FORENSIC AND MANAGEMENT CHALLENGES IN WIRELESS AND MOBILE NETWORK ENVIRONMENTS

A Dissertation Presented

by

SOOKHYUN YANG

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2015

Computer Science

FORENSIC AND MANAGEMENT CHALLENGES IN WIRELESS AND MOBILE NETWORK ENVIRONMENTS

A Dissertation Presented

by

SOOKHYUN YANG

Approved as to style and content by:

Jim Kurose, Chair

Brian Neil Levine, Member

Arun Venkataramani, Member

Lixin Gao, Member

James Allan, Chair
Computer Science

To my parents, Leensoon Jang and Yeongcheol Yang,
and my precious cat, Hongsam Yang

ACKNOWLEDGMENTS

"Even the knowledge of my own fallibility cannot keep me from making mistakes. Only when I fall do I get up again." –Vincent Van Gogh

I would like to express my deepest gratitude to my advisor, Professor Jim Kurose, for persistently and passionately educating me to be a successful Ph.D. From him, I have learned countless invaluable lessons and philosophies (e.g., top down!) in both technical and nontechnical aspects. Before him, I was such a thankless and stubborn student. His patient guidance made me a better person and helped me to survive my personal crisis in the middle of my Ph.D. years multiple times. Through him, I could have been honored to collaborate with Professors Brian Neil Levine and Arun Venkataramani, Postdoctoral researchers Sudarshan Vasudevan and Simon Heimlicher, and also researchers in the BAE system.

I also would like to express my gratitude to Professor Brian Neil Levine for guiding me in network forensics. I have been honored to have a great opportunity to work on a novel interdisciplinary research area which helped me broaden my research spectrum beyond networking. This most challenging but most intriguing problem during my Ph.D. years taught me how to jump into a new area without fear and how to build a new notion as a first mover. I also would like to express my gratitude to Professor Arun Venkataramani for guiding me in my first-year Ph.D. research and the MobilityFirst project. His ambitious research questions helped me pursue research excellence.

I am very grateful to my family for their sincere support, my mother Leensoon Jang, and my father Yeongcheol Yang. Especially without my mother's great will, endless patience, and encouragement, I could not have survived Ph.D. Finally, I am also grateful to all my labmates in the Networks group for being together for our long Ph.D. journey.

ABSTRACT

FORENSIC AND MANAGEMENT CHALLENGES IN WIRELESS AND MOBILE NETWORK ENVIRONMENTS

SEPTEMBER 2015

SOOKHYUN YANG

B.Sc., YONSEI UNIVERSITY

M.Sc., KOREA ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Jim Kurose

The Internet recently passed an historic inflection point, with the number of broadband wireless/mobile devices surpassing the number of wired PCs and servers connected to the Internet. Smartphones, laptops, tablets, machine-to-machine (M2M) devices, and other portable devices have penetrated our daily lives. According to Cisco [8], by 2018, wired devices will account for only 39% of IP traffic, with the remaining traffic produced by wireless/mobile devices. This proliferation of wireless/mobile devices is profoundly changing many of the characteristics of network applications, protocols, and operation, and posing fundamental challenges to the Internet architecture. In light of this new trend, this thesis focuses on forensic and mobility-management challenges in wireless/mobile network environments.

The first half of this thesis addresses two network-forensic challenges that arise due to the broadcast nature of wireless communications. In the first network-forensic challenge, we develop a mechanism to detect anomalous forwarding behaviors such as packet dropping, and packet reordering, and to identify the source of forwarding-behavior attacks that can disrupt a wireless ad hoc network. Our mechanism employs witness nodes that can overhear transmissions made by nearby wireless network nodes. In the second challenge, we investigate a method for gathering network-based evidence, based on constraints imposed by current U.S. law, for remotely disambiguating a sender's network access type (wired versus wireless); such a technique could be used to determine that a sender is connected physically to a network inside a building. We discuss several factors that might affect our classification results and identify the scenarios in which residential network access type can be accurately determined.

The second half of this thesis takes a more global and network-level point of view on mobility management and delves into a clean-state approach to designing a future Internet architecture that considers mobility as a first-order property. Before discussing architectural design issues, we present a measurement and modeling study of user transitioning among points of attachment to today's Internet. These transitions could result from a user's physical mobility or a stationary "multi-homed" user's changing his/her devices or NICs. This research provides insights and implications regarding control-plane workload for a mobility-management architecture. Our measurement results to date show that users spend the majority of their time attached to a small number of networks, and that a surprisingly large number of users access two networks contemporaneously. In the last part of our thesis research, we design techniques for efficiently handling group mobility in the context of the MobilityFirst architecture [68]; MobilityFirst uses flat, globally unique names, binding a flat name to its network location via a logically centralized name- and location-resolution server. Using the empirical model from our measurement study as well as more abstract models of group mobility, we evaluate our group mobility management techniques.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS v

ABSTRACT vii

LIST OF TABLES xiii

LIST OF FIGURES xiv

CHAPTER

INTRODUCTION 1

1. WITNESS-BASED DETECTION OF FORWARDING MISBEHAVIOR IN WIRELESS NETWORKS 10

1.1 Introduction 10
1.2 Detecting Forwarding Misbehavior 12

1.2.1 Model 12
1.2.2 Evidence dissemination methods 13
1.2.3 Forwarding misbehavior attacks 13

1.3 Witness-based Detection 14

1.3.1 Evidence generation 14
1.3.2 Evidence dissemination 16
1.3.3 Judge's decision 17

1.4 Detection Properties 21

1.4.1 Forwarding misbehavior 21
1.4.2 Bypassing 23
1.4.3 Badmouthing 23
1.4.4 Collusion 24

1.5 Performance Evaluation 25

1.5.1 Metrics of detection accuracy 25
1.5.2 Numerical evaluation of detection accuracy 28
1.5.3 Communication overhead 30

1.6 Conclusion 31

2. DISAMBIGUATION OF RESIDENTIAL WIRED AND WIRELESS ACCESS IN A FORENSIC SETTING 32

2.1 Introduction 32
2.2 Investigative Method and Justification 34
2.3 Problem Statement 37

2.3.1 Factors Affecting TCP segment spacing and burst size 39

2.4 Experimental Environment and Methodology 42

2.4.1 Experimental setting 42

2.5 Properties of Classification Features 44

2.5.1 Model 45
2.5.2 Percentiles and entropy of a distribution observed at B_{remote} 49

2.6 Evaluation of Classifier Performance 52

2.6.1 Classification algorithms 52
2.6.2 Experimental procedure 53
2.6.3 Evaluating classifiers for single full-rate flow cases 54
2.6.4 Other Scenarios 56

2.7 Related Work 59
2.8 Conclusions 60

3. MEASUREMENT AND MODELING STUDY OF USER TRANSITIONING AMONG NETWORKS 61

3.1 Introduction 61
3.2 Measurement Methodology 64

3.2.1 Why IMAP mail access logs? 64
3.2.2 IMAP log collection 66
3.2.3 IMAP traces: network information 68
3.2.4 From IMAP log data to sessions 70

3.3 Measurement Analysis and Findings 73

3.3.1 Network residence time 73
3.3.2 User's multi-sessioned behavior 77

3.4 A parsimonious Markov Chain model 80

3.4.1 Markov Chain Model of User-Centric Network Transitioning 81
3.4.2 Trace properties 82
3.4.3 Model estimation and validation procedure 85
3.4.4 Prediction with aggregate user population 86
3.4.5 Prediction with user clusters 87

