Mapping the Urban Wireless Landscape with Argos

Ian Rose and Matt Welsh
School of Engineering and Applied Sciences, Harvard University
{ianrose,mdw}@eecs.harvard.edu

Abstract

Passive monitoring is an important tool for measuring, troubleshooting, and protecting modern wireless networks. To date, WiFi monitoring has focused primarily on indoor settings or ephemeral outdoor studies though wardriving. We present *Argos*, the first urban-scale wireless sensor network designed explicitly to support measurement of ambient WiFi traffic across an entire city. Urban-scale wireless monitoring presents unique challenges due to limited packet-capture ability, heterogeneous traffic loads, and limited backhaul capacity between sensor nodes. Argos addresses these through in-network traffic merging and processing, plus an intelligent approach to coordinated channel sampling by multiple sniffers. Argos provides a rich query interface allowing users to study the complex dynamics of ambient wireless traffic.

We present a detailed evaluation of a 26-node Argos network deployed on streetlights and rooftops around a city, demonstrating its ability to detect and classify wireless access points and clients; monitor Web page usage; detect malicious traffic; track the mobility of WiFi-equipped public transport vehicles; and fingerprint individual users through 802.11 probe request packets.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication; C.2.3 [**Network Operations**]: Network monitoring

General Terms

Design, Experimentation, Measurement

Keywords

Wireless networks, 802.11, outdoor monitoring

1 Introduction

Wireless networks are becoming the *de facto* access network for many Internet users, stemming from the tremendous growth in the use of laptops, smartphones, and other mobile devices. Even many desktop systems come equipped with 802.11 interfaces and can be used without a wired Ethernet connection, and devices such as Apple's Time Capsule permit automated backup over a wireless connection. As a result, the performance and behavior of wireless networks are critical for supporting the Internet. At the same time, growth in wireless network deployments, especially in densely populated urban settings, has been tremendous. A city block might have hundreds of separate access points and thousands of wireless clients. Of course, all of these devices must share limited ISM spectrum, leading to potentially high congestion.

Yet, we have little understanding of what wireless network traffic looks like "in the wild" at urban scales, with many access points and clients interacting. There are many open questions that we would like to answer: What are the characteristics of wireless network traffic? How does traffic vary over space and time? Is there evidence of malware or other malicious traffic traversing the airwaves? Can we characterize user mobility patterns, and what can we learn about individual clients?

In this paper, we propose *Argos*, a city-wide wireless sensor network designed to study wireless network traffic and dynamics over urban scales. Argos collects data from multiple WiFi sniffers mounted on streetlights and rooftops around a city, and performs decentralized trace merging and filtering to support multiple user queries processing the captured traffic. Argos sensor nodes are themselves connected with a backhaul wireless mesh network (in an orthogonal frequency band) to enable scalability and large spatial coverage. Using a network of 26 Argos nodes deployed around Cambridge, MA, we present the first city-wide study of wireless network traffic dynamics, comprising 2630 access points and 65,073 unique clients spanning an area of 9.7 km^2.

Capturing and studying wireless network traffic across an entire city raises many new challenges. The first is dealing with the inherently lossy nature of wireless data capture in this setting. Unlike dense indoor monitoring studies, where 99% or more of the traffic can be captured, in an outdoor sniffer network the typical capture rates are on the order of 5-10%. This requires that sniffers merge their packet traces to increase coverage, and new techniques to infer missing data and recover useful information from an extremely lossy data stream.

SenSys '10, November 3–5, 2010, Zürich, Switzerland.

The second challenge deals with the vastly more complex nature of traffic emanating from the multitude of APs and clients in an urban setting, each operating on different channels with varying amounts of traffic. To maximize capture fidelity, multiple sensor nodes in a vicinity should coordinate their channel changes to "track" interesting traffic patterns and dynamically tune dwell times based on traffic volume.

The third challenge deals with scalability in terms of both spatial coverage and the number of sniffers. It is not practical to assume that sensor nodes have a wired back-channel to the Internet; this would substantially increase cost and limit the number of sites where sensors can be deployed. Rather, we leverage mesh networking over a separate, 900 MHz radio between sensor nodes, in a manner similar to conventional wireless sensor networks. The limited capacity of the backhaul mesh mandates that sensor nodes perform distributed filtering and aggregation of their individual packet streams to avoid overloading the mesh links.

This paper makes the following contributions. First, we describe the architecture of Argos, the first sensor network designed to capture and evaluate ambient wireless traffic at an urban scale. Argos provides a rich programmatic interface enabling multiple concurrent queries to capture and process wireless traffic in different ways. Second, we describe a novel *in-network trace merging* algorithm that substantially reduces the amount of cross-traffic required to merge captured packet traces across multiple sniffers. Third, we present a detailed evaluation of Argos running across a city through several case studies: tracking of popular Web sites and Google searches from different parts of the city; detection of malicious traffic; tracking of public trains and buses; and inferring clients' past mobility patterns based on their wireless network behavior. We demonstrate that, using these techniques, Argos is effective at capturing and processing data from a large number of clients and networks with (a) minimal traffic load imposed on the backhaul network, and (b) increased capture fidelity from coordinated channel hopping and packet merging across sniffers.

We recognize that passive monitoring of wireless networks raises numerous privacy concerns. Indeed, Google has recently found itself under fire for inadvertent packet captures from their Street View cars [43]. Argos anonymizes captured packet traces to protect against obvious user privacy violations, but we recognize that this does not eliminate all risks, and that reconciling the conflicting goals of data collection and user privacy will be an ongoing effort.

2 Background and Motivation

The vast growth of wireless LANs has led to new challenges for characterizing traffic and user behavior, as well as understanding the complex dynamics underlying the interaction between multiple access points and clients. Up until now, studies have focused on either microscopic analyses of individual networks (say, in an office building [23, 34]), wardriving studies that take a static snapshot of network deployments [15, 27], or macroscopic analyses of isolated mesh networks [12, 14]. The performance, behavior, and variation of wireless networks "in the wild" has never been studied at the urban scale.

This trend has created new difficulties for wireless networks. Increasing density can induce severe performance problems when multiple users share the same limited radio spectrum. Given the cooperative nature of 802.11 MAC protocols, it is possible for buggy implementations or malicious users to hog the spectrum. Likewise, the shared nature of the RF medium means that the efficiency of spectrum utilization is a real concern. Malware that blasts a large number of packets or bandwidth-hogging file-sharing applications can cause performance problems for many nearby users. New applications and mobile devices are putting increasing pressure on wireless networks and leading to new user behaviors, such as free-riding on open networks.

There is also a serious concern about privacy and security of wireless LANs. The existing encryption standards, such as WEP and WPA2 [28], have all been shown to have weaknesses [26, 46], and it is possible to break WEP keys in a matter of minutes [17]. Moreover, many networks are unencrypted [19], so absent some form of end-to-end encryption, user traffic travels over the airwaves in the clear. Even in encrypted networks, it is possible to fingerprint individual wireless users based on leaked information, such as 802.11 probe requests [37]. A variety of end-user tools exist to capture wireless traffic, break encryption keys, and perform spoofing, denial-of-service and other attacks [1, 5, 11]. One of the goals of this paper is to highlight the privacy risks that arise with large-scale, coordinated traffic snooping.

2.1 Why Argos?

The goal of Argos is to enable urban-scale monitoring of wireless networks, permitting multiple users to execute queries against the captured traffic. To achieve high spatial coverage, this requires multiple sensor nodes deployed throughout a city that can capture ambient wireless network traffic and perform filtering, aggregation, and trace merging with other sensors. To protect the privacy of users, sensor nodes should anonymize their raw traffic traces to remove sensitive information. To ensure scalability, sensor nodes should not require a wired back-channel, and preferentially use a scalable wireless mesh to communicate with each other and the Internet.

A wide range of potential users would find value in a citywide wireless monitoring network. Wireless networking researchers can conduct detailed studies of traffic dynamics and network performance in a complex urban setting. Captured packet traces can be used to drive new protocol and system designs rather than relying on synthetic traffic that may not exhibit realistic behavior. Security researchers can use Argos to understand privacy and security risks inherent in wireless networks, such as the presence of malicious traffic and privacy leakage through side-channels. Finally, researchers in fields such as sociology and anthropology have an increasing interest in understanding wireless network traffic and user behaviors, such as characterizing the popularity of online media sources or Web search patterns across different parts of a city.

2.2 Challenges

There are many challenges associated with deploying a city-wide wireless monitoring network. Firstly, individual

sniffer nodes often exhibit relatively poor packet capture rates. In a sparse, outdoor deployment, it is unlikely that we will be able to capture packets from a given source (client or access point) with high fidelity. Although many Argos queries focus only on high-level, aggregate observations of wireless network activity, when "drilling down" into a particular source or packet stream is called for, we can improve capture coverage by coordinating between multiple sniffer nodes. For example, multiple sniffers in an area can synchronize their channel hopping schedules to maximize the joint probability of collecting packets from a given source.

