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*—Law enforcement investigates thousands of cases each year of trafficking in images of child exploitation on P2P file sharing networks, such as BitTorrent. These and many other Internet-based crimes lead back to a home from an IP address by way of subpoenaed ISP billing records. Having identified the home associated with a specific IP address, investigators enter with a search warrant. An informative first step is to determine if the user's open Wi-Fi access network or the closed wired Ethernet network was likely used for trafficking, and therefore, in the latter case, whether a resident user is the responsible party.**

In this paper, we propose methods that use remotely measured traffic to disambiguate wired and wireless residential medium access. We place our work in a practical forensic setting by evaluating our approaches using only data that can be gathered legally from p2p networks before a warrant or wiretap is required. Our techniques work across the Internet by estimating the per-flow distribution of inter-arrival times for different home access network types, as well as the location at which remote measurements are made. We observe that the change of inter-arrival time distribution is subject to several residential factors, including wireless channel contention and differences between OS network stacks. 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 remote traffic measurements.

I. Introduction

Law enforcement investigate thousands of cases each year of trafficking in images of child sexual abuse on P2P file sharing networks, such as BitTorrent and Gnutella [15, 16]. The goal of these and related network-based criminal investigations is to gather sufficient evidence to support a court-issued warrant to enter and search an associated residence.

Having identified the home associated with a specific IP address, investigators must determine if a wired Ethernet network was used, and therefore whether a resident user is more likely to be the responsible party, or if the home's open Wi-Fi access forms a justifiable alibi. Indeed, the ubiquity of open APs has created a new avenue for persons to act with anonymity in many crimes. Those who wish to possess digital contraband can simply drive around looking for open access, download, and then leave; this has been a documented practice for years [8, 12].

In this paper, we investigate methods that use remotely measured traffic to disambiguate wired and wireless residential medium access. 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. Our goal is to provide information to investigators as they execute a search warrant inside a home. Homes today typically contain many computers and networked devices; limiting the search to a smaller set of computers (for example, only those connected by Ethernet) can save valuable time on scene. Off-scene, backlogs of six months are typical for criminal forensics labs, and the easiest way to reduce the queue is to not add to it using good triage [10]. Similarly, having information about the type of access network used can be beneficial for the interview of a suspect. Solving a crime by an admission is the most efficient resolution; investigations that can leverage knowledge beyond what is expected by a suspect are more likely to achieve such an outcome.

We place our work in a practical forensic setting by constraining our analysis to only *plain view* data that can be gathered legally from P2P networks before a warrant or wiretap is required (in the US) [17]. However, our results are intended to inform and improve the search and seizure process and not to obtain evidence, not even to support the warrant application itself.

To address these challenges, we offer a number of contributions. We present a model of traffic sent from a wired or wireless device, via a cable modem, and received by a remote measurement endpoint. Our model illuminates and explains the results of using several different classifiers to disambiguate traffic being sent via wired and wireless access networks. We evaluate the efficacy of these classifiers using real network traces in a variety of settings. Our results show 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 classifier that uses only measurements of a flow's *(i) packet inter-arrival time entropy* and *(ii) inter-arrival time at the median or 25th percentile*.** For Linux end-points, the classifier achieves a true positive rate of TPR=1.0 and false positive rate of FPR=0.0. For some file-sharing Windows applications, we can classify with TPR=1.0 and FPR=0.0. For other Windows applications, our results achieve TPR=0.9 and FPR=0.0 but can be poor sometimes, though we understand the cause.
- **Our classification accuracy is indirectly affected by the amount of burstiness at the traffic source.** Bursts

of packets that arrive at the cable modem are likely to be concatenated into a single, large link-layer frame sent to the headend of a cable network. This concatenation removes all inter-packet delay information, lowering classification accuracy using median or 25th percentile but improving classification accuracy using entropy.

- **The primary cause of burstiness in our experiments was the very small, 8KB send buffer enforced by default on Windows[1].** With default socket settings, unfortunately, we see FPR=0.8 for Windows if concatenation occurs less. However, because the small buffer impedes bulk-transfer performance, some file sharing applications, such as *eMule* and *ktorrent*, set the send buffer higher, which increases our accuracy to FPR=0.0.
- **Our classifier must be trained for the upstream throughput at the target.** We can select the correctly trained classifier according to the observed upstream throughput. We consider the case of both single and multiple P2P flows from the source, but finds that this distinction does not affect our results; only the upstream throughput of each flow has an impact on the classifier.
- **Higher contention for a wireless channel at the target greatly affects classification accuracy, though this can be overcome.** High wireless contention allows a remote peer to clearly distinguish wired and wireless traffic. The amount of contention is unknown to a remote investigator, but our results show that training the classifier with the more challenging, low-contention scenario allows high accuracy in the high contention case.
- **Measurements from points "near" the target do not guarantee better classification results.** For example, remote measurements taken at another home attached to the same cable network as the source actually performs worse than measurement performed from a well-connected server 13 Internet router hops away from the source.

These findings suggest that it is difficult at best to find a foolproof classifier that works well in all scenarios for remotely identifying a target's network access type. Our goal is thus 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. Moreover, we demonstrate both the restrictions and advantages provided by the setting of criminal investigations.

The remainder of this paper is organized as follows. In Section II, we discuss the legal and practical issues that provide the background and motivation for the particular problem addressed in this paper. We also discuss related past research in the network measurement community. In Section III, we define the problem setting, the classification problem, and the application, network, protocol and environmental factors impacting our work. In Section IV, we describe the classification algorithms evaluated and the experimental environment in which our evaluation is made. Section V analyzes classification results for the case that the measurement point is a well-connected remote server; we identify and discuss a set of predictive features that can be used to infer network access type, and the scenarios in which they can be accurately applied. We also conduct experiments by locating a monitoring point on the same cable network as the target, and again discuss the classification accuracy. In the last section, we summarize our conclusions.

[1] See http://support.microsoft.com/kb/214397/EN-US.

II. Background

Our techniques and goals are related to many previous works on network measurement, which we review below. Unlike much related work, we meet the requirements of practical criminal investigations. And our design and evaluation are in the context of forensics and relevant U.S. law. In particular, the mechanisms we use are within the restrictions placed on law enforcement before a warrant is issued.