3.5 Related Work 91
3.6 Conclusion 92

4. GROUP MOBILITY INDIRECTION 94

4.1 Introduction 94
4.2 Group Mobility Indirection 96

4.2.1 The MobilityFirst architecture 96
4.2.2 Group-mobility indirection 97

4.3 Group-mobility-indirection design 99

4.3.1 AGUID-membership establishment 100
4.3.2 Name/location resolution semantics at the GNS database 100
4.3.3 Location-tracking signaling 102

4.4 Determining the network associated with an AGUID 103

4.4.1 Event-based algorithm for determining an AGUID's default network 103
4.4.2 Periodicity-based algorithm for determining an AGUID's default network 104
4.4.3 Optimal algorithm 105

4.5 Evaluation of Event-based Group Indirection: Synthetic Model 105

4.5.1 Synthetic group-mobility model 106
4.5.2 Quantifying the amount of signaling overhead 107
4.5.3 Numerical results 108

4.6 Evaluation: Clustering and Empirical investigation 111

4.6.1 Empirical traces 112
4.6.2 User clustering for AGUID membership identification 115

4.6.3 Performance Results: reducing the normalized signaling overhead . . . 117

4.7 Related Work . . . 120
4.8 Conclusion . . . 121

5. CONCLUSION . . . 122

5.1 Thesis Summary . . . 122
5.2 Future Work . . . 123

BIBLIOGRAPHY . . . 125

LIST OF TABLES

Table **Page**

1.1 Occurrence of False positives and False Negatives 27

2.1 DT classification results for a single flow case. 55

2.2 DT classification results with **10 Mbps wireless cases** 57

2.3 DT classification results, application-limited rates 58

2.4 DT classification results for multiple flow cases. 58

3.1 House, work, and mobile categorization of a user's home networks. 73

3.2 OIT-Spring. Four clusters resulting from the EM clustering. 90

3.3 OIT-Spring. Model-based and empirically observed state occupancies. 91

4.1 Five clusters 117

LIST OF FIGURES

Figure **Page**

1.1 FPP in the no-collusion case 28

1.2 Breakdown of the causes of false positives in the no-collusion, where $p_c = 0.5$ 29

1.3 FPP and FNP in the collusion case 29

1.4 Communication overhead (when node $x[\in X = \{C\} \cup W]$ does not retransmit ED packets) 30

2.1 An illustration of our expected network topology. 38

2.2 Our measurement network topology for experiments. 42

2.3 An example on a TCP flow's time-sequence diagram during the congestion avoidance phase. 45

2.4 CDF of segment inter-arrival times at B_{remote}, wired versus wireless access at A. (a): $T = 80$ bps ($\alpha = 0.00$); (b): $T = 2.3$ Mbps ($\alpha = 0.50$). No concatenation ($\beta = 0.00$). 49

2.5 Entropy of segment inter-arrival times at B_{remote}, wired versus wireless access at A. (a): $T = 80$ bps, $\beta = 0.00$, $\alpha = 0.00$; (b): $T = 2.3$ Mbps,$\beta = 0.00$, $\alpha = 0.50$; (c): $T = 2.5$ Mbps, $\beta = 0.17$, $\alpha = 0.83$; (d): $T = 3.5$ Mbps, $\beta = 0.78$, $\alpha = 0.83$. 50

3.1 CDF of the average number of IMAP entries per day over all users. 66

3.2 CDF of the number of observed IP prefixes associated with an ASN over all users. 68

3.3 CDF of the number of unique ASNs visited daily per user over all users. 69

3.4 CS-IMAP. Cumulative number of unique ASNs accessed by all users over time. 69

3.5 Aggregate number of sessions over all users. 72

3.6 Aggregate number of multi-sessioned time-slots over all users. 73

3.7 OIT-IMAP. Time series plot of network residence time over all users. 74

3.8 CS-IMAP. Time series plot on network residence time over all users. 75

3.9 OIT-IMAP. Box plot with whiskers with average and maximum for hourly and weekly network residence time over all users. 76

3.10 CS-IMAP. pdf of the fraction of the (three) longest residency ASNs' residence times to the total residence times. 77

3.11 pdf of ASN-based multi-session time per user. 78

3.12 Time series plot of "daily" aggregate cost of X, Y over all users (using IP prefix distinction). 83

3.13 Autocorrelation function for X_t and Y_t at different time lags (n), OIT-Spring data. 84

3.14 CS-Fall. Aggregate cost over all users. 85

3.15 OIT-Spring. pdf of aggregate cost over 5,793 users. 87

3.16 OIT-Spring. Log likelihood of cross-validation data for different numbers of clusters. 88

3.17 OIT-Spring. pdfs of aggregate cost over cluster users. 89

3.18 OIT-Spring. Q-Q plots for aggregate cost over manually picked 100 users. 90

4.1 Normalized signaling overhead ($\gamma = 0.5$): event-based algorithm for three leader strategies versus no group indirection 110

4.2 Periodicity in sequences of users' network locations: autocorrelation, entropy, and correlation 111

4.3 User frequency over pairs of first and second most frequently associated networks. 113

4.4 Best number of clusters. 116

4.5 Aggregate normalized signaling overhead over all clusters: diverse algorithms for determining an AGUID's default network versus no group indirection 118

4.6 Relation of the number of AGUIDs (resulted from manual clustering) to normalized signaling overhead. 119

INTRODUCTION

The Internet recently passed an historic inflection point, with the number of broadband wireless/mobile devices surpassing the number of wired PCs and servers connected to the Internet [8]. Smartphones, laptops, tablets, machine-to-machine (M2M) devices, and other portable devices have penetrated our daily lives. New wireless/mobile (accessory) devices such as Google glass [5] and smart watches [6] that synchronously operate with other wireless/mobile devices, are emerging. The growth in the wireless/mobile markets has been further accelerated by cloud computing that extends the capabilities of resource-limited wireless/mobile devices to be able to globally retrieve and store unlimited information anywhere, anytime from cloud storage. According to Cisco [8], by 2018, wired devices will account for only 39% of IP traffic, with the remaining traffic produced by wireless/mobile devices. This proliferation of wireless/mobile devices is profoundly changing many of the characteristics of network applications, protocols, and operation; among those of interest to us in this dissertation are:

1. With the broadcast nature of wireless communications, it is possible for devices to overhear and monitor each other, providing new opportunities for (among other things) detecting misconfiguration or misbehavior.

2. The mobility of wireless/mobile devices and the openness of mixed mode (wired/wireless) access points can disguise the physical location – inside a house versus outside a house – of a user attached to an open access point. An open access point would be more susceptible to the misuse of an uninvited stranger and an individual suspected of a crime or misuse that had been traced

to an access point could plausibly deny allegations. However, when the user is attached via a wired connection, it is easier to physically locate a user inside a house, making the resident user more easily associated with network traffic from that access point.

3. Mobility among edge and provider networks. A mobile user, a multi-homed user, or a user carrying multiple devices (or NICs) shifts his/her Internet access among multiple wired/wireless/mobile networks or contemporaneously connects to multiple networks while persistently keeping his/her identity (name). This "virtual mobility" among edge and provider networks is a more recent concern of location management and name/location translation protocols that map a user's identity to his/her current location in location-independent architectures such as Mobile-IP, XIA and MobilityFirst.