Secondly, the system should scale up to many sniffers deployed across a large urban area. It is unreasonable to expect every sniffer node to have a wired connection to the Internet; this would substantially limit the locations at which sniffers could be installed, and increase deployment cost and management overhead. For this reason, Argos makes use of wireless mesh networking to enable connectivity between sniffers and the Internet. The mesh backhaul uses an orthogonal frequency band (900 MHz) to avoid interference with the traffic sources being monitored. However, this gives rise to a related challenge in that wireless mesh networks are typically very bandwidth-constrained, and the total amount of traffic being captured by a set of sniffers may exceed the capacity of the mesh. In our deployment, the backhaul mesh links range from a few Mbps to just a few hundred Kbps. For this reason, it is generally not possible to send the raw packet traces back to a central server for processing; as in conventional sensor networks, filtering and aggregation must be performed within the sniffer network itself.

Thirdly, there is substantial diversity in the ambient wireless traffic present throughout a city, ranging from home networks with a small amount of traffic to large, enterprise-wide wireless LANs supporting thousands of users. The spatial distribution of wireless LANs is highly irregular and temporal diversity is seen across multiple timescales. In addition, increasing diversity of the client device population, including (mostly) static laptops to highly mobile smartphones, makes it challenging to obtain a clear picture of traffic patterns.

2.3 Related Work

The standard approach to studying urban wireless network deployment is *wardriving*, in which a mobile sniffer is used to detect the presence and static properties of deployed wireless LANs [15, 27]. Wardriving studies in numerous cities have revealed an extremely high penetration of wireless networks, and the online Wigle.net database reports over 19 million unique wireless LANs [10]. However, these studies do not typically include any analysis of wireless traffic itself since observations are made only over short periods of time. A related measurement study [19] explored the extent to which wireless clients in moving vehicles can establish Internet connectivity via open access points.

Passive monitoring has been used for detailed measurements of indoor wireless LANs. Jigsaw [23] is a system for diagnosing MAC-layer behavior by capturing detailed packet traces from a large number of densely-deployed wireless sniffers. The authors show that by capturing nearly-complete traces of the wireless activity in a building, a number of analyses can be performed that are difficult to accomplish from a single vantage point. Jigsaw was deployed on 39 sensor pods across a single building. A contemporary system, Wit [34], has similar goals to Jigsaw, but emphasizes the use of a formal language for specifying 802.11 behavior, which is then used to perform inference on the captured packet traces. Wit was deployed on five sniffers to monitor traffic during a conference. The Dartmouth MAP system [42] is designed to detect malicious wireless network activity, such as the presence of rogue access points. In order to monitor all 11 of the 802.11 b/g channels, MAP employs intelligent channel-hopping schemes, some of which we make use of and build upon in Argos. All of these systems rely on a dense deployment of sniffers and perform a "microscopic" observation of a single wireless network environment.

Passive monitoring has also been explored in low-power wireless sensor networks (WSN), including SNIF [40] and LiveNet [21]. These systems are useful for debugging and diagnosing network performance anomalies but focus on a single sensor network rather than many existing wireless LANs. Our goal in this paper is to study the efficacy of wireless monitoring on an urban scale, capturing traffic from thousands of existing wireless LANs.

A number of studies of the performance and traffic characteristics of urban-scale mesh networks have been published. The RoofNet project has undertaken detailed studies of link-layer and routing performance across a WiFi mesh of 38 nodes across a city [16, 14]. These studies focused on the low-level network performance rather than ambient traffic. A study of the Google WiFi mesh [12] in Mountain View, CA explored the behavior of mesh network users, but did so through capturing data at the wired gateways, and did not look explicitly at the wireless network dynamics.

3 Argos Architecture

Argos utilizes a two-tier architecture consisting of multiple *sniffer nodes* and a single *central server*. Sniffer nodes are interconnected via a backhaul network, such as a wireless mesh, allowing sniffers to communicate with each other as well as the central server. Sniffers capture wireless traffic and perform local filtering and processing on the raw packet streams. In addition, sniffers can perform *in-network traffic merging* to merge their individual streams, increasing capture fidelity. Results from queries are delivered to the central server where they are conveyed to the user. Figure 1 shows the architecture of a single sniffer node; the remainder of this section details each component.

3.1 User Queries

In Argos, wireless packet streams are processed by *user queries*. Multiple concurrent queries can execute on the Argos network at one time. Relatively few constraints are placed on the structure of queries, allowing for a wide variety of applications. Simple examples include generation of summary statistics over a sliding time window or filtering packets according to header fields.

Each query consists of two dataflow graphs composed of packet-processing operators: a *sniffer* dataflow, which is replicated on each of the sniffer nodes, and a *server* dataflow which is executed only on the central server. Each sniffer dataflow instance receives input packets from the Argos node

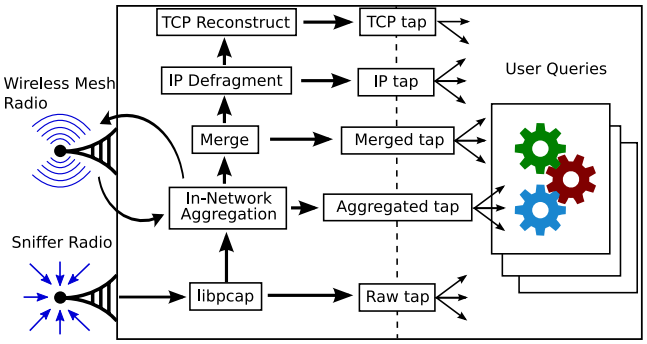

Figure 1. This figure shows a block diagram of how packets flow through a single Argos sniffer. Starting with packet capture from the sniffer radio in the lower left, packets are passed up the chain of processing elements. At each step, a *tap* provides access to the packet stream at that point for any interested user queries.

that it is running on via one or more *taps*, and sends results (via the backhaul network) to the server dataflow. The server dataflow may perform additional processing or aggregation on the results received from the sniffers, as well as log the output of the query and report the results to the user.

Each Argos sniffer provides five taps by which sniffer dataflows can receive packets, based on what degree of preliminary processing is desired. The *raw* tap provides packets as soon as they are captured by the local interface. The *aggregated* and *merged* taps provide packets after any duplicate captures are aggregated and then merged from multiple sniffer nodes, as described below. The *IP* and *TCP* taps provide defragmented IP datagrams and reconstructed TCP sessions, respectively. Many queries make use of only a single tap, but in some cases combinations of multiple taps are useful.

User queries are implemented using the Click software router [31], which provides a rich set of interfaces for network packet processing. A query's sniffer and server dataflows are written directly in the Click language. Each query also specifies a small number of configuration and performance-related parameters, including the list of network taps that the sniffer dataflow uses. For each specified tap, the dataflow must have one input over which Argos will pass the appropriate packets. Any packets output by the dataflow are automatically transferred over the backhaul network to the central server and passed as input to the query's server dataflow. This allows easy aggregation of results from the entire sniffer network without having to deal with explicit network connections between sniffers and the central server.

3.2 Sniffer Nodes

Argos sniffer nodes consist of single-board PCs coupled with one or more 802.11 radios for packet capture. Our prototype nodes, described further in Section 4, use a Wistron CM9 802.11a/b/g radio with an 8 dBi omnidirectional antenna for packet capture. Each node runs an instance of the Argos sniffer process, which accepts raw captured packets from the operating system, performs in-network processing (as described below), and emulates a TCP/IP network stack (for IP defragmentation and TCP stream reconstruction). Packet capture is performed using the standard *libpcap* [9] interface provided by the OS. The wireless device driver prepends each captured packet with a *radiotap* pseudo-header [8] which includes fields such as the received signal strength and type of physical layer modulation used.

Anonymization: Argos sniffers support a range of data anonymization mechanisms to prevent private information from leaking into user queries. At present, use of these mechanisms is optional, since we have not yet opened our prototype up to external users. Argos implements widely-accepted techniques for masking source and destination MAC addresses and IP addresses through hashing [38]. In addition, Argos can optionally drop arbitrary portions of a captured packet including individual header fields and the entire payload. Of course, the impact of these techniques depends very much on the semantics of the query, and there is no one-size-fits-all approach that will work for all queries. We are currently investigating the use of a stronger differential privacy guarantee that will prevent Argos queries from leaking information about specific users (§8).

Wireless mesh backhaul: Sniffers are interconnected via a backhaul network that provides them with connectivity to each other and the central server. Depending on the deployment scenario, a wide range of backhaul network options are possible. Direct Ethernet connectivity to sniffers may be possible in some situations, although the cost can become prohibitive, especially for sniffers deployed on rooftops and streetlights in a city, which is our focus. Cellular or WiMax connections may also be possible, although they incur additional subscription costs and would not enable direct connectivity *between* sniffers. Also, most current commercial offerings have fairly limited uplink capacity (< 1 Mbps) [30, 47].