A. Legal and Practical Issues

The problem we address is motivated by thousands of investigations of P2P trafficking of child pornography (CP) performed by law enforcement using our forensic tools [9]. The general procedure for these cases is as follows. Investigators search for content on P2P networks. Items found are downloaded and typically a sufficient foundation for *probable cause* for a magistrate-issued search warrant of a residence associated with the IP address. Once on scene, a search begins for evidence associated with child pornography, which might not be the previously downloaded content. The new evidence found is then used to support a criminal trial for receipt, possession, or distribution of CP.

The process 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 browsing, smart phones, desktops, and laptops. Investigators have three main *triage* aims: *(i)* reducing the numbers of devices that must be examined on-scene since warrants are time-limited; *(ii)* 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 *(iii)* 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. Because our work is posed for use during the execution of a search warrant, it makes sense to train the classifiers only once in the home. The collection of data can take place when CP is downloaded as evidence supporting the warrant. The training process can takes minutes with a preconfigured program and a laptop with both wired and wireless interfaces. It would only reduce accuracy to pre-train a classifier from general Internet scenarios.

A number of legal issues restrict the initial process of gathering the data we use to infer a target's medium access type [5, 7]. First, US courts have found consistently that users of P2P networks have no such reasonable expectation in a p2p

file sharing network; see *U.S. v. Breese, 2008 WL 1376269* Such networks can be investigated by law enforcement without a warrant or 4th Amendment protections when acting as a peer. If instead of being a peer, investigators intercepted this traffic from anywhere else on the Internet, it would violate the *Wiretap Act* (18 U.S.C. §2510-2522).

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)*. Recently, in *U.S. v. Gabel, 2010 WL 3927697* the court ruled that software designed for law enforcement to monitor activity on P2P networks does not violate Kyllo 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, below, that has been motivated by network monitoring and measurement is also governed by several US federal laws. Sicker et al. [13] 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.

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.

B. Related Work

Several past studies have addressed the problem of classifying a sender's access network type using traffic measurements.

Wei et al. [20] classified sender network access types into 802.11b, Ethernet, and low-bandwidth access (i.e., 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 using only (P2P application) TCP traffic, with the sender potentially engaging in multiple TCP sessions with multiple receivers.

In subsequent work, Wei et al. [18, 19] monitored ACK packets exiting a university gateway and built a classifier for distinguishing between Ethernet and 802.11b/g traffic. They used expectation maximization (EM) inference and sequential hypothesis testing and examined TCP traffic generated from within the university. Gateway measurement is not possible in our forensic setting, as this would violate the Wiretap Act.

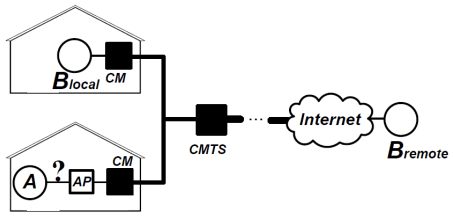

Fig. 1. An illustration of our expected network topology.

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. [6] address the problem of identifying a suspect's mobile device despite being located behind a wireless AP/NAT router. As we do, Chen et al. start from the assumption that an investigation results in discovery of a public IP address associated with some crime, which leads to a residence via billing records. To determine which of several devices behind the AP is the computer they seek, they mark the traffic flow and sniff the wireless traffic. This monitoring of traffic broadcast by the router without a warrant is a violation of the Wiretap Act. Even if the traffic is unencrypted, a warrant is required [7]. But if a warrant is obtained, then the marking is unnecessary: with a warrant in hand, law enforcement can inspect machines directly, or install spyware (covertly) on machines to track live traffic without inference [17]. Moreover, courts require that applications to sniff traffic otherwise protected by the Wiretap Act meet a significantly higher standard than warrants issued to allow search and seizure of machines located in a residence.

III. Problem Statement

Our problem setting is illustrated in Fig. 1. We begin by assuming that investigators have already identified a peer sharing CP or involved in some other crime. The peer, denoted as A in the figure, is now a *target* that uploads illegal content to the investigator. The target accesses the Internet through an access point (AP) located in the home. *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 information gathered from measurements made at a location remote to A.

We assume the AP used by A is connected to the Internet via a *cable modem* (CM). The CM communicates with the coordination system of a regional head-end known as a *Cable Modem Termination System* (CMTS). The standard protocol stack specifying communication between the CM and CMTS is *Data Over Cable Service Interface Specification* (DOCSIS) [4]. The DOCSIS protocol supports full-duplex communications. In the downstream direction, a CMTS broadcasts data to a set of

CMs, but the upstream channel is modeled as a stream of time slots; the CMTS grants time slots to each CM. The CMTS regulates the use of upstream and downstream bandwidth based on A's level of contracted service with the cable network service provider.

We evaluate two locations from which the investigator, B, can legally make measurements. Moreover, 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. Accordingly, the two locations we examine are as follows (see Fig 1.)

- $\mathbf{B_{local}}$: In this case, the remote peer is connected to the same residential cable network as A, but at a different residence (e.g., access purchased by the investigator). The remote measurement point is thus "near" A, both in terms of the number of intervening router hops and in terms of physical distance. Traffic from A to B_{local} will be routed through a small number of cable network routers (and a final CM) before reaching B_{local}.
- $\mathbf{B_{remote}}$: In this case, the remote peer is located outside of the cable ISP network. A's traffic will be transmitted through the cable network and then through a number of additional networks before arriving at B_{remote}. In our evaluation scenarios, we assume B_{remote} has rich, high-speed connectivity to the Internet.

In both cases, B simply records the inter-arrival times of TCP packets sent from A. We expect the TCP stream to be offered by A as part of a p2p file sharing application. We discus the amount of traffic and measurement duration in Section IV.C.

A. Factors Affecting Accurate Classification

In this setting, a number of critical application, protocol, network, and environmental factors might affect the inter-arrival time distribution of A's TCP segments as measured by B. Below, we list these factors, and denote which must be accounted for by a classifier. The order is based on the sequence of protocol mechanisms encountered as data flows from the application and into the Internet.