4. Group mobility – a group of (mobile) users whose mobility among networks may be correlated – occurs when users travel together (e.g., in a vehicle), when users are engaged in social relationships (e.g., affiliation, conference attendance), when users are regularly associated with a small number of preferred networks, or when multiple devices are carried by a user. Such group mobility provides new opportunities for efficiently handling user-location information for groups of users, reducing the location management and name/location translation workload – a central concern of modern location-independent architectures.

The first half of this thesis is motivated by the first two observations above, and addresses two network-forensic challenges in wireless/mobile environments. In the first network-forensic challenge, we propose a mechanism to detect anomalous behaviors and identify the source of attacks disrupting the routing/forwarding operation of a wireless ad hoc network. In the second challenge, from the perspective of law enforcement, we develop a method for gathering network-based evidence, based on constraints imposed by current U.S. law, for remotely disambiguating a sender's network access type (wired versus wireless); such a technique could be used to locate a sender inside a building.

The second half of this thesis is motivated by the last two observations above, and takes a more global and network-level point of view on mobility management and delves into a clean-state approach to redesigning network architecture that considers *mobility* as a first-order property. In the MobilityFirst architecture that we have been developing [70], a globally unique flat-ID name (denoted by GUID) is used to identify and locate an end-point or content. The name resolution functionality for binding a name and its location is migrated into a cloud infrastructure as a logically centralized service. In this thesis, we present a measurement and modeling study of user mobility among networks; this research provides a workload model that can be used in evaluating MobilityFirst and other mobility-centric architectures. Using this model as well as more abstract models of group mobility, we then design and evaluate techniques for efficiently handling group mobility in the MobilityFirst architecture. Before delving into details of this thesis, we give a brief overview of the following Chapters and the structure of this thesis.

Network-forensic challenges

Our first piece of research in wireless network forensics (Chapter 1) deals with the problem of proactively protecting a wireless network resource from an attacker's disrupting hop-by-hop forwarding in a wireless ad hoc network. The second piece of our network-forensic research (Chapter 2) considers a scenario in which a network user does not harm the network but commits crimes by misusing a network resource, such as distributing illegal content over a p2p network, or communicating with a conspirator using an open wireless access point; we suggest a legal method (according to US law) to locate a network user inside a building by disambiguating the network access type (wired versus wireless) used by a user.

Detection of forwarding misbehavior in a wireless ad hoc network

In Chapter 1, we consider the problem of detecting a malicious node that incorrectly forwards data and disrupts network connectivity in a wireless ad hoc network. Since a pre-authenticated node can be compromised and misused by adversaries, detecting a maliciously behaving node is an orthogonal problem to blocking an adversary from joining a network via secure authentication. Numerous studies [14, 33, 35, 48] have suggested methods to monitor a node's behavior for detecting a misbehaving node, such as a watchdog scheme, and secure data forwarding (SDF). However, as we will see, the detection metrics proposed in prior work can be easily thwarted by adversaries or be faulty in a lossy link. We thus suggest a witness-based detection scheme to employ a *tamper-proof* and *reliable* detection metric and enhance the accuracy of identifying a misbehaving node in a lossy network environment.

Our witness-based detection scheme employs wireless nodes near a data path that overhear a node's transmissions and gather tamper-proof evidence during reliable hop-by-hop data transmission. We describe how our scheme detects forwarding misbehavior attacks such as packet dropping, and packet reordering, and examine how our scheme successfully detects forwarding misbehavior in various threat scenarios where adversaries compromise one or more wireless nodes near a data path or on a data path. We also discuss how our scheme can identify compromised nodes. Using an analytical model, we quantify the detection accuracy in identifying misbehaving nodes on a data path and the communication overhead for detecting forwarding misbehavior attacks. We also discuss the trade-off between detection accuracy and communication overhead, compared with a previously suggested detection scheme not using wireless nodes near a data path.

This work appeared in the *IEEE Workshop on Wireless Mesh Networks 2010* [81].

Disambiguation of wired and wireless access in a forensic setting

In Chapter 2, we tackle the problem of legally determining an alleged criminal's physical location in a forensic scenario from the perspective of law enforcement. Images of child sexual exploitation have been common on BitTorrent, Gnutella, and other file-sharing networks [47, 67]. The end result of network-based investigations of these crimes is evidence that supports a court-issued warrant to enter and search the home associated with an observed IP address [45, 61]. A common alibi is that a third party used the home's open Wi-Fi. Thus, a useful first step during execution of the warrant would be to determine if contraband was distributed on the p2p network using the house's wired network, therefore making the resident user more likely to be the responsible party, or if Wi-Fi was used and that such an alibi might be justified.

We investigate methods that use remotely measured traffic to disambiguate wired and wireless residential medium access. Importantly, we place our work in a practical forensic setting by constraining our approaches to only use remotely gathered "plain view" data that can be gathered legally from p2p networks before a warrant or wiretap is required (in the US). This constraint distinguishes our work from previous research on wired/wireless disambiguation, which has assumed that measurements are taken from the target's gateway router, which is not only a much less challenging problem but impractical from a forensic setting since it violates the *Wiretap Act* [62]. Using a set of traces that we collected, we evaluate the ability of a number of classifiers to *remotely* distinguish wired from wireless access within the same house, considering several residential factors that might affect our classification results but might not be known to law enforcement before a warrant.

This work appeared in the *IEEE INFOCOM Mini-conference 2013* [77].

MobilityFirst architecture

The second half of this thesis is part of a clean-state approach to developing the MobilityFirst architecture. Our first piece of research here performs a measurement and modeling study on a user's transitioning among networks and provides insights and implications for our architectural design principles. In the second part, we describe the detailed design of a group mobility mechanism for the MobilityFirst architecture, and evaluate our approach using a model as well as empirical traces.

Characterization of a user's transitioning among networks

Physical human mobility has played a central role in the design and operation of mobile networks (including cellular, Wi-Fi, and mobile ad hoc networks) and their protocols for hand-off, routing, location management, and more. However, physical user mobility is quite different than mobility from a network or network-layer addressing point of view. This distinction between physical mobility and mobility among networks (i.e., a changing network address associated with a device or an end user) is an important one, since it is this mobility among networks that is important to location management protocols such as mobile-IP [40], HLR/VLR registration in cellular networks [10], and name/address resolution protocols in current (e.g., LISP [24]) and next generation (e.g., MobilityFirst [70], XIA [32]) network architectures and protocols.

In Chapter 3, we perform a measurement study of user-transitioning among access networks and discuss insights and implications drawn from these measurements. Based on these measurements, we also develop and validate a parsimonious Markov chain model of canonical user transitioning among networks. Our measurement study, conducted using two sets of IMAP server logs (a year-long log of approximately 80 users, and a four-month log of a different population of more than 7,000 users) quantitatively characterizes network transitioning in terms of transition rates among networks, network residency time, degree of contemporaneous connection to multiple networks, and more.

This work appeared in the *IEEE INFOCOM 2015* [78].

Group Mobility Management in the MobilityFirst Architecture

In Chapter 4, we tackle the problem of efficiently handling the signaling traffic needed to track the access networks (locations) to which each of a group of users is attached. Intuitively, there can be significant savings in location-tracking traffic and name/location translation in location-independent architectures (e.g., GNS in MobilityFirst [68]), if users move as a group and we track the location of the group, rather than each of the individuals in a group.

We introduce the notion of “group-mobility indirection” in which a single group identifier, registered in the name/location translation server, references a group of users and keeps track of the network location of the group (rather than the location of all individual users). The major complexities of group-mobility indirection occur when users associated with a group identifier are split from the group location, resulting in different users being located in different access networks at a given point in time. We describe the architectural design and algorithms of group-mobility indirection, and then evaluate the reduction in the location-tracking workload using synthetic traces (produced by our proposed group-mobility model) and using empirical traces (consisting of approximately 4000 users).

