0.11($\pm$0.00) | 0.11($\pm$0.00) | 0.10($\pm$0.00) | 0.10($\pm$0.00) |
| **Features** | **TPR** | **FPR** | **TPR** | **FPR** |
| 25th-percentile | **1.0*** | **0.1*** | 0.1 | 0.1 |
| entropy | **0.9*** | **0.0*** | **1.0*** | **0.1*** |

TABLE III
DT CLASSIFICATION RESULTS, APPLICATION-LIMITED RATES

*2) Application-limited rates:* Table III shows the classification results for Linux and Windows. Here, traffic was generated by transmitting one full-sized TCP segment every 100ms, thus mimicking the behavior of a rate-limited application. Comparing Tables I and III, we see that classification of rate-limited flows results in more accurate classification than classification of full-rate flows for Linux and Windows. For Linux, both the 25th-percentile and entropy classifiers indeed provide acceptable classification. For Windows, a rate-limited

flow consistently generates $\alpha \approx 0.8$. Thus, the entropy classifier works well but the 25th-percentile classifier does not work well.

| | **Linux (Trace 9)** | | **Windows (Trace 10)** | | **Windows (Trace 11)** | |
|---|---|---|---|---|---|---|
| | wired | wireless | wired | wireless | wired | wireless |
| $\alpha$ | 0.29 | 0.28 | 0.83 | 0.81 | 0.83 | 0.81 |
| $\beta$ | 0.00 | 0.00 | 0.65 | 0.65 | 0.65 | 0.65 |
| $T$ (Mbps) | 0.53 ($\pm$0.00) | 0.54 ($\pm$0.01) | 0.89 ($\pm$0.01) | 0.86 ($\pm$0.03) | 0.89 ($\pm$0.00) | 0.85 ($\pm$0.03) |
| **Features** | **TPR** | **FPR** | **TPR** | **FPR** | **TPR** | **FPR** |
| 25th-percentile | **1.0*** | **0.0*** | 0.0 | 0.0 | 0.0 | 0.0 |
| entropy | 0.0 | 0.0 | **0.9*** | **0.0*** | **1.0*** | **0.1*** |

TABLE IV
DT CLASSIFICATION RESULTS FOR MULTIPLE FLOW CASES.

*3) Multiple flow cases:* Traces 9 and 10, shown in Table IV show the cross-validated classification results for the multiple flow case. Comparing Tables I and IV, we see that classification of the multiple-flow scenario results in more accurate classification than classification of full-rate, single flows. We performed experiments with other multiple-flow traces and consistently found these observations across all of the experiments. The trace 11 column shows classification results for evaluating trace 10 when we use a classifier trained based on trace 11. Trace 11 is a rate-limited flow and has equivalent values of $\beta$ and $T$ with trace 10 as we see in trace 11 column. In this case, we see good classifier performance.

## VII. Related Work

Several past studies have addressed the problem of classifying a sender's access network type using traffic measurements. Wei et al. [22] classified sender network access types into 802.11b, Ethernet, and low-bandwidth access (e.g., dial-up, cable modem, ADSL) categories, using cooperatively transmitted back-to-back UDP packet pairs between sender and receiver. Like our work, Wei et al. took measurements of packet inter-arrival times at the receiver. However, unlike our work, they assume UDP packet pairs are sent by a *cooperative sender*; instead we perform classification without the target's knowledge, using only (P2P application) TCP traffic, with the sender potentially engaging in multiple TCP sessions with multiple receivers.

In subsequent work, Wei et al. [20], [21] monitored ACK packets exiting a university gateway and built a classifier for distinguishing between Ethernet and 802.11b/g traffic. Gateway measurement is not possible in our forensic setting, as this would violate the Wiretap Act. In contrast, we are focused on measurements taken from outside the source's network domain. Additionally, we see our problem as more challenging: we expect bottleneck links in a heavily managed and shaped residential cable modem network to more often obscure distinctions between wired and wireless, as compared to a high-capacity university network.

More recently, Chen et al. [3] address the problem of identifying a suspect's mobile device despite being located behind a wireless AP/NAT router. However, they mark the traffic flow and sniff wireless traffic. This marking and monitoring of traffic broadcast without a warrant is a violation of the Wiretap Act. Moreover, courts require that applications to wiretap traffic meet a significantly higher standard than warrants issued to allow search and seizure of machines located in a residence.