- **A's application data rate.** Our setting involves file sharing applications that attempt to maximize transfer rates (i.e., always have data to send), and in these cases, the TCP transmit queue plays an important role in determining the outgoing inter-spacing of packets. Some P2P applications use rate-limit algorithms [14], and in these case, if throughput is severely and purposefully limited by the application, we expect to encounter the same challenges we face with purposefully small TCP send buffers, as discussed later.
- **OS's TCP/IP network stack**. TCP's sliding window algorithm typically results in "flights" of packets that are sent back-to-back with only short inter-departure times. These flights appear at the local cable modem as burst of packets, affecting the CM's upstream transmission policy and shaping the distribution observed by B.
- **Multiple flows in node** A. P2P applications typically exchange data with multiple peers simultaneously. These competing TCP flows can affect the inter-arrival time distribution of packets within each individual flow due to multiplexing at routers. However, we find that we only need to determine (by measurement) the data rate of the of flow being monitored in order to perform accurate classification; the number of interfering flows need not be known.
- **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 the medium and access protocols 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 the next section. These differences are easier to detect when local contention for the medium increases: wireless MACs introduce much greater delays between frames during contention than Ethernet MACs.
- **Cable network's characteristics.** At the CM, packets will enter a queue while waiting for upstream bandwidth. The inter-arrival time distribution will be affected most significantly by the upstream transmission policy of the DOCSIS protocol [2, 4]. A CM does not transmit a DOCSIS frame until it has acquired time slots from the CMTS. Therefore, a CM might *increase* the time between two successive TCP segments (in two separate Ethernet frames) if a delay occurs before a transmission time-slot is granted. The CMTS grants time-slots using *MAP messages* every 2ms, and therefore packets containing TCP segments experience at least a 2ms time-slot-granting delay. The time-slot-granting delay can be more than 2ms for high upstream loads. Furthermore, a CM can *decrease* the inter-packet spacing by concatenating multiple TCP segments into a single DOCSIS frame. The maximum number of concatenated TCP segments in a DOCSIS frame is limited by the maximum burst size of a cable network, with the modulation and frequency deciding the maximum burst size. When back-to-back TCP segments are concatenated into a single DOCSIS frame, their inter-arrival time (at the CMTS) becomes effectively zero.

IV. Experimental environment and methodology

In this section, we describe the different classification algorithms that we use to distinguish wireless from wired access at a target A. The classifiers use packet inter-arrival times measured at a remote point B. We then describe the experimental setting in which we obtained the measurement traces. In the following two sections, we evaluate these classifiers using the traces.

A. Classification Algorithms

The decision tree (DT) classification algorithm [11] builds a tree using the concept of information entropy. A decision tree consists of attributes, branches, and predicted output values. Each node (except for leaf nodes) represents an attribute, and each branch descending from an attribute node corresponds

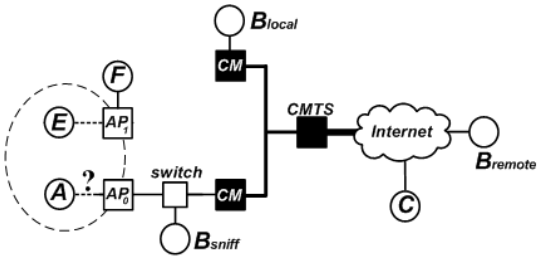

Fig. 2. Our measurement network topology for experiments.

to one of the possible values for that attribute. Leaves denote the predicted output values. Decision trees take as input data described by a set of attributes and return the predicted output value for the input by following the branch condition from the root to a leaf node. The predicted output value can be discrete or continuous. Learning a discrete-valued function is called classification; learning a continuous function is called regression. A decision tree is built by selecting the best attribute (making the most difference to the classification) at the root and testing a path to a leaf using training sets. Decision-tree classification is best suited to problems of classifying data having nonlinear relationships and useful for learning functions having more than two predicted output values.

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 make a clear gap between two possible values for the predicted output 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 classification results for LR and SVM here; see Appendix for these details.

B. Experimental setting

Fig. 2 illustrates the experimental setting for our packet measurements. We have three monitoring points: B_{sniff}, B_{local}, and B_{remote}. Node A generates TCP traffic to B_{local} and 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 purposes; as discussed above, our practical forensic setting would preclude making measurements at this point in practice. We measure packet inter-arrival times at B_{sniff} so that we can better understand how packet inter-arrival times change from their transmission from A to their reception at B_{local} or $B_{remote.}$ Classification is performed using *only* traces gathered at either B_{local} or B_{remote}.

A and B_{local} are located in a house in Amherst, MA using Comcast's residential cable network. B_{remote} is located at the Univ. of Massachusetts Amherst. B_{local} was 3 hops from A; and B_{remote} was 13 hops away from A, 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 or 1 Gbps Ethernet in our study. For emulating contention traffic for the wireless network, we setup an independent subnet near A as shown in Fig. 2. The subnet consisted of nodes E, F and AP_1, all using the same wireless channel as AP_0.

The Comcast cable network supports DOCSIS v2.0; the upstream and downstream capacities of a contract are 3 Mbps and 11 Mbps, respectively. The cable network's upstream modulation uses 4 ticks per time-slot and supports the maximum burst size of 8,160 bytes. (The maximum burst size can be verified by incrementally putting UDP traffic until it reaches maximum burst size.) The peak upstream throughput is therefore roughly 10Mbps: one tick is equal to 6.25 microsec, and one tick transfers 8bytes. Five 1,460-byte TCP segments can therefore be concatenated in a single DOCSIS frame during such bursts.

We varied the experimental environment as follows.

- **Single or multiple flows**. In order to understand the effects of competing flows at A, we evaluate single and multiple flows cases separately. In the single flow case, A generated a single TCP flow destined for B_{local} or B_{remote}. In the multiple flow case, A generated one TCP flow destined for B_{local} or B_{remote} and four competing TCP flows in parallel that were sent to C.
- **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.
- **Linux vs. Windows**. For evaluating the effects of different OS network stacks, we used either Ubuntu with Linux Kernel 2.6.22 or Windows Vista at node A. Linux Kernel 2.6.22 uses the Cubic TCP algorithm and has maximum send buffer size of 4MB by default. Windows Vista uses its own TCP/IP stack [1] and supports receive window auto-tuning: a sender determines the optimal receive window size by continually measuring the bandwidth-delay product. Windows Vista has TCP send buffer size of 8 KB by default; the same 8 KB default is also found in Windows XP and Windows 7 beta. Since some P2P applications such as *eMule* and *ktorrent* override the send buffer size with their own default size, we performed Windows Vista's experiments with send buffer sizes of 8KB and 4MB. (We verified these applications' behavior by examining the source code.)
- **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.