Contributions

Having overviewed this thesis, we summarize the contributions of our research as follows.

- In our research on forwarding misbehavior in a wireless ad hoc network, we show that our witness-based detection scheme can unambiguously identify a compromised node, as long as there is at least one uncompromised observing node and compromised nodes do not collude. For the case when compromised nodes do collude, we show that our scheme can detect the existence (but not identity) of compromised nodes, as long as there is at least one uncompromised observing node. In a lossy wireless network, our scheme can achieve very low false positive and false negative rates, requiring relatively low communication overhead.

- In our research on disambiguating wired and wireless access in a forensic setting, we use a simple decision-tree classifier that uses remotely measured traces and identify 25th percentiles and entropy of inter-arrival times distribution as classification features, achieving a true positive rate (TPR) of 0.9 to 1.0 and false positive rate (FPR) of 0.0 to 0.1 in our studies. Overall, our findings suggest that it is difficult at best to find a foolproof classifier for remote identification in all scenarios, but we determine the scenarios in which network access type can be accurately determined, and discuss when and why these techniques cannot be reliably used in other scenarios.

- Our measurement study of users transitioning among networks finds that users spend the majority of their time attached to a small number of access networks, and that a surprisingly large number of users access two networks contemporaneously. We also show that our Markov chain model of a canonical individual user, in spite of its many simplifying assumptions, can accurately predict aggregate transition rates, the degree of contemporaneous multi-homing, and other key network-transitioning performance metrics for an aggregate population.

- In our research on efficiently handling group mobility, we introduce signaling strategies for handling group-mobility-indirection. We show that an event-based algorithm which elects one group member as a "leader" and reactively associates the group location with this leader's location significantly reduces location-tracking traffic, as long as a group of users move together frequently enough. For the case that the sequence of networks associated with a group of users has periodicity, we show that a periodicity-based algorithm that periodically updates the group location with a predicted network location at a predicted time reduces location-tracking traffic more than the event-based algorithm. We also show the gain in reducing the location-tracking signaling as the number of groups increases.

Outline of Thesis

The rest of the thesis thesis is structured as follows. In Chapter 1, we present our witness-based detection scheme and analyze its vulnerabilities and performance, compared to a well-known data-path-based detection scheme. In Chapter 2, we discuss the legal issues of network measurement methodologies and present a technique for remotely disambiguating a suspect's access network type for geographically locating the suspect in illegal content distributing scenarios. Chapter 3 presents the results of a measurement and modeling study of user transitioning among networks. In Chapter 4, we describe our group-mobility-indirection architecture to efficiently handle a group of users moving among networks and evaluate the architecture using our group mobility model as well as empirical traces. Last, Chapter 5 concludes this thesis and discusses future work.

CHAPTER 1

WITNESS-BASED DETECTION OF FORWARDING MISBEHAVIOR IN WIRELESS NETWORKS

1.1 Introduction

This Chapter deals with our first network-forensic challenge to detect the source of an attack on forwarding behavior in a wireless MANET by leveraging the broadcast nature of a wireless channel. In a wireless ad hoc network, secure data transmission requires a mechanism to verify that an authenticated node on a path correctly forwards packets and detects a compromised node. The detection of a compromised node is particularly important in military MANETs [44, 57, 64] and other networks in which nodes can be compromised either physically or remotely. Path-verification mechanisms can be classified into two categories: *(i)* control-plane verification mechanisms that detect routing disruption attacks that inject false routing control messages, and *(ii)* data-plane verification mechanisms that deal with data forwarding misbehavior attacks such as packet drops, reordering, and message corruption. This Chapter focuses on detecting forwarding misbehavior attacks and identifying the source of attacks in the data plane.

Prior work on detecting forwarding misbehavior can be divided into two broad categories, based on whether evidence of forwarding behavior is gathered passively or actively. In watchdog schemes employing passive measurements [14, 33, 48], each node monitors its neighboring node's forwarding behavior by checking the integrity of the node's incoming and outgoing data packet pairs. Even though they attempt to precisely detect forwarding misbehavior by using a series of incoming and outgoing data packet pairs, or by gathering multiple observations from collaborating nodes, these solutions can be easily thwarted by sending out false information signed using a stolen key if a node becomes compromised. On the other hand, schemes using active measurements (e.g., [35]) involve a node actively sending a probe packet to an observing node. The absence of a response from the observing node within a predefined time is used as evidence to indicate the existence of a compromised intermediate node. These schemes are also vulnerable to an attacker generating a valid probe response using a stolen key without correctly forwarding a data packet and are complicated in a lossy network environment.

In this Chapter, we present a witness-based detection scheme that utilizes *witness nodes* located in the neighborhood of a forwarding node, that improve the accuracy of a detection metric using a new *tamper-proof* and *packet-by-packet* evidence format. This Chapter focuses on discussing the impact of having witness nodes on detection accuracy and communication overhead under diverse threat scenarios. We show that our scheme can unambiguously identify a compromised node, as long as there is at least one uncompromised observing node and compromised nodes do not collude. For the case when compromised nodes do collude, we show that our scheme can detect the existence (but not the identity) of compromised nodes, as long as there is at least one uncompromised observing node. Using an analytical model, we also show that our scheme can achieve very low false positive and false negative rates in a lossy wireless network, requiring relatively low communication overhead.

The rest of this Chapter is organized as follows. In Section 1.2, we introduce our network and detection model and threat scenarios. In Section 1.3, we describe the witness-based detection scheme in more detail. Section 1.4 shows how our scheme detects attacks in various threat scenarios. In Section 1.5, we evaluate our scheme's detection accuracy and communication overhead in the presence of lossy links. In the last Section, we conclude the Chapter.

1.2 Detecting Forwarding Misbehavior

In this Section, we describe our model with a list of assumptions, and briefly describe two detection schemes: data-path-based detection and witness-based detection. Then we define various forwarding misbehavior attacks.

1.2.1 Model

In our model, each node on a data path exchanges data and ACK packets with its next-hop neighbor as part of the normal forwarding of packets. For simplicity, we consider a *"static"* wireless ad hoc network that is composed of authenticated nodes using public-key authentication and thus the data path is fixed during a detection procedure. Without loss of generality, we consider data path $S \rightarrow A \rightarrow B \rightarrow C \rightarrow D$, where S is a source, D the ultimate destination, and A, B, C are intermediate nodes. From now on, we focus on node B's forwarding behavior verification by its upstream node A[1]. We call node A a **judge** and node B a **defendant**. A **witness** is a node that may overhear the data packet from B to C, or the ACK packet from C to B, and is located within node B or node C's transmission radius. Let W be the set of witness nodes excluding nodes B and C. As **evidence** of B's forwarding, witness nodes, C and W, will use observed data and ACK packets by B. Then judge A will make a decision on the state of defendant B using the evidence transmitted from C or W[2]. Our model assumes the following.

1. A judge node is a *"trusted"* node.

[1]The verification can operate by having each node on a data path verify the forwarding behavior of its next hop neighbor, cumulatively resulting in the verification of every node in the path from S to D.

[2]In this Chapter, we focus on a decentralized approach, i.e., each node on a source-destination path is individually responsible for monitoring the forwarding behavior of its downstream node. By having a single judge responsible for deciding the state of all nodes on a data path, our detection model can be easily implemented in a centralized way.