In Argos, we focus on the use of wireless mesh as a cost-effective and scalable backhaul network solution. Wireless mesh has been widely studied and deployed in a number of research and commercial settings [2, 4, 13, 16]. Given the reasonably close spatial proximity of most Argos sniffers, mesh is a good choice for interconnectivity, and can be deployed at low cost with no recurring fees. Mesh also allows sniffer nodes to communicate directly with each other to enable cross-sniffer collaboration, such as in-network merging of packet streams captured from multiple sniffers.

Of course, it is important that the wireless mesh backhaul does not interfere with the ambient traffic being monitored by Argos. In our prototype, the wireless mesh uses a secondary radio operating in a different frequency band (900 MHz) than the sniffer radio, to ensure that there is no crosstalk. The mesh itself is also an 802.11 network but operates in the non-standard frequency band, and provides up to a few Mbps of throughput on each link.

3.3 In-network Traffic Processing

Given that an individual sniffer will only receive a partial view of the traffic from a given source, merging the packet streams from multiple sniffers can enable higher packet coverage. Merging also removes duplicates, which is important for the accuracy of some queries. Yeo et al. [48] were the first to use this technique and it has since been used by other wireless network monitors [23, 34]. In each of these systems, every sniffer sends its raw packet stream to a central server for merging, which is only viable for indoor environments

with ample backhaul network capacity. However, such a centralized approach is unsuitable for large-scale outdoor sniffer networks with constrained backhaul network capacity. Even with compression, the volume of captured traffic can overload the backhaul network, leading to substantial packet loss. Likewise, the centralized approach does not scale as the number of sniffers and network diameter increase.

To address this problem, we develop an approach to *in-network traffic processing* that attempts to balance the traffic load on each backhaul network link to minimize saturation and packet loss. The idea is to logically partition the global packet stream so that each sniffer node is responsible for some disjoint fraction of the stream. Sniffers forward each captured packet to its designated *aggregator* node, which (a) merges the partial streams in receives, and (b) processes the merged packets by executing the user queries.

Formally, we define the partitioning function $p : S \rightarrow N$ where S is the set of traffic sources (i.e., a wireless client or access point being monitored) and N is the set of aggregator nodes. Each sniffer capturing packets from a source $s \in S$ applies the partitioning function $p(s)$ to determine the aggregator $a \in N$ that should receive packets sent by s. Each aggregator merges the packets that it receives, before passing them to a locally-running instance of each user query. This approach is inspired by the *MapReduce* paradigm used for large-scale distributed processing in datacenters [24].

Traffic partitioning: Argos must take into account the cost of transferring each packet from the capturing node to the aggregator node. In a wireless mesh environment, it is desirable to avoid sending captured packets over multiple network hops, as this increases overall network load. Also, the choice of partitioning method will determine the set of packets that are observed by each aggregator node after merging. This affects user queries, since each sniffer dataflow will only be able to observe its locally-merged portion of the global packet stream. Ideally, the partitioning method should not restrict the query logic.

Our approach to traffic partitioning is based on assigning all traffic captured for a given *basic service set* (BSS) to a single aggregator node. The BSS represents a single access point and all of its associated clients. The set of sniffer nodes that capture traffic from a given BSS tend to be within close physical proximity to each other, so this choice of partitioning function should minimize backhaul traffic. Also, many queries naturally wish to observe properties of individual clients or access points, so partitioning according to BSS allows queries running on each aggregator node to observe the full merged packet stream from each source.

To implement this approach, each sniffer maintains a table mapping each BSSID its aggregator node. For each packet captured, the sniffer looks up the appropriate aggregator and forwards the packet. In cases where an aggregator has not yet been assigned, this table is constructed as follows. Each sniffer maintains a count of the number of packets captured from each BSSID. These counts are periodically reported to the central server. The central server assigns aggregation duties to the node with the largest number of received packets for each BSSID. This approach ensures that the fewest number of packets will need to be sent from *other* sniffer nodes to the aggregator, since the aggregator has already captured the largest number of packets for this BSS.

Although most 802.11 frames specify a BSSID, this is not the case for control frames (e.g. ACK and RTS frames), which only specify their destination. To properly associate these packets with their BSSID, sniffers maintain a cache of MAC address to BSSID mappings, which is updated whenever a non-control frame is captured. The BSSID can then be determined for a control frame by consulting the cache. In addition, we need a special case for frames sent to the broadcast address. Instead of attempting to aggregate and merge these frames, we simply pass all broadcast frames through for local processing by the capturing sniffer.

Stream merging: On each aggregator node, stream merging is performed in a manner similar to Jigsaw [23]. Received packets' timestamps are adjusted to account for clock skew differences between sniffers. As suggested by Yeo et al. [48], we use beacon frames to update the current timeskew estimates between sniffers; unlike other wireless frames, beacons provide unambiguous synchronization points by virtue of their 64-bit timestamp field, which ensures that every beacon is unique. Next, the packet stream is merged by buffering packets for a window of time and searching for duplicate packets (from different sniffers) with similar timestamps; these are assumed to be copies of the same original packet transmission and are merged into a single packet. When a packet is output by the merger, it is annotated with a list of all of the sniffers that captured the original transmission, along with each sniffer's local capture time and received signal strength. This information is important for a variety of spatially-aware queries as we will discuss below.

In conventional approaches to in-network aggregation, nodes route sensor values up a collection tree to the root, merging values at each hop along the way. In comparison, Argos merges packets only once (at an aggregator node). This leads to two efficiency improvements: first, routing a packet to its aggregator node is generally cheaper than routing it all the way to the root; second, execution of user queries can occur in-network on aggregator nodes instead of only at the root. Conventional approaches generally cannot push queries into the network because merging of values can happen at every node and thus the final value is not obtained until the root is reached. In Argos, however, a packet is "final" after its one and only merge (on its aggregator node) and thus we can immediately apply user queries.

Channel identification: After performing traffic merging, it is necessary to determine the original 802.11 transmission channel of each packet. When packets are first captured, Argos annotates them with the current channel of the capturing interface. However, this is not necessarily the correct channel, since 802.11 channels overlap and packets transmitted on one channel can be received on nearby channels. For example, in one 10-minute packet trace from an outdoor sniffer tuned to channel 2, over 75% of the received packets were actually transmitted on channel 1 (a much busier channel, leading to frequent packet bleed-over).

To handle this situation, each sniffer node maintains a cache of which channel each AP is operating on, which is announced in the AP's beacon frames. This information is

used to determine the transmission channel for most captured packets. In cases where packets are captured from a BSSID for which no beacons have yet been received, we fall back to using the capturing interface's channel as a guess.

3.4 Protocol Stack Emulation

Merged traces are subject to protocol stack emulation, allowing for higher-level analysis of the network traffic by user queries: IP fragments are reassembled and TCP flows are reconstructed. Due to sniffers' limited capture abilities, many TCP flows are expected to be only partially reconstructed, leaving "holes" in the stream where one or more TCP packets were missed. Rather than rejecting these flows, we annotate each with a listing of which sequence-number ranges were and were not captured. We use timeouts to determine when to "close" and output each TCP flow, as there is no guarantee of capturing the FIN packet to mark its completion.

We expect that many classes of queries will be able to make good use of partially-reconstructed TCP flows. For example, any query examining application-level headers (e.g. HTTP requests or peer-to-peer traffic) may need to capture only the first 1-2 packets in a TCP stream. Rule-based intrusion detection systems, such as Snort [41], can detect many attacks with only a small number of captured bytes if they happen to line up with the appropriate attack signature.

3.5 Sniffer Channel Management

A unique challenge in Argos is that of simultaneously monitoring multiple radio channels. There are 14 total 802.11b/g channels (of which 11 are permitted for use in the US). One approach is to equip each sniffer node with multiple radios. Previous indoor WiFi monitoring systems, such as Jigsaw [23], used 3 radios on each sniffer node tuned to the most common 802.11 channels (1, 6, and 11); however, in a large, urban-scale setting we expect to see a substantial amount of traffic on nonstandard channels. For example, we estimate (by observing captured beacons) that 20% of the APs located in the vicinity of our current sniffer deployment utilize channels other than 1, 6 and 11. Clearly, if we want to include all nearby wireless networks in our monitoring, it would be prohibitive to equip sniffers with as many radios as there are available channels.

Our approach is to perform intelligent channel hopping on the sniffers to maximize packet capture coverage for user queries. The simplest approach would entail a simple static schedule with fixed dwell times, although this is unlikely to achieve the best results given the variations in channel occupancy. A better approach is to dynamically weight dwell times based on channel occupancy [25]. Since query requirements may vary, Argos does not stipulate any single channel hopping policy, but instead provides mechanisms for queries to specify their own policy.

Given that traffic of interest to a user query may be picked up by multiple sniffers, it is also desirable to *coordinate* channel hopping across nearby sniffers in order to maximize capture rates. Argos provides a *channel focusing* mechanism, whereby a sniffer can request that other nearby sniffers switch to a given channel so as to improve the overall chances of picking up interesting traffic. For example, a query running on a given sniffer may detect traffic of interest (say, TCP packets destined for a certain IP address) and instruct other nearby sniffers to switch to the same channel to improve capture fidelity for the remainder of the traffic stream. Currently we define sniffer "nearness" as geographic distance, although other options exist, such as the degree of overlap in captured traffic.