To characterize wireless channel contention caused by in-range, third-party interfering wireless transmitters in our experimental environment, we measured background traffic near A in *monitor mode* [3]. In monitor mode, all traffic using the same or overlap wireless channels across different SSIDs is captured. The measured background traffic was commonly between 60 and 120Kbps.

C. Experimental procedure

Each experiment consisted of running *iPerf* or a simple application-rate-limit emulator for 10 seconds and gathering trace data as discussed above. We ran each such experiment ten times. Then, we computed per-flow inter-arrival time datasets using *tcpdump* at the monitoring point. We calculate 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 sequential TCP segments. We considered only segments that experienced neither retransmission nor loss.

For each dataset across all scenarios, we evaluated each classifier as follows.

1) **Training phase**. We trained classifiers using datasets of wired and wireless (without interference) traffic for different experimental settings. We investigated various features such as 25th, 50th, 75th percentiles, entropy of inter-arrival times datasets and various combinations of multiple features. For different features, we ran 10-fold stratified cross validation[2] in order to reduce the data dependency of evaluation results on features. The values we report in the following sections are each an average of ten classification results. We hypothesized that classification of wireless access with no interference would be the most difficult case to distinguish from wired access, since the gap between wireless and wired access features would only increase as the amount of interfering wireless traffic increased. Therefore we used wireless traces with no interference for classifier training.
2) **Test phase**. We evaluated the trained model using several testsets: *(i)* test sets that had the same characteristics (defined in section V, e.g., the burstiness of a packet arrival process to a CM, concatenation rate, and upstream throughput) as the models training sets but being generated with10Mbps of wireless contention, *(ii)* testsets having the same characteristics as the models training sets but being generated with P2P rate limiting, and *(iii)* testsets having completely different characteristics. We also report an average of ten classification results for test cases.

We will quantify training accuracy using the True Positive Rate (TPR) and False Positive Rate (FPR) as metrics. 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.

V. Evaluation of Classifier Performance

In this section, we present our results of evaluating the efficacy of DT classification. We use the set of inter-arrival times measured at B_{remote} and B_{local} to distinguish between senders whose access to the network is either via wireless 802.11 or wired Ethernet, identifying the best feature sets to precisely build parametric classifiers.

[2]For stratified cross validation, the filtered feature set is randomly reordered and then split into 10 folds of equal size, and the folds contain the same proportions of wired and wireless classification features. In each round, one fold is used for testing and the other nine folds are used for training the classifier.

A. Metrics for characterizing datasets

We denote as α the burstiness of a packet arrival process to a CM on average. We calculated α for datasets measured at a sniffer at A, allowing us to characterize the sender's traffic before it reaches the CM but after leaving the host computer. Based on our observation that most of inter-arrival times of packets in a burst measured at a sniffer were less than 1ms, α is defined as follows:

$$\alpha = \left(\frac{\text{number of inter-arrival times less than 1ms at } B_{sniff}}{\text{total number of inter-arrival times at } B_{sniff}} \right)$$

To characterize how the cable network shapes inter-arrival times distributions, we denote β as the concatenation rate. The concatenation rate is the fraction of packets that were concatenated in a single DOCSIS frame. A receiver can easily identify concatenated TCP segments as those segments having an inter-arrival times of less than 1ms, since a CM must wait for at least 2ms to obtain a time slot from the CMTS. β is defined as follows:

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

We use upstream throughput as the third metric. The upstream throughput is the average throughput over a 10-second interval. We calculate β and the upstream throughput on datasets measured at a receiver only.

The value of α monotonically increases with the growth of the TCP congestion window size and the application *chunk* size, unless they are constrained by a capped send buffer size (or a receiver window constraint). We observed different values of α for different operating systems and different chunk sizes. Linux with *iPerf* attempts to transmit maximum chunks that are not greater than 1,460 bytes. It produced α values of $0.29, 0.50,$ and 0.77 when the upstream throughputs of the observed flow were 0.5 Mbps, 2.3 Mbps, and 3.5 Mbps, respectively. For large application chunks of 5 Kbytes per 15 ms, Linux produced $\alpha = 0.88$ with the upstream throughput of 2.3 Mbps; Windows with a default send buffer size of 8 KB, commonly produced around $\alpha \approx 0.8$ for different upstream throughputs and different application chunk sizes. For Windows with a large send buffer size of 4 MB, we observed using *iPerf* that $\alpha = 0.66$ with an upstream throughput of 3.5 Mbps. Windows transmits 8 KB data in flight by splitting five 1,460 byte-sized and one 892 byte-sized TCP segments, where the maximum transmission unit (MTU) of an Ethernet frame is 1,500 bytes. We see five short inter-arrival times and a long inter-arrival time at a sniffer; here, α is equal to $5/6$ on average.

Fig. 3 shows the remotely measured inter-arrival times distributions of three cases: two flows with upstream throughputs of 2.3 Mbps and 3.5 Mbps from a Linux machine with wired access, and a third flow of 2.5 Mbps upstream throughput for Windows with wired access. The distribution of the 2.3 Mbps

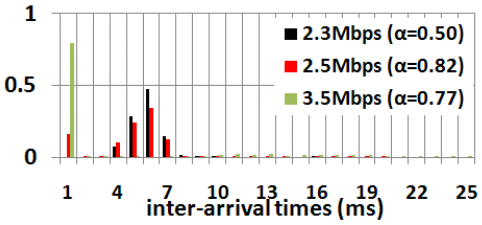

Fig. 3. B_{remote}: PDF of wired access for Linux with the upstream throughputs of 2.3 and 3.5Mbps and PDF of wired access for Windows with the upstream throughput of 2.5Mbps (bin size = 1ms)

case with $\alpha = 0.50$ shows that concatenation does not occur as there are no interarrival times of less that 4 ms in this case. The distributions for 2.5 Mbps and 3.5 Mbps flows show two different concatenation cases. The distribution for the 2.5 Mbps flow has $\beta = 0.16$ under the same conditions as the 2.3 Mbps flow except that the value of α here is exceedingly high - $\alpha = 0.82$. In the case of 3.5 Mbps, a CM experiences at least 6ms time-slot granting delay before transmission and concatenates roughly 80% of TCP segments. Three or four full-sized segments concatenated in a DOCSIS frame need 3.6 ms or 4.8 ms for transmission time-slots; a 10 ms interarrival-time means that a CM received time-slots only after at least three MAP messages of 6ms. This delay happens because at high upstream loads,the CMTS reduces the number of contention-based request time-slots and a CM experiences a longer waiting time for time-slots with consequently higher concatenation.