2. A judge node on a data path knows the next hop from its downstream node (i.e., the node two hops away along the path to the destination). Such information is readily available in the case of link-state routing algorithms and distance vector routing algorithms using source routing (e.g., DSR). As we will see, this information is used by a judge to verify whether B forwards data to a correct next-hop.

1.2.2 Evidence dissemination methods

As discussed above, the evidence of B's forwarding behavior needs to be disseminated to reach judge A. In accordance with the type of evidence or the role of witnesses, we can have two evidence-dissemination methods as follows.

- **Data-path-based detection** was suggested in [35] and only relies on nodes on the data path without the intervention of witness nodes. As the evidence of B's forwarding behavior, an ACK packet from downstream neighbor C is delivered to A via B, not considering alternative paths.

- **Witness-based detection** is our newly proposed scheme. Witnesses operating in promiscuous mode, can overhear a data packet as well as an ACK packet as evidence, and transmit this evidence through diverse paths to node A. For decision making, node A utilizes data and ACK packets as evidence received from both defendant node B and a witness node in set W. If there are no witnesses in range of nodes B and C, then this approach is the same as the above data-path approach.

1.2.3 Forwarding misbehavior attacks

In our threat scenario, we consider the case that a previously authenticated node is compromised and its private key is stolen. The compromised node then launches one or more of the following forwarding misbehavior attacks:

- *Drop (Blackhole or Grayhole)* is an attack in which a compromised node does not forward a data packet. This drop attack includes complete, partial, or selective dropping of packets.

- *Fake forwarding* is an attack in which a compromised node forwards a data packet to a nonexistent node.

- *Route deviation* is an attack in which a compromised node forwards a data packet to an incorrect next-hop neighbor.

- *Power control* is an attack in which a compromised node forwards a data packet with insufficient transmission power, causing the data packet to be unreachable to its nexthop neighbor on the data path.

- *Reorder (Jellyfish)* is an attack in which a compromised node forwards a data packet out-of-order.

- *Message corruption* is an attack in which a compromised node corrupts a message field in a data packet.

1.3 Witness-based Detection

In this Section, we explain the details of our witness-based detection scheme that consists of three sequential steps: *(i)* evidence generation, *(ii)* evidence dissemination, and *(iii)* judge's decision. Without loss of generality, we focus our attention on verifying the correctness of B's forwarding behavior from now on.

1.3.1 Evidence generation

In the evidence generation step, tamper-proof evidence is generated as part of packet forwarding, ensuring that the evidence of B's forwarding behavior can be created only if B indeed forwards that data packet, whose format is as follows:

$$M, K_B(\theta_B), addr(A) \tag{1.1}$$

As shown in (1.1), data forwarder B sends a packet that adds an evidence field $K_B(\theta_B)$, and a judge-address field $addr(A)$, into a generic data packet which originally contains message M – a copy of source S's message in correct forwarding. Let θ_B be a message checksum generated by B that is a one-way hash of message M and its next-hop recipient address; for instance $\theta_B = H[M|addr(C)]$. The evidence field, denoted by $K_B(\theta_B)$, is the checksum signed with node B's private key K_B. The judge-address field is the address to which evidence associated with this packet is to be sent. (1.1) claims that B sends M to C and such evidence needs to be delivered to A.

Once B forwards this data packet conveying message M to C, nodes C and W who receive or overhear B's data forwarding may have **Data-based evidence** or **ACK-based evidence** per data packet, as we will see below. Data-based and ACK-based evidence contain the claims made by two independent communication parties, node B and the node who has overheard or received B's data packet, so that this can block either B or other nodes from fabricating valid evidence without collusion between each other.

Data-based evidence. A witness node w in set W that successfully overhears a data packet constructs Data-based evidence in the following manner:

1. $K_B(\theta_B)$: w extracts $K_B(\theta_B)$ from the data packet formatted in (1.1). *Note that $K_B(\theta_B)$ means that B claims that B forwarded M to its next-hop recipient C.*

2. θ_w: w computes its own message checksum, denoted by θ_w, from the overheard data packet, without decrypting $K(\theta_B)$ with B's public key. Thus θ_w is used for w to claim that w has overheard that B forwarded M to C.

3. t_w: w records the timestamp[3], denoted by t_w, when it successfully overhears the data packet.

4. concatenates and signs the above three pieces of information using its private key (denoted by K_w), and returns $K_w(K_B(\theta_B), \theta_w, t_w)$ as Data-based evidence. Thus, $K_w(K_B(\theta_B), \theta_w, t_w)$ means that w claims that w has overheard that B forwarded the data packet that contained M and B's claim of its correct forwarding at time t_w.

ACK-based evidence. Using the same procedure as in Data-based evidence, node C generates ACK-based evidence, denoted by $K_C(K_B(\theta_B), \theta_C, t_C)$. Thus, $K_C(K_B(\theta_B), \theta_C, t_C)$ means that C claims that C has received the data packet that contained M and B's claim on its correct forwarding at time t_C.

1.3.2 Evidence dissemination

In this step, nodes W and C transmit Data-based and ACK-based evidence to A using new control packets as follows.

- ED **packet** is used by W and C to convey evidence to A.

- ED_ACK **packet** is used by A to acknowledge the receipt of the ED packet and force C and W to retransmit evidence when node A did not receive the evidence. The packet contains the message checksum signed by A for preventing malicious ED_ACK packets sent by a node other than node A. The ED_ACK packet traverses the data path (A-B-C) to reach all nodes participating in evidence dissemination.

Since communication overhead during this step is directly proportional to witness node density, *randomized feedback suppression* among witness nodes might be used. Witness

[3]The timestamp is used only if the network needs to detect a reorder attack, with the assumption that there is a global clock in a network, or Lamport-like time ordering. The timestamp can be also useful to block evidence replay attacks.

nodes stay idle for a random time up to a maximum backoff duration. Witness nodes are suppressed if they overhear an ED_ACK packet that contains more than or equal to the amount of evidence that they have to transmit. Highest priority might be given to a witness node having both Data-based and ACK-based evidence by setting the shortest maximum backoff duration.

1.3.3 Judge's decision

After the above two steps, judge node A will have Data-based and ACK-based evidence from W and C, respectively. Judge A uses a decision making algorithm that exposes the presence of B's forwarding misbehavior attacks, and identifies the source of misbehaving nodes, extensively considering the cases when one or more than one nodes among B, C, and W are compromised[4].

We introduce the notion of *evidence consistency.* We let $\xi_x^n = K_x(K_B(\theta_B), \theta_x, t_x)$ be evidence that is associated with data packet n and is generated by node $x[\in W \cup \{C\}]$. We let θ_x^n be a message checksum that is produced by x and is associated with data packet n, and let t_x^n be the timestamp when x sends data packet n, or evidence associated with data packet n. We let $D[\theta_x^n]$ and $D[t_x^n]$ be a message checksum and a timestamp that are decrypted from evidence ξ_x^n, respectively. When node A receives evidence ξ_x^n, A evaluates the following.

$$f(\xi_x^n) \;\leftarrow\; \{(\theta_A^n == D[\theta_B^n]) \wedge (\theta_A^n == D[\theta_x^n])\} \wedge \{(t_A^n < D[t_x^n]) \wedge \left(D[t_x^{n-1}] < D[t_x^n]\right)\}$$

$f()$ first compares the equality of message checksum in the first and second terms. The remaining terms check the correctness of message order. If $f(\xi_x^n)$ evaluates to true, we say that evidence ξ_x^n is *consistent.* Otherwise, we call ξ_x^n *inconsistent.*

[4]We also consider a Sybil attack, i.e., launching attacks using multiple fake IDs, as our threat scenario. Note that a Sybil attack is equivalent to the case of multiple compromised nodes.