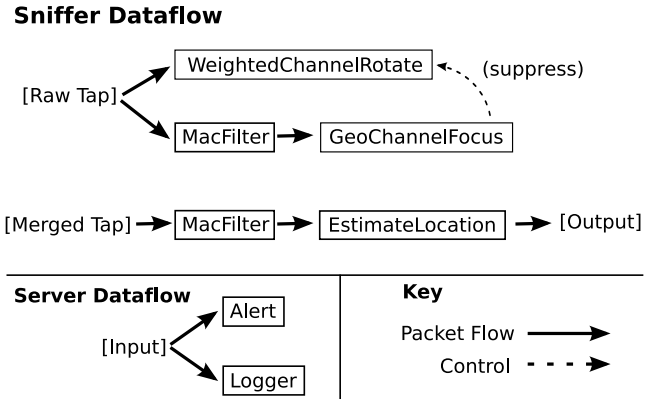

Figure 2. The sniffer and server dataflows of a "stolen laptop finder" example query; packets are passed to or from the query at bracketed elements (e.g. `[Raw Tap]`).

With multiple concurrent user queries, each query may wish to listen on a different channel at the same time. In Argos, we address this through the use of *channel leases* and *prioritization*. A query can request that the sniffer radio be changed to a given channel c for duration d with priority p. The channel manager will grant the lease to the query with the highest priority. If the priority of the request is greater than the priority of the current lease, the current lease will be preempted. When the current lease expires, pending lease requests are processed in decreasing priority order. Often, each query is assigned a static priority at configuration time (e.g. according to user ID), although many policies are possible.

As an example, the default weighted channel policy will iterate through the available channels according to an estimate of the amount of traffic being captured on each channel. A query can temporarily override this default policy by requesting a coordinated channel focusing across a set of sniffers, by making a lease request as described above. Once the lease expires, the sniffer will return to the default policy.

3.6 Example Query: Stolen Laptop Finder

An example Argos query is illustrated in Figure 2, which implements a simple stolen laptop finder. The query maintains a list of target MAC addresses representing stolen laptops that should be tracked (e.g., after being reported stolen by the owner). On each sniffer, locally captured packets are passed (via the *raw* tap) to a *WeightedChannelRotate* element that periodically adjusts the channel-hopping schedule according to the number of packets observed on each channel. Packets are also passed to a *MACFilter* element that drops all packets whose source MAC address is not in the list of targets. Matching packets are passed on to trigger the *GeoChannelFocus* element, which recruits nearby sniffers to switch to the same channel. This will increase the network's overall ability to recover packets from the stolen laptop (which may aid in identifying its location or current

Figure 3. Map of the Argos sniffer deployment in Cambridge, MA. The area shown is just under 7x5 km.

Figure 4. Argos sensor node deployed on a street lamp. The thick antenna pointed down on the right is for the 900MHz backhaul radio, and the thinner antenna on the left for the 802.11 sniffer. The node is powered from the streetlight.

user). The *WeightedChannelRotate* element uses a low priority when obtaining leases for channel-hopping. This ensures that the *GeoChannelFocus* element, using a higher priority, can to suppress channel-hopping when a laptop is detected.

After in-network merging, the merged packet stream is passed to each query via the *merged* tap, where a second filtering for target MAC addresses is performed. Any matching packets are passed to an *EstimateLocation* element that uses each merged packet's annotation of which sniffers captured that packet and the associated received signal strength to estimate the transmitting laptop's location. Finally, lists of detected laptops and estimated locations are pushed to the central server, which alerts the end user.

4 Implementation and Deployment

We have implemented and deployed a complete prototype of the Argos system. The deployment consists of 26 nodes mounted on streetlights and rooftops around Cambridge and Somerville, MA, in four (disconnected) mesh clusters together spanning over 9.7 km^2 (see Figure 3). Each sniffer node consists of either a Soekris net4826 (233 MHz CPU, 128 MB of RAM) or ALIX 2c2 (500 MHz CPU, 256 MB of RAM) single-board computer, housed in a NEMA 6x-rated weatherproof enclosure. Both configurations use a Wistron CM9 802.11a/b/g mini-PCI radio with an 8 dBi omnidirectional antenna as the sniffing device, and a Ubiquiti XR9 900MHz mini-PCI radio with a 6 dBi omnidirectional antenna for backhaul connectivity via wireless mesh. This is a high-power radio designed for long-range operation, with a peak transmission power of 28 dBm. The XR9 uses the 802.11 MAC and PHY, albeit in a nonstandard frequency band, with PHY rates up to 54 Mbps, although in practice we observe typical *link-level* TCP throughputs from 2-16 Mbps. OLSR-NG [6] is used as the backhaul mesh routing protocol. Five of the nodes also have Ethernet connectivity and function as wired sinks. Figure 4 shows one of our sniffer nodes deployed on a streetlight. Note that Argos sensor nodes are continuously powered, which is easy to do on most rooftop and streetlight installations. We are currently designing a solar-powered Argos node for deployment in locations where AC power is not available.

It is important to note that the nodes in our deployment were not located to maximize packet capture. Instead, the siting of sniffers was dominated by physical and logistical factors; where we are allowed access, there are suitable mounting areas, and electric power is available). Hence, the geographic distribution of the sniffers is nonuniform and not intended to be ideal for ambient wireless monitoring.

As noted previously, Argos is implemented using Click [31]. In addition to reusing a number of existing Click elements, we implemented a total of 46 new elements, which perform operations such as packet partitioning (by BSSID), packet-stream merging, and channel hopping control. We also make use of the QuickLZ [7] compression library for compressing network traffic.

5 Performance Evaluation

In this section, we evaluate the performance of Argos' approaches to in-network traffic processing and coordinated channel focusing. For traffic processing, the primary metric we are concerned with is the amount of backhaul bandwidth required to send packet traces between sniffer nodes. We demonstrate that Argos' dynamic aggregator assignment substantially reduces the bandwidth requirements compared to sending all traffic to the central server for merging.

For coordinated channel focusing, the goal is to improve the sniffer network's ability to detect packet streams of interest by focusing multiple sniffer nodes on the same channel once a sniffer detects the target. We show that Argos' triggered channel focusing improves capture of event traffic compared to simpler channel hopping approaches.

5.1 In-network traffic processing

First, we consider the impact of Argos' approach to in-network traffic processing on backhaul bandwidth usage. The benefits depend on several factors, including the volume of traffic being captured, the backhaul network topology, and the backhaul capacity. Given that our deployment of Argos represents a single point in a large parameter space, to study these effects we make use of a simplified analytical model

Class	Frequency	Offered Traffic Load
APs	18%	5514 Bytes/sec
Active Clients	14.5%	287 Bytes/sec
Idle Clients	65%	49 Bytes/sec
Ad-hoc Stations	2.5%	931 Bytes/sec

Table 1. Model for distribution and traffic load of transmitters, based on live deployment data.

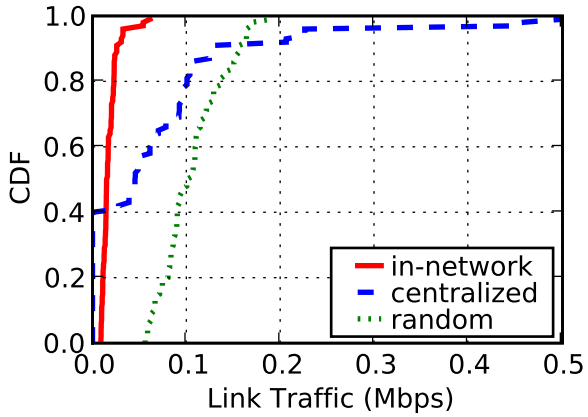

Figure 5. Distribution of traffic load on backhaul network links with centralized and in-network traffic processing, using 25 sniffers and 4000 transmitters, resulting in a total offered load of 34 Mbps.

that allows us to vary each parameter separately.

The model is based on a square grid of 25 sniffers, with each sniffer having direct mesh backhaul links to the four adjacent sniffers. The central server is placed in the center of the grid. There are 4000 traffic sources uniformly randomly distributed throughout the space; each source is randomly chosen from the four classes in Table 1, based on observations from the real Argos deployment described in Section 6. Each source i generates CBR traffic according to the rate shown in the table, denoted r_i. A sniffer s captures a fraction of the traffic from each source i according to the function $F_s(i) = \texttt{dist}(i,s)^{-\alpha}$, where `dist` is the distance from the source to the sniffer. We chose a pathloss exponent of $\alpha = 3$ which is a good estimate for urban settings [39]. Hence, the total volume of traffic captured by s is $\sum_i F_s(i) \cdot r_i$.

Though intended to be general, this setup is based on parameters from our live deployment (§6), including the number of sniffers (25), the average number of traffic sources typically detected during the day (4000), and the offered traffic that we infer per transmitter (table 1). The constant factors of the packet capture function were scaled such that the average capture rate was 8%, to match capture performance inferred from our deployment. Some aspects of this model are undoubtedly unique to our deployment, although prior campus-based studies have reported comparable values for client and AP traffic rates [32], as well as for the ratio of the number of active clients to active APs [32, 45].