B. Classification results

Tables I and II show DT classification results for Linux and Windows, respectively, for varied datasets measured in different experimental settings. The columns in the tables are datasets with different values of α, β, and upstream throughput. Each column is labeled as showing experiments for single or multiple flows. For the multiple flows cases, we show the total upstream throughput of multiple flows and the upstream throughput of the observed flow destined to B_{remote}. Each entry in the table is an average over ten experiments. The rows in the tables are for different sets of different classification features and feature combinations. We investigated the 25th and 50th percentile value of the inter-arrival time distribution as features, as shown in Tables I and II. Though we tested it, we do not show results for the 75th percentile, as it does not work well as a classification feature for Linux or Windows. We also investigated the entropy of the inter-arrival time distribution as a feature, also shown in Tables I and II, as well as combinations of these features.

Tables I and II, excluding columns (4), (5), show the effect of wireless channel contention on classification results for different datasets. Each dataset's column is divided into the cross validated classification results of a trained model with wired and wireless (without interference) datasets, as well as the the classification results of 10Mbps wireless test sets using the model. Table II(4) shows the classification results of testsets with P2P rate-limiting using the model in Table II(3) and investigates whether we can use the same classifiers for a multiple flows case and a P2P rate-limited case when they have equivalent values of α, β, and an upstream throughput. The model column in Table II(5) shows the cross validated classification results of wired and wireless (without interference) datasets with large send buffer of 4MB.

The evaluation of the DT classifiers shown in Tables I and II results in the following observations:

1) **The best performing features for Linux and Windows are different.** For Linux, Table I shows that 25th percentile generates the best classification performance and entropy does not work well. For Windows with a default send buffer size of 8KB, Table II shows entropy achieves best but observing the difference of individual inter-arrival times (percentiles) does not work well. However, if we change Windows's send buffer size to 4MB, the classification results for different features show a similar pattern as with Linux; percentiles give the best classification performance but entropy does not work.
2) **The classification model trained with wireless features under lower contention can perform well for distinguishing wireless access under higher contention.** Tables I and II (except for Table II(1)) show that the gap between wired and wireless becomes obvious and classification accuracy becomes better (TPR=1, FPR=0) as the amount of interfering traffic increases.
3) **For Linux and Windows without a capped send buffer size, concatenation at the CM causes 25th percentile's classification performance to degrade (TPR=0.7)** as shown in Tables I(2) and II(5).
4) **For Windows with a send buffer capped at 8 KB, classification results improved when there is a large amount of concatenation.** When the amount of concatenation is small ($\beta = 0.16$) as shown in Table II(1), classification performance is severely degraded (FPR=0.8). Otherwise, if concatenation largely occurs as shown in Table II(2), (3), and (4), the classifier using entropy performs well.
5) **For multiple flows cases, we can get accurate classification results if we choose the model having the same values of α, β, and the upstream throughput of an observing flow with test sets**, as shown in Table I(3) and II(3).
6) The datasets with P2P rate limiting can have the same values of α, β, and upstream throughput with the datasets with multiple competing flows. Since a receiver cannot distinguish those cases, we investigate whether to interchangeably use a model. Table II(4) shows the evaluation results of testsets Table II(4) using the model of Table II(3).
7) **The trained model did not work if we used test sets with different values of α, β, and an upstream throughputs from those of training sets.** Due to space limitations, we do not report classification results; see

	(1) 2.3Mbps as a single flow $\alpha = 0.50$, $\beta = 0.00$ send buffer: 4MB				(2) 3.5Mbps as a single flow $\alpha = 0.77$, $\beta = 0.79$ send buffer size: 4MB				(3) 2.3Mbps as multiple flows; the observing flow of 0.5Mbps $\alpha = 0.29$, $\beta = 0.00$ send buffer: 4MB			
	Model 0Mbps		Test set 10Mbps		Model 0Mbps		Test set 10Mbps		Model 0Mbps		Test set 10Mbps	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
25th percentile	1.0	0.1	1.0	0.0	0.7	0.1	1.0	0.0	1.0	0.0	1.0	0.0
50th percentile	0.9	0.2	0.9	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0
entropy	0.6	0.1	1.0	1.0	0.5	0.1	0.6	0.1	0.0	0.0	0.0	0.0
25th percentile w/ entropy	1.0	0.1	1.0	0.0	0.5	0.1	1.0	0.0	1.0	0.0	1.0	0.0
50th percentile w/ entropy	0.9	0.2	0.9	0.0	0.5	0.1	0.6	0	0.0	1.0	0.0	0.0

TABLE I
DECISION TREE CLASSIFICATION RESULTS FOR **Linux**. ALL TEST AND TRAIN DATA IS CAPTURED AT B_{remote}. GREYED ENTRIES REPRESENT THE BEST-PERFORMING CLASSIFICATION FEATURE.

	(1) 2.5Mbps as a single flow $\alpha = 0.82$, $\beta = 0.16$ send buffer: 8KB				(2) 3.5Mbps as a single flow $\alpha = 0.83$, $\beta = 0.78$ send buffer: 8KB				(3) 3.5Mbps as multiple flows; the observing flow of 0.9Mbps $\alpha = 0.83$, $\beta = 0.65$ send buffer: 8KB				(4) 0.9Mbps as a single flow $\alpha = 0.83$, $\beta = 0.65$ send buffer: 8KB		(5) 3.5Mbps as a single flow $\alpha = 0.66$, $\beta = 0.79$ send buffer: 4MB	
	Model 0Mbps		Test set 10Mbps		Model 0Mbps		Test set 10Mbps		Model 0Mbps		Test set 10Mbps		Test set 0Mbps		Model 0Mbps	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
25th percentile	0.5	1.0	1.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.0	0.0	0.8	0.0
50th percentile	0.6	0.1	0.6	0.8	0.8	0.4	1.0	0.9	0.9	0.3	1.0	0.7	0.5	0.2	1.0	0.0
entropy	0.9	0.3	1.0	0.8	0.9	0.1	1.0	0.0	0.9	0.0	1.0	0.1	1.0	0.1	0.0	0.2
25th percentile w/ entropy	0.9	0.3	1.0	0.8	0.9	0.1	1.0	0.0	0.9	0.0	1.0	0.1	1.0	0.1	0.8	0.0
50th percentile w/ entropy	0.9	0.3	1.0	0.8	0.9	0.1	1.0	0.0	0.9	0.0	1.0	0.1	1.0	0.1	1.0	0.0

TABLE II
B_{remote}: DECISION TREES CLASSIFICATION RESULTS FOR **Windows**. ALL TEST AND TRAIN DATA IS CAPTURED AT B_{remote}. GREYED ENTRIES REPRESENT THE BEST-PERFORMING CLASSIFICATION FEATURE.