Node A runs one of two decision-making algorithms based on evidence consistency, depending on whether or not collusion is possible. Algorithm 1 assumes there is no collusion among misbehaving nodes, and determines if node B is compromised, uncompromised or suspicious. Under the assumption of collusion among misbehaving nodes, Algorithm 2 exposes the presence of one or more compromised nodes among node B, node C, and a set W. The two algorithms make a precise decision using three cheat-proof lemmas below with the following assumptions:

1. Node B has at least one downstream neighbor that can generate ACK-based evidence.

2. Generated evidence successfully reaches node A using one or more reliable paths.

3. At least one uncompromised node x is present.

Lemma 1 (Absence of evidence)**.** The absence of evidence implies that defendant node B is compromised.

Proof. The absence of evidence results from defendant B's drop or power control attack, loss of evidence in transit to judge node A, or the failure of a downstream neighbor to reply with an ACK packet. Given assumptions 1) and 2) above, we infer that absence of evidence implies that node B launches a drop or a power control attack. □

Lemma 2 (Existence of consistency)**.** Suppose that there is no collusion (defined in Section 1.4). Consistent evidence exists if and only if defendant node B is not compromised.

Proof. ($\Rightarrow$) We first prove that the existence of consistent evidence implies that node B is not compromised by proving its contrapositive version: if node B is compromised, then no consistent evidence can be generated. Suppose that node B is compromised. Without collusion, node x does not know if the sequence of message, message field, or next-hop recipient address in node B's data packet is correct or not. Node x can generate a message checksum and timestamp either randomly or using the incorrect message field or recipient address of an incorrectly forwarded data packet by node B. In either case, $f(\xi_x^n)$ evaluates to false by node x's message checksum (θ_x) or timestamp (t_x).

($\Leftarrow$) Given assumptions 2) and 3) above, if node B is not compromised, $f(\xi_x^n)$ evaluates to true because uncompromised node x puts the correct message checksum and timestamps into evidence.

Thus, the lemma holds. $\square$

Lemma 3 (Existence of inconsistency)**.** The existence of inconsistent evidence implies the presence of one or more compromised nodes among nodes B, C, and set W of witness nodes.

Proof. We prove this lemma by using its contrapositive version: if there are no compromised nodes among nodes B, C, and set W, then inconsistent evidence does not exist. Suppose that there are no compromised nodes among nodes B, C and set W. For every evidence generated by nodes B, C, and set W, $f(\xi_x^n)$ evaluates to true. Thus, the lemma holds. $\square$

Algorithm 1 Identify defendant node's state (w/o collusion)

```
1:  $\chi^n \leftarrow$ **false**
2:  **while** Timeout for which A awaits evidence does not expire **do**
3:      $\chi^n \leftarrow \chi^n \vee f(\xi_x^n)$
4:      **if** ($\chi^n$ == **true**) **then**
5:          **if** ($x == C$, i.e., $\xi_x^n$ is ACK-based evidence) **then**
6:              **return** uncompromised
7:          **else**
8:              **return** suspicious
9:          **end if**
10:     **end if**
11: **end while**
12: **return** compromised
```

In Algorithm 1, if χ^n evaluates to true (in other words, there exists at least one consistent piece of evidence) and if the evidence is ACK-based evidence, Algorithm 1 decides that the defendant node is not compromised, in accordance with Lemma 2. However, if χ^n is true but all evidence is Data-based evidence, node A classifies node B's state as suspicious, since the absence of ACK-based evidence from node C leaves open to the possibility of a power-control attack by node B, where node B transmits with just enough power to be overheard by the witness nodes, but not high enough to reach node C. Last, if χ^n is not true, node B is determined to be compromised, based on Lemmas 1 and 2.

Algorithm 2 Expose the existence of attacks (w/ collusion)

```
1: $\chi^n \leftarrow$ **true**
2: **while** Timeout for which A awaits evidence does not expire **do**
3:     $\chi^n \leftarrow \chi^n \wedge f(\xi_x^n)$
4:     **if** $\chi^n ==$ **false then**
5:         **return** existent
6:     **end if**
7: **end while**
8: **if** No evidence has been received **then**
9:     **return** existent
10: **end if**
11: **if** ACK-based evidence has been received **then**
12:     **return** non-existent
13: **else**
14:     **return** suspicious
15: **end if**
```

Algorithm 2 exposes the presence of a malicious attack potentially launched in collusion by two or more among nodes B, C and set W of witness nodes. If inconsistent evidence is received at line 4 or no evidence exists at line 8, Algorithm 2 declares that an attack exists based on Lemma 3 and 1 respectively. On the other hand, if a judge node only receives consistent evidence at line 11, Algorithm 2 declares that there is an attack or that some node is suspicious, depending on whether or not ACK-based evidence is present.

1.4 Detection Properties

We show how Algorithm 1 identifies a compromised node launching various forwarding misbehavior described in Section 1.2. Additionally, we show that Algorithms 1 and 2 are invulnerable to three new attacks defined below, which attempt to circumvent the witness-based detection scheme:

- **Bypassing**: Compromised defendant node B that launches forwarding misbehavior attempts to circumvent the witness-based detection scheme by including a false message checksum or a false address of a judge in a data packet.
- **Badmouthing**: Compromised node x generates evidence that falsely accuses uncompromised defendant node B.
- **Collusion**: Compromised node x and compromised node B generate fake consistent evidence together by including a false message checksum or a false timestamp to conceal node B's forwarding misbehavior.

1.4.1 Forwarding misbehavior

We describe how our witness-based detection scheme identifies misbehaving defendant node B based on Algorithm 1, when the remaining nodes behave correctly and do not launch the three attacks described above.

- A **drop** attack results from node B not forwarding data packet n and causes the absence of both Data-based evidence and ACK-based evidence at node A. From Algorithm 1, χ^n evaluates to false because node A has no evidence. Thus, node A decides that node B is compromised.
- **Fake forwarding** and **Route deviation** attacks are detected, based on inconsistent Data-based evidence, which contains different message checksums. Let us consider the scenario in which node B maliciously forwards its data packet to a node E (as opposed to node C under correct behavior). When

node E is non-existent, we term the attack as *fake forwarding*. Otherwise, the attack is called a *route deviation* attack. For every received Data-based evidence $\xi_w^n [= K_w(K_B(\theta_B), \theta_w, t_w)]$, it is easy to see that $f(\xi_x^n)$ evaluates to false, because $\theta_A^n \neq D[\theta_B^n]$ and $\theta_A^n \neq D[\theta_w^n]$, where $\theta_A^n = H[M|addr(C)]$ and $D[\theta_B^n] = D[\theta_w^n] = H[M|addr(E)]$. From Algorithm 1, node A decides that node B is compromised. A route deviation attack is also detectable based on node E's inconsistent ACK-based evidence. Fake forwarding and Route deviation attacks can be distinguished, based on whether or not node A receives ACK-based evidence.

- A **power control** attack allows for node A to receive consistent Data-based evidence. However, it also results in the absence of ACK-based evidence. After expiry of a timeout period, node A determines node B to be *suspicious.*

  Power control attacks are feasible only when node C is farther from node B than node A and each of the witness nodes (assuming homogeneous wireless signal propagation). Note that this attack requires precise distance calculations by node B to its neighboring nodes in order for it to adjust its transmission power appropriately.