Given the capture rates for each sniffer, we can calculate the backhaul network load that arises with (a) a centralized policy in which each sniffer transfers all of its captured traffic directly to the sink, (b) a randomized assignment of traffic streams to aggregator nodes; or (c) Argos' in-network processing. In each case, we calculate the load imposed on each link of the backhaul network. Traffic is compressed by each sniffer prior to transmission, using an experimentally-determined compression ratio of 65%. For the centralized policy, we calculate a shortest path from each sniffer to the sink. The total load on each backhaul link is just the sum of the load induced by each sniffer whose traffic traverses that link. For the two in-network processing cases ((b) and (c)), aggregator nodes reduce the traffic stream by a percentage and transmit the remainder to the central sink node.

The amount by which in-network processing reduces a traffic stream depends on two independent factors. First is packet merging, which reduces traffic at a rate proportional to the frequency of duplicates in the captured stream. This depends on the sniffer network's density and the channel selection pattern (since nearby sniffers can only capture duplicate packets if they are tuned to the same channel). In our network we see an average merge rate of only 7%.

Secondly, in-network processing also reduces traffic streams by executing the user queries. The queries' summed output data rate depends on the number and types of the queries, but in our case the 10 queries that performed all of the logging and data collection for sections 6 and 7 resulted in a 90% data reduction (this includes the 7% reduction just from packet merging). In other words, the combined output of this set of queries was one-tenth the data capture rate. From this, we assume in our model that in-network processing results in a 90% data reduction on each aggregator node.

Figure 5 shows the distribution of load on backhaul links for each policy. In this case, the total offered load from the traffic sources is 34 Mbps. As the figure shows, the centralized approach induces a wide variation of traffic loads on each backhaul link, since the links closest to the sink must carry much more traffic than nodes near the leaves. In-network processing spreads the load more evenly across the mesh, with a max link usage of 0.06 Mbps, 8 times less than with the centralized policy. The randomized scheme only partially realizes the benefits of in-network processing, resulting in a max link usage of 0.19 Mbps, and incurs the highest mean and median link usages (0.1 Mbps each). Figure 6 shows how the max link usage varies as the total offered load from traffic sources increases.

5.2 Coordinated channel focusing

Next we evaluate our technique of channel focusing in order to allow multiple sniffers to capture traffic from a given source. The objective is to capture as many packets as possible from an "event of interest," such as an anomalous traffic pattern or transmissions from a specific wireless client. Since an interesting event can occur on any 802.11 channel at any time, an uncoordinated approach to channel hopping is unlikely to capture many packets from an event. In the channel focusing strategy, if any one of the sniffers detects a packet representing an interesting event, it recruits the 3 (geographically) nearest sniffers to switch to the same channel and begin capturing packets. Our metric is the number of packets captured for each event.

In order to have a fair comparison, an ideal experiment

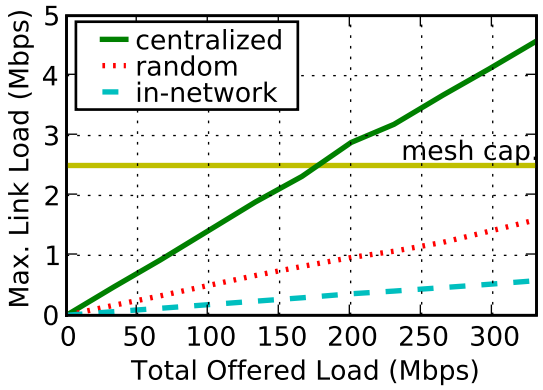

Figure 6. Scalability of in-network processing with increasing load. The number of sniffers was fixed at 25 while the number of transmitters was scaled up. For context, 2.5 Mbps is a (generous) estimate for current wireless mesh capacity limits; we expect that sniffer networks with any links loaded beyond this amount will suffer from network congestion.

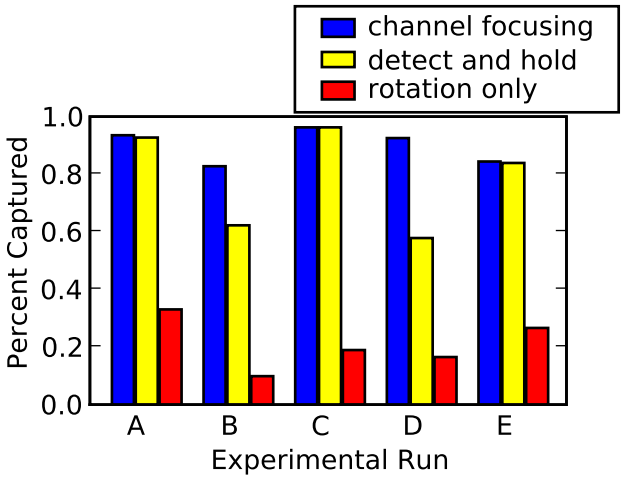

Figure 7. Event detection capture rates with and without channel focusing.

should use the same source traffic for each channel hopping policy. However, given that each sniffer can only listen to a single channel at a time, we have to emulate this setting with a packet trace captured offline. We captured a set of 5-minute packet traces from 9 nearby sniffers in our Argos deployment. There are 11 traces in total, one for each 802.11 channel. For the experiments, sniffer nodes read from the traces instead of capturing packets from the radio. We emulate a situation where a sniffer can "change channels" by reading data from the appropriate trace; that is, the traces virtually overlap in time, although they were initially captured at different times.

We randomly chose 5 "interesting events" within the captured traces. Each event is defined as a packet chosen at random followed by all packets transmitted from the same station over the next 10 seconds. Each event is comprised of at least 100 packets. The performance metric is the fraction of "interesting event" packets that are successfully captured, in aggregate, by the entire sniffer network.

We evaluate three different channel hopping schemes. *Rotation Only* uses weighted channel hopping in which each sniffer independently rotates through channels with dwell times proportional to the amount of traffic observed on each channel. *Detect and Hold* causes a sniffer to remain on the same radio channel for 10 sec after detecting a single packet from an interesting event, thereby increasing its chance of picking up more packets from the event. Finally, *Channel Focusing* has the first node that detects an event recruit nearby sniffers to switch to the same channel for 10 sec.

Figure 7 shows the fraction of packets captured for each of the 5 events (labeled A–E) for each of the three policies. In each case, channel focusing greatly improves the capture fidelity compared to the *Rotation Only* policy. Indeed, a single sniffer holding on the same channel (*Detect and Hold*) results in a substantial improvement to event capture, with improvements ranging from 41% to 77%. Channel focusing with neighbor recruitment further improves capture rates up to 34% above the *Detect and Hold* policy. This shows that there is clear benefit from coordination across sniffer nodes.

6 Urban Wireless Traffic Characterization

In this section, we leverage the Argos deployment to perform a detailed characterization of wireless network usage across the city. Unlike previous wardriving studies [15, 27], we have the benefit of continuously monitoring wireless traffic from multiple vantage points, rather than taking a single snapshot view of wireless traffic from a single mobile sensor. This yields a much richer picture of the urban wireless landscape than previous studies have revealed.

6.1 Overall network population

We conducted a number of different measurement studies on Argos that spanned a total of 6 months. This section presents data from a 12 day period over which detailed traffic measurements were recorded. In total, we detected 2630 access points and 65,073 wireless clients, although our rate of capture varied widely across this population. By way of comparison, these counts are each over 25 times larger than those reported for a 24-hour trace taken by the largest indoor wireless sniffer deployment [23]. A usage study of Google's 500+ node outdoor mesh network [12] recorded 30k clients over a 28 day trace, but this includes only the mesh traffic, as opposed to all nearby wireless networks.