Appendix for these details.

Except for the worst datasets Table II(1), the classifier can perform well if it satisfies the following three conditions: 1) the proper feature sets should be chosen according to the values of α and β, 2) wireless (without interference) datasets are adopted for training sets, and 3) the classifier's upstream throughput, α and β match those of test sets. In our forensic setting, α cannot be known, unlike β and upstream throughput. In Windows, which uses a capped 8 KB send buffer, α is almost fixed and can be known to a receiver by observing a series of five full-sized TCP segments and a single 892-byte TCP segment at a receiver. Therefore, we need to only match β and upstream throughput for finding the correct classifier. For Linux (and Windows without a large send buffer), we need a method to remotely estimate α for selecting a correct classifier.

C. Analysis

Percentiles work well as classifier features if packet inter-arrival times are shaped mainly by network limitations. Fig. 4(a) and (b) show that the PDF of inter-arrival times for Linux of 2.3 Mbps and 3.5 Mbps. The figure shows that the 25th percentile gives the best classification performance as the distribution of wireless access have longer inter-arrival times than wired.

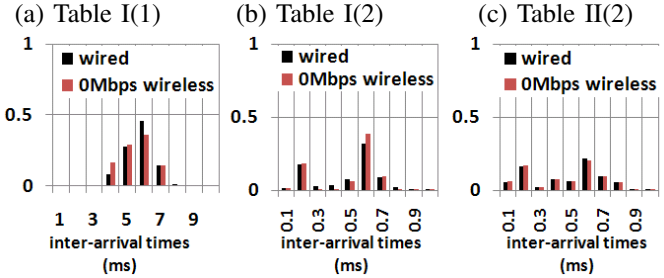

Fig. 4. B_{remote}: PDFs of inter-arrival times for Linux of 2.3 and 3.5Mbps, and Windows of 3.5Mbps

The 25th percentile of wired and wireless inter-arrival times is located at the least overlapping area of PDFs of wired and wireless access. Fig. 4(c) shows the PDF of inter-arrival times for a 3.5 Mbps Windows flow with the send buffer limitation. The figure indicates that the PDFs of wired and wireless access are almost overlapped and the distinction of 25th percentile between wired and wireless is lost.

Now we look at the case when entropy gives the best results. Fig. 5 shows the entropy of ten experiments for wired and wireless (without interference) access of Windows and Linux

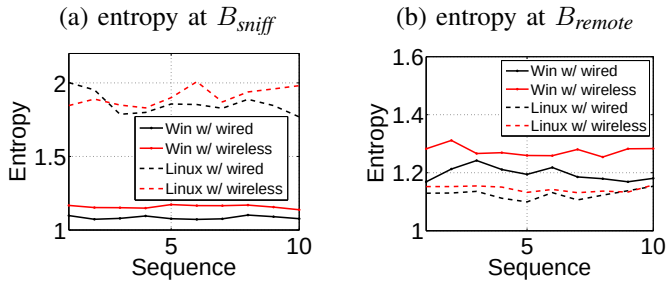

Fig. 5. Entropy of inter-arrival times for Linux and Wiindows of wired and no interfering wireless access at B_{sniff} and B_{remote}

with 3.5Mbps upstream throughput as shown in Tables I(2) and II(2). Fig. 5(a) and (b) are the entropy of inter-arrival times at B_{sniff} and B_{remote}, respectively.

Fig. 5(a) shows that Windows with burst packet transmission has stable and low entropy over ten experiments but Linux's entropy is high and variable over ten experiments. For both B_{sniff} and B_{remote}, the entropy difference between wired and wireless for Windows is obvious over ten experiments as opposed to Linux's small gap. This happens since for burst packet transmission in 802.11 and Ethernet, a sender may transfer several packets without contending for the channel in between and packets via 802.11 experiences more randomness than Ethernet. Especially the entropy difference between wired and wireless can be more clearly preserved if packets are concatenated and do not experience randomness at a cable network. Thereby, we can observe that entropy-based classification results with large β get improved as α increases in Tables I and II.

D. Local measurement results

We discuss the classification results of traces measured at B_{local}. We focus on the concatenation-free case for Linux with a single flow of 2.7Mbps upstream throughput, $\alpha = 0.52$, and $\beta = 0.00$. We compare with B_{remote}'s concatenation-free dataset, Table I(1). At B_{local}, the classification result using 25th percentile degrades as TPR=0.7, and the entropy performs best for 10Mbps wireless test sets (classification results with other features are found in [**?**]). We conjecture that this happens because packet transmission between A and B_{local} experiences the upstream time-slot granting procedure of a cable network in both directions, making the inter-arrival times more exposed to randomness and obscuring the gap in the inter-arrival times between wired and wireless.

VI. Local measurement results

This section discusses the classfication results of traces measured at B_{local} when a sender uses Linux and generates a single flow of 2.7Mbps upstream throughput, $\alpha = 0.52$, and $\beta = 0.00$ and compares with B_{remote}'s concatenation-free dataset, Table I(1). The classifcation result using 25th percentile at B_{local} degrades as TPR=0.7, and the entropy performs best for 10Mbps wireless test sets (classification results with other features are found in [**?**]). We conjecture that this happens because packet transmission between A and B_{local} experiences the upstream time-slot granting procedure of a cable network in both directions, which makes inter-arrival times more exposed to randomness and obscure the gap of inter-arrival times between wired and wireless.