- **Reorder** attacks are detected based on the timestamps in the received evidence. Suppose node B transmits data packet n before data packet $n-1$. As a result, each witness node overhears data packet n before $n-1$. Since $D[t_x^{n-1}] > D[t_x^n]$ for all evidence, $f(\xi_x^n)$ evaluates to false and node A, therefore, decides that B is compromised.

- A **message corruption** attack is detected when node A receives evidence with different checksums. Let M$'$ be the message transmitted by node B in its data packet, which is different from the original message M received from node A. For every received evidence, the message checksums are not equal, i.e.,

$\theta_A^n \neq D[\theta_B^n]$ and $\theta_A^n \neq D[\theta_x^n]$, where $\theta_A^n = H[M|addr(C)]$ and $D[\theta_B^n] = D[\theta_x^n] = H[M'|addr(C)]$. Thus, $f(\xi_x^n)$ evaluates to false and node A decides that node B is compromised.

1.4.2 Bypassing

In addition to launching forwarding misbehavior attacks, compromised node B can potentially disrupt our witness-based detection scheme by launching a bypassing attack as defined earlier. We explain why our witness-based detection scheme cannot be disrupted by a bypassing attack.

First, a false judge's address results in node A receiving no evidence because evidence-generating nodes cannot transmit evidence to judge A. This attack causes node A to decide that node B is compromised based on Lemma 1. Second, the compromised node B may attempt to manipulate the evidence field in a data packet so as to hide its forwarding misbehavior. For instance, suppose that node B launches a message corruption attack manipulating the evidence field as follows:

$$B \rightarrow \{C, W\}: \quad K_B(\theta_B), \mathrm{addr(A)}, \mathrm{M'}, \quad \text{where } D[\theta_B^n] = \theta_A^n = H[M|addr(C)]$$

Through evidence field manipulation, node B bypasses the first equality check of message checksum in equation (1.2). However, the evidence is still inconsistent, because $\theta_A^n \neq D[\theta_x^n]$ if node x is not compromised. Node A decides that node B is compromised based on Lemma 2 or 3.

1.4.3 Badmouthing

Thus far, we discussed attacks launched by node B. Let us now consider the case when node C or a witness node is compromised but node B is uncompromised. In particular, we consider the scenario where node C or a witness node includes false evidence attributes (e.g., a random message checksum or a false timestamp) or transmits no evidence despite overhearing node B's correct forwarding. This may cause node A to decide that the uncompromised node B is compromised, which produces a false positive. We refer to this attack as *badmouthing*.

In data-path-based detection, node A cannot distinguish a bypassing attack by compromised node B from a badmouthing attack by compromised node C. However, as long as at least one uncompromised witness node exists, this node can produce consistent evidence of node B's correct forwarding and allow node A to distinguish that node C is compromised, as described in Algorithm 1.

Identifying a compromised node using consistent evidence also implies that the detection accuracy of the witness-based detection scheme is unaffected by multiple compromised nodes that badmouth node B, as long as there exists at least one uncompromised node x. Our scheme results in false positive only if every witness node in the neighborhood of node B badmouths node B.

1.4.4 Collusion

Now we consider the case when compromised nodes B and x attempt to bypass the detection scheme by generating fake consistent evidence as defined earlier. For each of node B's forwarding misbehavior, we describe how nodes B and x generate fake consistent evidence to conceal node B's forwarding misbehavior as follows:

Fake forwarding, Route deviation, and Message corruption

1. Node B launches one or more of the three forwarding misbehavior above and transmits a data packet with a false message checksum $\theta_B^n = H[M|addr(C)] = \theta_A^n$.

2. After colluding node x receives or overhears the data packet from node B, node x includes a false message checksum $\theta_x^n = H[M|addr(C)] = \theta_A^n$, either by decrypting node B's false evidence field in the data packet or randomly guessing a message or message recipient. Finally, node x generates consistent but false

evidence, $K_x(K_B(\theta_B), \theta_x, t_x)$ where $D[\theta_B^n] = D[\theta_x^n] = \theta_A^n = H[M|addr(C)]$, even though node B has launched one or more of the three forwarding misbehavior.

Reorder attack

1. Node B launches a reorder attack.

2. After colluding node x receives or overhears the data packets from node B, node x includes false timestamps for the reordered data packets to conceal node B's reorder attack, and finally generates false inconsistent evidence. This step requires that the correct sequence of data packets is known to node x.

Data-path-based detection cannot detect a collusion attack, whereas the witness-based detection scheme can expose a collusion attack as long as there is at least one inconsistent piece of evidence from an uncompromised witness node, as described in Algorithm 2. That is, our scheme produces a false negative only if every node surrounding node B is compromised and colludes with node B.

1.5 Performance Evaluation

In this Section, we compare the detection accuracy of the data-path-based and the witness-based detection schemes in the presence of lossy links. We also quantify the communication overhead of uncompromised nodes in the witness-based detection scheme and study the efficacy of the feedback suppression mechanism.

1.5.1 Metrics of detection accuracy

In our detection accuracy analysis, we individually study the two decision-making algorithms: one under the assumption of collusion, and the other when the malicious nodes are assumed to act independently. The proposed model analyzes the detection accuracy as a function of the following parameters:

- p_{loss} is the probability that a node fails to receive a packet from its one-hop neighbor or overhear a neighboring node's transmission. For simplicity, we do not differentiate overhearing from data transmission and assume that both data transmission and overhearing experience equivalent interference and the same value of p_{loss}.

- For simplicity, we restrict our attention to a witness's Data-based evidence and a witness directly reaching node A. Λ is the expected number of witness nodes located in the intersection area between node A's and node B's transmission ranges, where the number of witness nodes follows a 2D-Poisson distribution with the density parameter λ.

- p_c is the probability that a node is compromised. A compromised node launches various attacks described in section 1.4 except drop or power control attacks, or evidence drop. The event of a node being compromised and the event of packet loss are thus mutually independent.

- N is the maximum number of data or ED packet retransmissions in the $data$/ACK or ED/ED_ACK exchange.

We use false positive probability (FPP) and false negative probability (FNP) as detection accuracy metrics. For each of the non-collusion and the collusion cases, Table 1.1 illustrates the conditions under which the two algorithms result in false positives (i.e., (2), (3), (7)) and false negatives (i.e., (4), (8)). In the non-collusion case, we only observe false positives. Fake consistent evidence that may generate false negatives can be only created through collusion, as explained earlier (i.e., the value of the FNP is equal to 0 in the non-collusion case). In Table 1.1(a), we divide the occurrence of false positives in the non-collusion case into two disjoint events as follows:

Received evidence	A's decision	Node B's	actual state
		Uncompromised	Compromised
At least one evidence is consistent	Uncompromised	(1) True positive	(4) False negative
No evidence	Compromised	(2) False positive	(5) True negative
All evidence is inconsistent		(3) False positive	

(a) Non-collusion case

Received evidence	A's decision	Actual attack presence	among B, C, W
		Nonexistent	Existent
All evidence is consistent	Nonexistent	(6) True positive	(8) False negative
No evidence	Existent	(7) False positive	(9) True negative
At least one evidence is inconsistent		-	

(b) Collusion case

Table 1.1: Occurrence of False positives and False Negatives

- Node A receives no evidence, given that node B is not compromised (labeled (2)).
- Node A only receives inconsistent evidence, given that node B is not compromised (labeled (3)).