Overall, we captured a total of 1.1 TB of traffic across 2.4 billion packets. It turns out that a single wireless client in the vicinity was responsible for 81% of all the captured bytes and 30% of all captured packets; this "spammer" node appears to be continuously transmitting max-sized 802.11 frames. Unfortunately these frames are encrypted so there is not much we can say about them – this traffic is excluded from the remaining analyses in this paper. The amount of non-spammer traffic captured per sniffer varied greatly, from just 3 MB to nearly 23 GB, with an average and median of 8.5 GB and 9.6 GB, respectively. This variance is not surprising, due to differences in the sniffers' locations as well as the densities and activity levels of nearby wireless net-

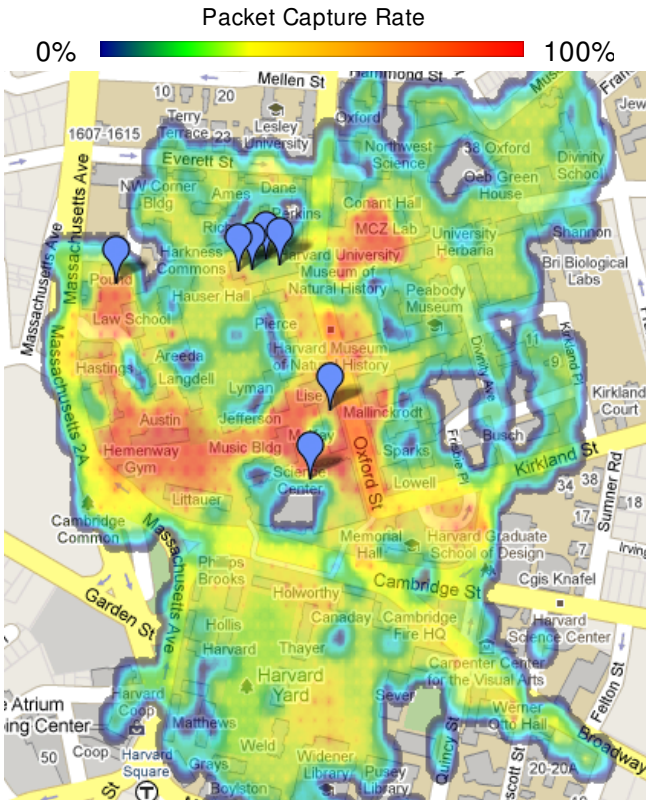

Figure 8. Spatial coverage of the Argos sniffer network, in terms of the fraction of packets captured from a mobile laptop in each location. Sniffers are shown as blue markers.

works. Overall, 61% of the packets and 44% of the networks utilized some form of encryption.

6.2 Spatial coverage

The first major question is, how much spatial coverage is afforded by the Argos sniffer network? That is, over what physical area can the sniffers observe ambient wireless traffic? We conducted an experiment in which a user with a laptop roamed around the Harvard University portion of our deployment area. This area includes a number of multi-story buildings as well as several open areas, typical of a university campus. The laptop was equipped with a GPS receiver and continuously broadcasted UDP datagrams containing the current GPS coordinates. The sniffers observed this traffic and, for each position of the laptop, we determined the fraction of packets captured by Argos.

Figure 8 shows the results; the (extremely noisy) packet reception rates from all 7 sniffers are combined and the data is smoothed to yield a clearer picture of the spatial coverage. The figure shows that although our capture ability is related to distance (as expected), it's a very rough correlation. On the one hand we were able to capture packets from a laptop up to 430 m from the nearest sniffer, which is surprising given the presence of several tall buildings obstructing the line-of-sight path between the laptop and sniffers. On the other hand, there are also areas quite close to multiple sniffers that had low packet capture rates.

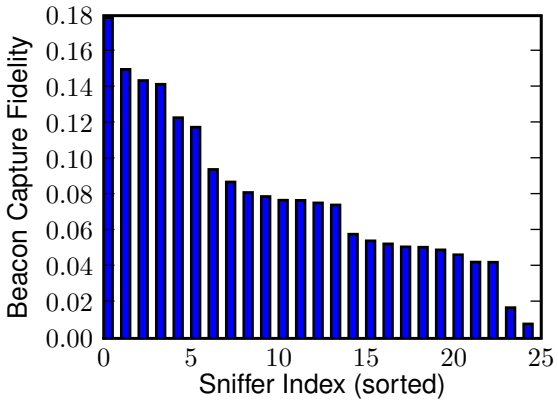

Figure 9. Capture fidelity as fraction of beacons captured by each sniffer.

6.3 Traffic capture coverage

The second question is, how much of the total ambient traffic was Argos able to detect? This is a difficult question to answer, as we do not have ground truth as to how much ambient traffic there actually is. As an estimate, we compute packet reception coverage for each sniffer by counting 802.11 beacons received from access points. Access points broadcast beacons at a fixed interval (typically 10 Hz), so it is straightforward to calculate the percent of beacons captured from only the total number of beacons captured and the elapsed time. As noted by others [29], beacon capture rates can be used as a rough proxy for overall packet capture rates, although their capture rates tend to be somewhat higher than other traffic as they are transmitted at low PHY rates.

As shown in Figure 9, sniffer coverages range from 18% to just under 1%; the overall coverage of the entire network was 8%. These values are much lower than what is typically seen with indoor monitoring networks, where coverage can exceed 95% [23]. We discuss the implications of this (and possible ways to improvement traffic capture) later in this section. Also note that this figure omits APs for which we captured only a small number of beacons (< 10) as fidelity estimates in these cases are likely to be inaccurate; 21% of detected APs fell into this category.

The next question we can ask relates to the variation in traffic over time. Figure 10 shows the types of TCP traffic (classified by port number) captured over a representative weekday. Unsurprisingly, HTTP and HTTPS are the dominant traffic classes, with Email (POP3, IMAP), FTP and NetBIOS (not shown) making up most of the identifiable remainder. Although the expected diurnal patterns are clearly present, the captured traffic was quite bursty with hourly per-class traffic spikes up to 147 MB. Each of the two prominent HTTPS traffic spikes represents traffic from a single AP. However, in both cases multiple clients were associated with the AP so we could not unambiguously determine which specific client was the source.

Similarly, the two HTTP traffic spikes at 1:00am on days 1 and 3 were each traced to a single AP. Since the traffic events occurred in the middle of the night, only 2-3 clients

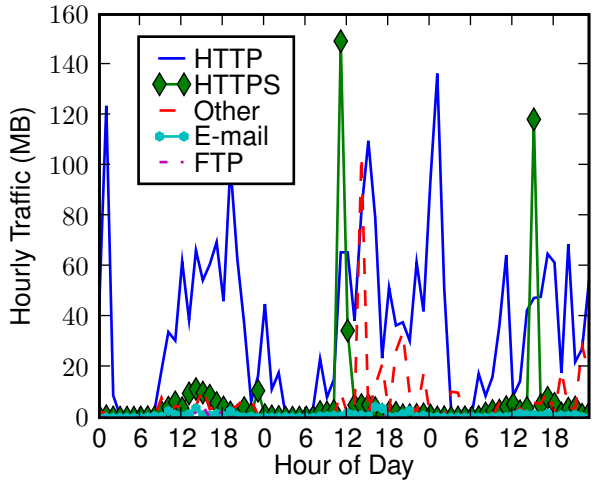

Figure 10. TCP traffic, by class, over 3 consecutive weekdays.

were observed associated with the AP. One client in particular is the likely source for both of the traffic spikes; this client was associated throughout both events, during which we observed the client make hundreds of web requests. Many of these requests were to video-hosting sites (e.g. `youtube.com`, `netflix.com`) further explaining the sudden traffic usage.

6.4 Discussion

From these measurements, it is clear that Argos' ability to monitor large populations over significant geographic areas comes at the cost of missing a significant percentage of the ambient traffic. Although this somewhat constrains the applications suitable for Argos, our current performance turns out be adequate to perform many interesting analyses (§7). Nonetheless, its useful to briefly consider two sources of packet loss that we have identified, with an eye towards future system improvements.

Firstly, one must consider the urban environment, with its frequent line-of-sight path obstructions by nearby buildings, plus the extremely dense penetration of extant WiFi networks. Although these difficulties are largely innate to urban settings, we expect that one could reduce their impact as our sniffers were not optimally located for packet capture and simply use 8 dBi omnidirectional antennas, rather than dish or patch antennas that would have aided packet capture.

We also note that Argos' long-range packet capture may be biased towards packets sent at low PHY rates, as these packets can be received at lower signal strengths than those sent at higher PHY rates. Since we have no way to know the true distribution of PHY rates used in our measurement area, we instead utilize packet traces captured in other settings as an imperfect comparison. We examined the distribution of PHY rates present in packet traces from a conference in 2006 [20] and an academic building in 2007 [22]. In both cases, the sniffers were densely deployed and thus captured nearly 100% of the traffic, yielding a complete picture of the PHY rates in use. The indoor traces show 20% and 6% (respectively) of packets were sent at 36 Mbps or above, whereas this is true for only 0.5% of the packets captured by

Rank	*Requests*	*Clients*	*Site*
1	1856	301	www.facebook.com
2	736	239	www.google.com
3	267	5	www.huffingtonpost.com
4	249	5	www.craigslist.org
5	239	1	www.repubblica.it
6	232	143	www.apple.com
7	162	22	images.apple.com
8	139	17	my.lesley.edu
9	133	46	www.youtube.com
10	131	7	www.seas.harvard.edu

Table 2. Top ten websites visited by users, ranked by number of captured requests.

Argos. We tentatively conclude from this that Argos is indeed biased against high data rate packets. At PHY rates of 24 Mbps and below, there was no clear trend.

Secondly, many packet transmissions are missed simply because all nearby sniffers are tuned to other channels at the time. The above measure of fidelity estimates Argos' capture rate across *all* traffic, but this is not optimal for many queries. If a query is interested in a subset of the traffic, the rate of capture of *just that traffic* can be increased if the query can predict on which channels the traffic is most likely to occur. For example, our queries use the default channel policy, which weights channels by the total traffic volume on each, whereas a query interested only in TCP traffic might instead weight channels by just the 802.11 *data* traffic. This would deprioritize channels with lots of 802.11 management traffic (e.g. beacons) but little higher layer traffic.