VII. Conclusions

In this paper, we proposed methods that use remotely measured traffic to disambiguate wired and wireless residential medium access in a practical forensic setting. Our methods leverage the difference in inter-arrival times in the wired and wireless access networks. We observed that the inter-arrival times changed with several residential factors, including wireless channel contention, differences between OS network stacks, and P2P applications. We identified several characteristics of the measured traces that influenced the inter-arrival times: the burstiness of a packet arrival process to a CM (α), the concatenation rate (β), and the upstream throughput. For different parameter values, we separately built classifiers using wired and wireless datasets, identified the best feature sets for classification, and evaluated the trained model using test sets having the same values of α, β, and upstream throughput but different values of residential factors such as wireless channel contention, intervention of P2P rate-limit algorithms. We also discussed why classification using 25th percentile or entropy worked well for datasets with particular values of α, β and upstream throughput.

References

[1] http://technet.microsoft.com/en-in/library/bb878127(en-us).aspx.
[2] Understanding data throughput in a docsis world. http://www.cisco.com/application/pdf/paws/19220/data_thruput_docsis_world_19220.pdf.
[3] Network monitor 3.4. http://blogs.technet.com/b/netmon/archive/2010/06/28/network-monitor-3-4-has-released.aspx, 2011.
[4] CableLabs. Data-Over-Cable Service Interface Specifications (DOCSIS). http://www.cablelabs.com/cablemodem/downloads/specs/CM-SP-RFI2.0-I11-060602.pdf.
[5] E. Casey. *Digital evidence and computer crime: forensic science, computers and the Internet*. Academic Pr, 2004.
[6] Y. Chen, Z. Liu, B. Liu, X. Fu, and W. Zhao. Identifying Mobiles Hiding behind Wireless Routers. In *Proc. IEEE INFOCOM*, 2011.
[7] O. Kerr. *Computer Crime Law*. Thomson/West, 2006.
[8] 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.
[9] M. Liberatore, R. Erdely, T. Kerle, B. N. Levine, and C. Shields. Forensic Investigation of Peer-to-Peer File Sharing Networks. In *Proc. DFRWS Annual Digital Forensics Research Conference*, August 2010.
[10] 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.
[11] S. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Prentice Hall, 1995.
[12] 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.
[13] D. Sicker, P. Ohm, and D. Grunwald. Legal issues surrounding monitoring during network research. In *Proc. ACM IMC*, pages 141–148, Oct. 2007.
[14] M. Siekkinen, D. Collange, G. Urvoy-Keller, and E. Biersack. Performance limitations of adsl users: A case study. *Passive and Active Network Measurement*, pages 145–154, 2007.
[15] U.S. Dept. of Justice. National Strategy for Child Exploitation Prevention and Interdiction: A Report to Congress. http://www.projectsafechildhood.gov/docs/natstrategyreport.pdf, August 2010.

[16] 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 Director, Information Management Issues, March 2003.
[17] R. J. Walls, B. N. Levine, M. Liberatore, and C. Shields. Effective Digital Forensics Research is Investigator-Centric. In *Proc. USENIX Workshop on Hot Topics in Security (HotSec)*, August 2011.
[18] 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.
[19] 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 Internet Measurement Conference (IMC)*, pages 365–378, Oct. 2007.
[20] W. Wei, B. Wang, C. Zhang, J. Kurose, and D. Towsley. Classification of access network types: Ethernet wireless LAN, ADSL, cable modem or dialup? In *Proc. IEEE INFOCOM*, pages 1060–1071, March 2005.

APPENDIX

In the appendix, we show LR and SVM classification results at B_{remote}, DT, LR and SVM classification results at B_{local} using all the considered features, and DT classification results for the third test case (i.e., testsets having completely different characteristics) as explained in section IV. Tables III and IV show LR classification results for Linux and Windows, respectively. Tables V and VI show SVM classification results for Linux and Windows, respectively. Table VII shows DT, LR, and SVM classification results at B_{local}. Table VIII shows DT classification results for different models and testsets. In Table VIII(1), we used a trained model of Table II(5) and evaluated the dataset of Table I(2) when the values of α for the trained model and testset are different and the values of β and uptream thoughput for the trained model and testset are equivalent.

	(1) 2.3Mbps as a single flow $\alpha = 0.50$, $\beta = 0.00$ send buffer: 4MB				(2) 3.5Mbps as a single flow $\alpha = 0.77$, $\beta = 0.79$ send buffer size: 4MB				(3) 2.3Mbps as multiple flows; the observing flow of 0.5Mbps $\alpha = 0.29$, $\beta = 0.00$ send buffer: 4MB			
	Model		Test set		Model		Test set		Model		Test set	
	0Mbps		10Mbps		0Mbps		10Mbps		0Mbps		10Mbps	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
25th percentile	1.0	0.1	1.0	0.0	0.5	0.8	0.7	0.1	1.0	0.0	1.0	0.0
50th percentile	0.8	0.1	0.9	0.0	0.7	0.7	0.8	1.0	1.0	0.0	1.0	0.0
entropy	0.7	0.3	0.0	0.0	0.7	0.4	0.8	0.1	0.5	0.3	0.5	0.3
25th percentile w/ entropy	0.9	0.1	1.0	0.3	0.7	0.4	0.7	0.0	1.0	0.0	1.0	0.0
50th percentile w/ entropy	1.0	0.0	1.0	0.0	0.7	0.4	0.7	0.0	1.0	0.0	1.0	0.0

TABLE III
LR CLASSIFICATION RESULTS FOR **Linux**. ALL TEST AND TRAIN DATA IS CAPTURED AT B_{remote}. GREYED ENTRIES REPRESENT THE BEST-PERFORMING CLASSIFICATION FEATURE.

	(1) 2.5Mbps as a single flow $\alpha = 0.82$, $\beta = 0.16$ send buffer: 8KB				(2) 3.5Mbps as a single flow $\alpha = 0.83$, $\beta = 0.78$ send buffer: 8KB				(3) 3.5Mbps as multiple flows; the observing flow of 0.9Mbps $\alpha = 0.83$, $\beta = 0.65$ send buffer: 8KB				(4) 0.9Mbps as a single flow $\alpha = 0.83$, $\beta = 0.65$ send buffer: 8KB		(5) 3.5Mbps as a single flow $\alpha = 0.66$, $\beta = 0.79$ send buffer: 4MB	
	Model		Test set		Model		Test set		Model		Test set		Test set		Model	
	0Mbps		10Mbps		0Mbps		10Mbps		0Mbps		10Mbps		0Mbps		0Mbps	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
25th percentile	0.5	0.3	0.5	1	0.7	0.4	0.7	0.3	0.8	0.1	0.9	0.6	0	0	0.9	0.1
50th percentile	0.6	0.4	0.6	1	0.6	0.2	0.8	0.8	0.1	0.6	0.4	0	0.2	0	1	0
entropy	0.7	0.3	0.7	0.8	0.9	0.1	0.9	0	1	0.1	1	0.1	1	0.1	0.4	0.3
25th percentile w/ entropy	0.7	0.4	0.7	0.8	1	0.2	1	0	1	0	1	0	0.4	0	0.9	0.2
50th percentile w/ entropy	0.7	0.4	0.7	0.5	0.9	0.1	0.9	0	1	0	1	0	1	0.1	1	0

TABLE IV
LR CLASSIFICATION RESULTS FOR **Windows**. ALL TEST AND TRAIN DATA IS CAPTURED AT B_{remote}. GREYED ENTRIES REPRESENT THE BEST-PERFORMING CLASSIFICATION FEATURE.