## VIII. Conclusions

This paper proposed legal methods that use remotely measured traffic to disambiguate wired and wireless residential medium access of a criminal in a practical forensic setting, leveraging the difference in inter-arrival times in the wired and wireless access networks. We justified our method's legality based on US law and extensively considered the effect of unknown or hidden factors in a forensic setting (such as wireless channel contention, network stack parameters, and P2P application configuration) on classification performance. We identified 25th-percentile or entropy of inter-arrival times as the best performing features and figured out when and why these features worked reliably or poorly in diverse scenarios.

## References


[1] CableLabs. Data-Over-Cable Service Interface Specifications (DOCSIS).
[2] E. Casey. *Digital evidence and computer crime: forensic science, computers and the Internet*. Academic Pr, 2004.
[3] Y. Chen, Z. Liu, B. Liu, X. Fu, and W. Zhao. Identifying Mobiles Hiding behind Wireless Routers. In *Proc. IEEE INFOCOM*, pages 2651–2659, Apr 2011.
[4] Cisco.com. Understanding Data Throughput in a DOCSIS World.
[5] M. Gast. *802.11 Wireless Networks: The Definitive Guide Creating and Administering Wireless Networks*. O'Reilly Media, 2002.
[6] J. Gettys and K. Nichols. Bufferbloat: Dark buffers in the internet. *Queue*, 9(11):40:40–40:54, Nov. 2011.
[7] S. Ha, I. Rhee, and L. Xu. Cubic: a new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74, July 2008.
[8] O. Kerr. *Computer Crime Law*. Thomson/West, 2006.
[9] D. Kravets. Wi-Fi–Hacking Neighbor From Hell Sentenced to 18 Years. Wired Magazine (Threat Level) http://www.wired.com/threatlevel/2011/07/hacking-neighbor-from-hell/, July 2011.
[10] M. Liberatore, R. Erdely, T. Kerle, B. N. Levine, and C. Shields. Forensic Investigation of Peer-to-Peer File Sharing Networks. In *Proc. DFRWS*, August 2010.
[11] Microsoft.com. Sending small data segments over tcp with winsock.
[12] Microsoft.com. Network Monitor 3.4, 2011.
[13] R. P. Mislan, E. Casey, and G. C. Kessler. The growing need for on-scene triage of mobile devices. *Digital Investigation*, 6(3-4):112–124, 2010.
[14] S. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Prentice Hall.
[15] R. Shore. Pedophiles exploiting wireless loopholes. The Vancouver Sun, http://www.canada.com/vancouversun/news/story.html?id=cff3073b-ceea-4ba4-877f-d020715358e9, February 13 2007.
[16] D. Sicker, P. Ohm, and D. Grunwald. Legal issues surrounding monitoring during network research. In *Proc. ACM IMC*, pages 141–148, Oct. 2007.
[17] M. Siekkinen, D. Collange, G. Urvoy-Keller, and E. Biersack. Performance limitations of ADSL users. In *Proc. PAM Conference*, pages 145–154, 2007.
[18] K. Tan and J. Song. A compound tcp approach for high-speed and long distance networks. In *In Proc. IEEE INFOCOM*, 2006.
[19] U.S. General Accounting Office. File-Sharing Programs – Child Pornography Is Readily Accessible over Peer-to-Peer Networks. GAO-03-537T. Statement Before Congress of Linda D. Koontz, March 2003.
[20] W. Wei, S. Jaiswal, J. Kurose, and D. Towsley. Identifying 802.11 Traffic from Passive Measurements Using Iterative Bayesian Inference. In *Proc. IEEE INFOCOM*, April 2006.
[21] 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. In *Proc. ACM IMC*, pages 365–378, Oct. 2007.
[22] W. Wei, B. Wang, C. Zhang, J. Kurose, and D. Towsley. Classification of access network types. In *Proc. IEEE INFOCOM*, pages 1060–1071, March.
[23] S. Yang, J. Kurose, and B. Levine. Disambiguation of residential wired and wireless access in a forensic setting. *Tech. Rep. UM-CS-2011-032, U. of Massachusetts Amherst*.