We tested a simplistic scenario with a query interested solely in traffic from channel 1, which lead to a network-wide fidelity of 25%. This is a significant improvement over the 8% reported earlier, although this is somewhat of a best-case scenario because the channel prediction was trivial (all nodes statically assigned to channel 1) – we predict that most queries can realistically expect a fidelity somewhere between 8% and 25%, depending on the pattern of traffic they are interested in and how accurately one can predict when and on which channels the traffic will occur. Developing channel policies to that can forecast different classes of traffic is a challenging area of future work.

7 Case Studies

In this section, we showcase Argos' capabilities for enabling complex user queries against the rich data stream of ambient wireless traffic. We present four case studies to highlight potential use cases for Argos: tracking of popular Web sites and Google searches from different parts of the city; detection of malicious traffic and malware over the airwaves; tracking of public transport services; and fingerprinting individual clients based on their wireless network behavior. These case studies are intended to demonstrate Argos' capabilities to distill complex wireless network traffic to yield a high-level view.

7.1 Popular Websites and search patterns

The first case study deals with exploring the popularity of Web sites visited by wireless users around the city. Determining which website a user is visiting is non-trivial: simply looking at packets destined for port 80 reveals IP addresses, but many of these are for auxiliary servers (such as ad sites

Count	*Snort Rule*
1229	SQL ping attempt
794	x86 shell code exploit
204	SQL probe response overflow attempt
121	Web server buffer overrun attempt
87	FTP traffic encrypted

Table 3. Top five Snort alerts.

and CDN servers) that are not being directly visited by the user. We implemented a query that captures the HTTP request headers from users and looks for a `Host:` field in the request, which most browsers add to indicate which host is being visited. We also look for HTTP responses where a `Set-Cookie` header is used. Using this technique, we captured a total of 523,818 HTTP requests, for which we could determine the site name for 62,292 of them. 6143 unique websites were visited, and Table 2 shows the top 10 after we exclude ad sites and CDN servers.

We captured Web search queries by looking for HTTP requests to URLs containing appropriate strings (e.g., Google searches include `&q=` followed by the query phrase). We captured 1,725 web searches, 686 of which were to Google. The most popular search terms are not particularly surprising, dominated by words such as *in*, *of*, and *boston*.

7.2 Malicious traffic

Most studies of malware on the Internet [36, 33] are based on traces captured from wired networks or border routers. One notable exception is Stone-Gross et al.'s study of user traffic at a conference [44]; the authors demonstrate that identifying malware through wireless sniffing is possible, although they also utilized packet traces taken from the wired network. Additionally, they estimated a capture fidelity of more than 90% owing to their compact, indoor setting.

We are interested in whether Argos can detect malicious behavior over the airwaves. We set up an Argos query to feed merged traces to the Snort [41] network intrusion detection system, configured with a standard set of 5303 detection rules. We added two additional rules [3] to detect traffic from the Conficker botnet. Over 11 days, Snort raised 2979 alerts across 37 rules; 141 unique client devices were implicated by the alerts. Table 3 shows the top 5 triggered alerts. The Conficker shellcode botnet rules triggered 5 times on one client, which was subsequently observed engaging in what appeared to be typical scanning behavior, attempting to connect to TCP ports 80 and 445 across a large number of geographically-dispersed IP addresses.

Note that Snort, designed primarily for wired networks, assumes that it can observe complete packet traces – given the limited fidelity of wireless packet capture in Argos, it is promising that a wide range of malicious activity could still be detected. These results demonstrate that there is substantial promise for the use of passive monitoring for detecting and tracking malicious behavior in urban scale settings.

7.3 Tracking public transport services

Another interesting use for Argos involves detecting and tracking public transport vehicles and their passengers. Soon after deploying Argos, we noticed the occasional appearance of SSIDs such as `Coach0228_Box-078` and `MBTA_WiFi_Coach1601_Box-139`. A bit of research, and the observation that these SSIDs were only ever captured by the sniffers in the western Cambridge cluster near a commuter rail line, lead us to the realization that these were access points installed on public commuter trains.

To test the sniffers' ability to track moving vehicles, we wrote a query to detect the passage of trains and infer their direction of travel. Sniffers perform weighted channel hopping to search for packets from any known train BSSIDs; when a train packet is detected, channel focusing is initiated to maximize the network's capture potential while the train passes. To infer the direction of travel, we rank-order the 3 sniffers located alongside the railway tracks according to the mean timestamp of the packets captured at each.

Table 4 shows one typical day's worth of data. Argos captured 2801 packets from 42 unique access points, corresponding to 2–4 APs per train. Trains were detected passing by on 34 occasions, with directionality inferred in 29 of the cases (4 of which were incorrect). In total, we have observed 456 unique users associated with the train networks and have captured 804 Web requests from passing trains.

We have also detected WiFi networks related to buses in the area, with names such as `boston_bus_600-798`, `Dartmouth Coach 0801`, and `Concord Coach 921 Left Side`. 47 bus-mounted networks have been discovered by the Argos node that is mounted near Interstate 93, a major highway leading into and out of Boston. We have detected 144 unique users and 176 web requests on these networks.

7.4 Wireless client fingerprinting

Our final case study asks the question: how much information can we infer about individual users based on their wireless network traffic? Clearly, a great deal can be learned by watching an individual user's Web surfing or email activity, a concern which is readily addressed through encryption.

However, *even if a user associates with an encrypted network*, it is possible to glean a great deal of information based on data that the 802.11 protocol transmits in the clear. An example is the contents of 802.11 probe request packets. To discover networks in its vicinity, an 802.11 client broadcasts probe requests containing the plain-text SSIDs of networks it wishes to discover; if any of these networks receive the request they send a corresponding probe response. Many operating systems remember the full set of networks that a user has previously associated with, and transmit this set of SSIDs in probe request messages. As a result, by capturing 802.11 probe requests, we can learn the locations that a user has previously visited.

To explore this, we wrote a query to capture 802.11 probe requests and the SSIDs contained within them. We detected 21,546 unique clients with a mean of 470 probe requests sent per client. Looking at the number of *unique* SSIDs observed per client, the distribution is heavy-tailed, with only a single SSID observed for the majority of clients. Nonetheless, a number of clients probed for enough unique SSIDs, to reveal significant information about their past behavior.

We manually inspected the top five clients by number of unique SSIDs; they are summarized in Table 5. For each SSID, we used the `wigle.net` wardriving database to attempt to locate each network, and found for 4 of the 5 clients

Observed		Scheduled		Observed		Scheduled		Observed		Scheduled		Observed		Scheduled	
Time	Estimate	Time	Dir.	Time	Estimate	Time	Dir.	Time	Estimate	Time	Dir.	Time	Estimate	Time	Dir.
7:10	In	7:03	In	9:55	Out	9:52	Out	16:24	In	16:19	In	19:48	Out	19:48	Out
7:34	**Out***	7:30	In	10:11	(none)	10:04	In	16:52	(none)	16:52	Out	19:51	In	19:48	In
7:46	Out	7:39	Out	11:32	Out	11:32	Out	16:58	In	16:56	In	20:36	In	20:36	In
7:48	In	7:47	In	11:43	In	11:41	In	17:05	**In***	17:02	Out	20:58	Out	20:57	Out
8:18	(none)	8:11	In	12:41	In	12:36	In	17:35	Out	17:32	Out	21:37	In	21:36	In
8:33	Out	8:30	Out	13:36	Out	13:32	Out	17:53	(none)	17:51	In	22:55	(none)	22:52	Out
8:57	In	8:40	In	14:20	**Out***	14:19	In	17:56	**In***	17:52	Out	23:37	In	23:34	In
9:08	Out	9:07	Out	15:14	Out	15:12	Out	18:33	Out	18:32	Out	00:25	Out	00:22	Out
9:22	In	9:18	In	16:13	Out	16:12	Out								

Table 4. Train schedule (time and inbound/outbound direction) inferred from captured traffic (left) compared to the true schedule published on the MBTA website (right) for Dec 9, 2009.

Rank	*Total probe reqs*	*Unique SSIDs*	*Locatable SSIDs*
1	7431	49	28
Locations: *MBTA trains, Portland OR, Acton MA, Austin TX*			
2	87	48	11
Locations: *Harvard, BU, MIT*			
3	370	46	10
Locations: *Manchester UK, Belgium, Tulsa OK, Chicago IL, ...*			
4	632	47	10
Locations: *Little Rock AR, Sacramento CA, Atlanta GA ...*			
5	120	47	0
Locations *(none identified)*			

Table 5. Locations previously visited by users, inferred from probe requests.

that many of the SSIDs had a unique geographic location. The table lists some of the locations that each user is inferred to have traveled based on their probe requests.

Clearly, this is a case where the 802.11 protocol is revealing potentially sensitive information about a user's whereabouts, well beyond the sensing range of the Argos network itself. Most WiFi users are probably unaware of this feature of 802.11. It is worth underscoring that this information is revealed even if the user only ever associates with encrypted networks. The use of probe requests to track users has been previously proposed, although in a different context [37]; instead of considering clients' past behavior, as we do, the researchers instead used the set of SSIDs that each user probes for as a way to uniquely identify that user even if they spoof their MAC address.