	(1) 2.3Mbps as a single flow $\alpha = 0.50$, $\beta = 0.00$ send buffer: 4MB				(2) 3.5Mbps as a single flow $\alpha = 0.77$, $\beta = 0.79$ send buffer size: 4MB				(3) 2.3Mbps as multiple flows; the observing flow of 0.5Mbps $\alpha = 0.29$, $\beta = 0.00$ send buffer: 4MB			
	Model		Test set		Model		Test set		Model		Test set	
	0Mbps		10Mbps		0Mbps		10Mbps		0Mbps		10Mbps	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
25th percentile	0.9	0.0	0.9	0.0	0.6	0.8	0.7	1.0	1.0	0.3	1.0	0.0
50th percentile	0.9	0.2	0.9	0.0	0.7	0.9	0.8	1.0	0.6	0.2	0.6	0.0
entropy	0.5	0.0	0.6	1.0	0.4	0.0	0.4	0.0	1.0	0.3	1.0	0.0
25th percentile w/ entropy	0.9	0.0	0.9	0.0	0.5	0.8	0.6	0.0	0.9	0.0	1.0	0.0
50th percentile w/ entropy	0.9	0.2	0.9	0.0	0.4	0.0	0.4	0.0	0.9	0.1	0.9	0.0

TABLE V
SVM CLASSIFICATION RESULTS FOR **Linux**. ALL TEST AND TRAIN DATA IS CAPTURED AT B_{remote}. GREYED ENTRIES REPRESENT THE BEST-PERFORMING CLASSIFICATION FEATURE.

	(1)				(2)				(3)				(4)		(5)	
	2.5Mbps as a single flow $\alpha = 0.82$, $\beta = 0.16$ send buffer: 8KB				3.5Mbps as a single flow $\alpha = 0.83$, $\beta = 0.78$ send buffer: 8KB				3.5Mbps as multiple flows; the observing flow of 0.9Mbps $\alpha = 0.83$, $\beta = 0.65$ send buffer: 8KB				0.9Mbps as a single flow $\alpha = 0.83$, $\beta = 0.65$ send buffer: 8KB		3.5Mbps as a single flow $\alpha = 0.66$, $\beta = 0.79$ send buffer: 4MB	
	Model		Test set		Model		Test set		Model		Test set		Test set		Model	
	0Mbps		10Mbps		0Mbps		10Mbps		0Mbps		10Mbps		0Mbps		0Mbps	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
25th percentile	0.3	0.1	0.3	0.1	1.0	0.8	1.0	0.9	1.0	0.6	1.0	0.6	0.0	0.0	0.6	0.0
50th percentile	0.4	0.0	0.4	0.7	0.6	0.2	0.6	0.8	0.3	0.6	0.5	0.0	0.4	0.1	0.6	0.0
entropy	0.5	0.0	0.5	0.7	0.9	0.1	0.9	0.0	1.0	0.1	1.0	0.5	1.0	0.1	0.6	0.5
25th percentile w/ entropy	0.4	0.1	0.4	0.8	0.9	0.1	1.0	0.0	1.0	0.3	1.0	0.5	1.0	0.1	0.6	0.0
50th percentile w/ entropy	0.4	0.0	0.4	0.8	0.9	0.1	0.9	0	1.0	0.1	1.0	0.5	1.0	0.2	0.7	0.3

TABLE VI
SVM CLASSIFICATION RESULTS FOR **Windows**. ALL TEST AND TRAIN DATA IS CAPTURED AT B_{remote}. GREYED ENTRIES REPRESENT THE BEST-PERFORMING CLASSIFICATION FEATURE.

	2.7Mbps as a single flow $\alpha = 0.52$, $\beta = 0.00$ send buffer: 4MB											
	Decision tree				Logistic regression				SVM			
	Model		Test set		Model		Test set		Model		Test set	
	0Mbps		10Mbps		0Mbps		10Mbps		0Mbps		10Mbps	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
25th percentile	0.7	0.1	0.7	0.0	0.9	0.4	0.9	0.2	0.9	0.4	0.9	0.2
50th percentile	0.6	0.0	0.7	0.0	0.7	0.4	0.8	0.1	0.3	0.0	0.3	0.0
entropy	0.8	0.5	1	0.1	0.5	0.4	0.6	0.0	0.5	0.4	0.5	0.0
25th percentile w/ entropy	0.7	0.1	0.7	0.0	0.8	0.3	0.9	0.2	0.9	0.4	0.9	0.2
50th percentile w/ entropy	0.6	0.0	0.7	0.0	0.7	0.3	0.9	0.1	0.3	0.0	0.3	0.0

TABLE VII
DT, LR AND SVM CLASSIFICATION RESULTS FOR **Linux** ALL TEST AND TRAIN DATA IS CAPTURED AT B_{local}. GREYED ENTRIES REPRESENT THE BEST-PERFORMING CLASSIFICATION FEATURE.

	Model: Table II(5) Testset: Table I(2)	
	TPR	FPR
25th percentile	0.4	0.4
50th percentile	1	0.5
entropy	0.0	0.0
25th percentile w/ entropy	0.4	0.4
50th percentile w/ entropy	1	0.5

TABLE VIII
DT CLASSIFICATION RESULTS FOR THE THIRD TEST CASE. ALL TEST AND TRAIN DATA IS CAPTURED AT B_{remote}. GREYED ENTRIES REPRESENT THE BEST-PERFORMING CLASSIFICATION FEATURE.