The tracking techniques described here can become more powerful when used in conjunction. For example, user #1 above seems to be a frequent MBTA train user. We could conceivably perform (i) fine-grained spatial and temporal tracking over short distances (by mapping networks they are observed to associate with), (ii) inference of spatial and temporal behavior over moderate distances (by combining their past behavior with MBTA train schedules), and (iii) coarse-grained spatial (but not temporal) tracking over long distances (via mapping of probe requests' SSIDs). There are severe implications for user privacy if this technique were employed on a wide scale.

8 Future Work and Conclusions

Understanding wireless network dynamics at urban scales raises new challenges for capturing, processing, and analyzing multiple streams collected from passive sniffers. Argos is a unique kind of sensor network that combines the use of in-network traffic processing, intelligent channel management, and a rich user query interface to allow users to access this complex source of ambient data. We have shown that Argos' approach to in-network traffic processing substantially reduces backhaul network load and that our dynamic channel hopping strategy improves capture coverage. Through an extensive characterization of citywide WiFi traffic and several case studies, we have demonstrated Argos' ability to support detailed analysis of the network behavior.

Several exciting areas remain as future work. The first is introducing a more rigorous notion of privacy into the user query interface. We are exploring the use of ε-*differential privacy* [35] which provides a very strong guarantee: namely, that a user issuing an aggregate query against the system cannot learn anything more about an individual user than the *a priori* information they already have. This will require that Argos only respond to statistical queries, rather than yielding specific details of individual packets or users, but we believe this could be sufficient for many users.

A second future direction involves adding support for other wireless network standards, including Bluetooth. Bluetooth devices are increasingly prevalent and much can be learned from detecting their mobility and traffic patterns. Lastly, we are interested in attempting to improve capture coverage by incorporating mobile sniffer nodes into the system (e.g. mounted on cars or buses [18]), or allowing individual users with laptops or WiFi enabled mobile phones to run Argos sniffers on their own devices; this raises the substantial challenge of managing the population of participatory sensors and integrating their data with that captured from static Argos nodes.

Acknowledgements

The authors wish to thank Geoff Mainland, Rohan Murty, Glenn Holloway, Josh Bers, Abhimanyu Gosain, Matt Tierney, Yang Gao, and Peter Salas. We would also like to thank our shepherd, Cormac Sreenan, as well as the anonymous reviewers for their comments and advice. This work was supported via NSF grant CNS-0551417 as well as generous support from IBM, Microsoft Research, and Siemens.

9 References

[1] Aircrack-ng. `http://www.aircrack-ng.org/`.

[2] Champaign-urbana community wireless network. `http://www.cuwin.net/`.

[3] Detecting conficker — the honeynet project. `http://www.honeynet.org/node/388`.

[4] Freifunk. `http://www.freifunk.net/`.

[5] Lorcon. `http://802.11ninja.net/lorcon`.

[6] Olsr. http://www.olsr.org/.

[7] Quicklz. http://www.quicklz.com/.

[8] Radiotap.org. http://www.radiotap.org/.

[9] Tcpdump/libpcap public repository. http://www.tcpdump.org/.

[10] Wigle.net. http://www.wigle.net/.

[11] Wireshark. http://www.wireshark.org/.

[12] M. Afanasyev, T. Chen, G. M. Voelker, and A. C. Snoeren. Analysis of a mixed-use urban wifi network: when metropolitan becomes neapolitan. In *IMC '08*, 2008.

[13] M. Afanasyev, T. Chen, G. M. Voelker, and A. C. Snoeren. Analysis of a mixed-use urban wifi network: when metropolitan becomes neapolitan. In *IMC '08*, 2008.

[14] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *SIGCOMM '04*, 2004.

[15] A. Akella, G. Judd, S. Seshan, and P. Steenkiste. Self-management in chaotic wireless deployments. In *MobiCom '05*, 2005.

[16] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *MobiCom '05*, 2005.

[17] N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: the insecurity of 802.11. In *Mobicom*, 2001.

[18] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *IEEE INFOCOM*, 2006.

[19] V. Bychkovsky, B. Hull, A. K. Miu, H. Balakrishnan, and S. Madden. A Measurement Study of Vehicular Internet Access Using In Situ Wi-Fi Networks. In *12th ACM MOBICOM Conf.*, Los Angeles, CA, September 2006.

[20] R. Chandra, R. Mahajan, V. Padmanabhan, and M. Zhang. CRAWDAD data set microsoft/osdi2006 (v. 2007-05-23). Downloaded from http://crawdad.cs.dartmouth.edu/microsoft/osdi2006, May 2007.

[21] B.-R. Chen, G. Peterson, G. Mainland, and M. Welsh. Livenet: Using passive monitoring to reconstruct sensor network dynamics. In *DCOSS '08*, 2008.

[22] Y.-C. Cheng. CRAWDAD data set ucsd/cse (v. 2008-08-25). Downloaded from http://crawdad.cs.dartmouth.edu/ucsd/cse, Aug. 2008.

[23] Y.-C. Cheng, J. Bellardo, P. Benkö, A. C. Snoeren, G. M. Voelker, and S. Savage. Jigsaw: solving the puzzle of enterprise 802.11 analysis. In *SIGCOMM '06*, 2006.

[24] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. 2004.

[25] U. Deshpande, T. Henderson, and D. Kotz. Channel sampling strategies for monitoring wireless networks. In *WinMee*, 2006.

[26] F. M. Halvorsen, O. Haugen, M. Eian, and S. F. Mjlsnes. An improved attack on tkip. In *NordSec*, 2009.

[27] D. Han, A. Agarwala, D. G. Andersen, M. Kaminsky, K. Papagiannaki, and S. Seshan. Mark-and-sweep: getting the "inside" scoop on neighborhood networks. In *IMC '08*, 2008.

[28] IEEE. Part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE 802.11-2007*, 2007.

[29] A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer. Understanding link-layer behavior in highly congested ieee 802.11b wireless networks. In *E-WIND '05*, 2005.

[30] D. Kim, H. Cai, M. Na, and S. Choi. Performance measurement over mobile wimax/ieee 802.16e network. In *WOWMOM '08*, 2008.

[31] E. Kohler. *The Click modular router*. PhD thesis, Massachusetts Institute of Technology, Nov 2000.

[32] D. Kotz and K. Essien. Analysis of a campus-wide wireless network. In *MobiCom '02*, 2002.

[33] Z. Li, A. Goyal, Y. Chen, and V. Paxson. Automating analysis of large-scale botnet probing events. In *ASIACCS '09*, 2009.

[34] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Analyzing the mac-level behavior of wireless networks in the wild. In *SIGCOMM '06*, 2006.

[35] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS '07*, 2007.

[36] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage. Inferring internet denial-of-service activity. *ACM Trans. Comput. Syst.*, 24(2), 2006.

[37] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall. 802.11 user fingerprinting. In *MobiCom*, 2007.

[38] R. Pang, M. Allman, V. Paxson, and J. Lee. The devil and packet trace anonymization. *SIGCOMM Comput. Commun. Rev.*, 36(1), 2006.

[39] T. S. Rappaport. *Wireless Communications: Principles and Practice (2nd Ed.)*. Prentice Hall PTR, 2 edition, Jan 2002.

[40] M. Ringwald, K. Rmer, and A. Vitaletti. Snif: Sensor network inspection framework. Technical Report 535, Department of Computer Science, ETH Zurich, Oct. 2006.

[41] M. Roesch. Snort - lightweight intrusion detection for networks. In *LISA '99*, Berkeley, CA, USA, 1999.

[42] Y. Sheng, G. Chen, H. Yin, K. Tan, U. Deshpande, B. Vance, D. Kotz, A. Campbell, C. McDonald, T. Henderson, and J. Wright. MAP: A scalable monitoring system for dependable 802.11 wireless networks. *IEEE Wireless Communications*, 15(5):10–18, October 2008.

[43] M. Shiels. Google admits wi-fi data collection blunder. http://news.bbc.co.uk/2/hi/technology/8684110.stm.

[44] B. Stone-Gross, C. Wilson, K. C. Almeroth, E. M. Belding, H. Zheng, and K. Papagiannaki. Malware in ieee 802.11 wireless networks. In *PAM*, 2008.

[45] D. Tang and M. Baker. Analysis of a local-area wireless network. In *MobiCom '00*, 2000.

[46] E. Tews and M. Beck. Practical attacks against wep and wpa. In *WiSec '09*, 2009.

[47] M. P. Wittie, B. Stone-Gross, K. C. Almeroth, and E. M. Belding. Mist: Cellular data network measurement for mobile applications. In *BROADNETS*, 2007.

[48] J. Yeo, M. Youssef, T. Henderson, and A. Agrawala. An accurate technique for measuring the wireless side of wireless networks. In *WiTMeMo '05*, 2005.