In [80], we provide analytic expressions that quantify each of the FPPs and FNPs for both data-path-based and witness-based detection. As an example, assuming non-collusion, the probabilities of false positive event due to (2) in Table 1.1(a) for the data-path-based detection ($P[FP^{(2)}_{d,nc}]$) and the witness-based detection ($P[FP^{(2)}_{w,nc}]$) is given respectively by:

$$\begin{aligned} P[FP^{(2)}_{d,nc}] &= 1 - \big(1 - (1 - (1 - p_{loss})^2)^N\big) \cdot \big(1 - p^N_{loss}\big) \\ P[FP^{(2)}_{w,nc}] &= \big(1 - \big(1 - (1 - (1 - p_{loss})^2)^N\big) \cdot \big(1 - p^N_{loss}\big)\big) \times exp(-\Lambda(1 - p^N_{loss})^2) \end{aligned}$$

$P[FP_{d,nc}^{(2)}]$ computes the probability that node C's ACK-based evidence fails to reach node A via uncompromised node B. $P[FP_{w,nc}^{(2)}]$ is the probability that neither node C nor the witness nodes succeed in transmitting evidence to node A, when node B is not compromised. The first term of $P[FP_{w,nc}^{(2)}]$ is equal to $P[FP_{d,nc}^{(2)}]$, and the second term calculates witness nodes' failure of evidence transmission to node A. The probabilities of other events in the Table 1.1 are calculated similarly.

1.5.2 Numerical evaluation of detection accuracy

We now observe how the FPP and FNP vary with the expected number of witness nodes and packet loss probability in both the non-collusion and the collusion cases. Note that the case of data-path-based detection corresponds to the case of $\Lambda = 0$. In our results below, we assume that a packet can be re-transmitted up to three times ($N = 3$); once this maximum number of retransmission is reached, the packet is considered lost (dropped) by the sender and is not received at the receiver.

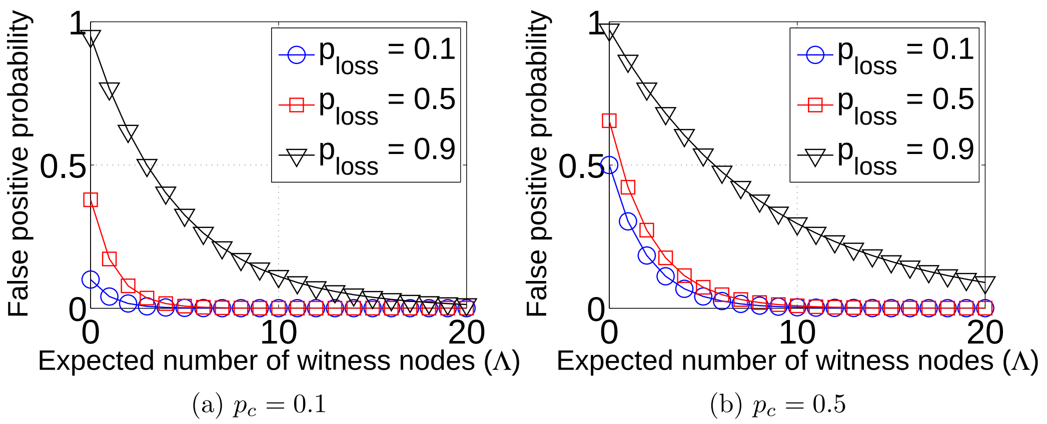
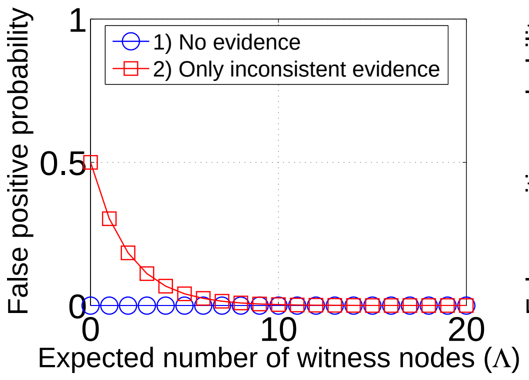
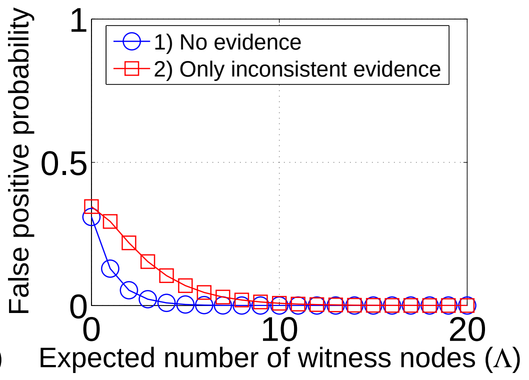


Figure 1.1: FPP in the no-collusion case

FPP in the non-collusion case: Figure 1.1 plots the FPP as a function of the expected number of witness nodes (Λ) for $p_{loss} = 0.1, 0.5, 0.9$, and $p_c = 0.1$ (Figure 1.1(a)) and $p_c = 0.5$ (Figure 1.1(b)). As expected, the FPP decreases (improving detection accuracy) as Λ increases, since an increased number of witness



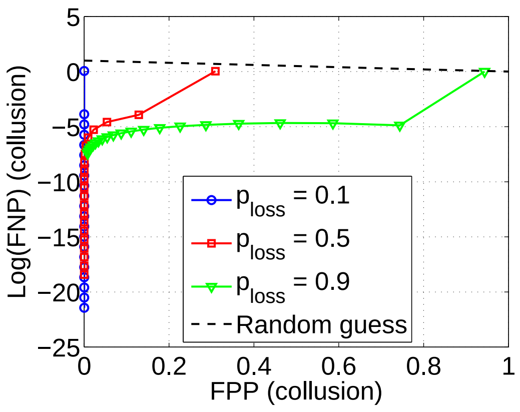
(a) $p_{loss} = 0.1$



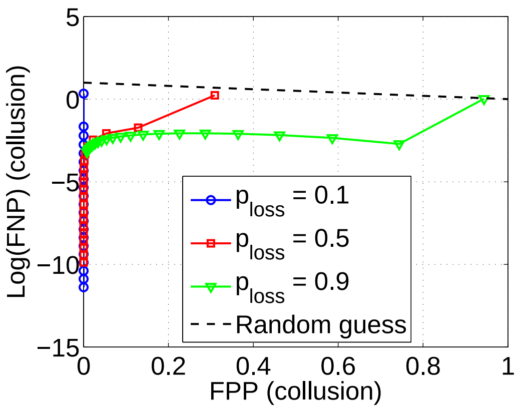
(b) $p_{loss} = 0.5$

Figure 1.2: Breakdown of the causes of false positives in the no-collusion, where $p_c = 0.5$

nodes improves the success probability of overhearing and the reliability of a path to node A, thus providing the increased evidence to the judge node. The FPP also decreases as p_{loss} decreases, for the same reason. Figure 1.2 breaks down the causes of the false positives for the case of $p_c = 0.5$. When p_{loss} is small (Figure 1.2(a)) the source of false positives is primarily badmouthing (event (3) in Table 1.1(a)), while the FPP due to lack of evidence (event (2) in Table 1.1(a)) is almost zero. When p_{loss} increases to 0.5 in Figure 1.2(b), the FPP resulting from a lack of evidence increases.



(a) $p_c = 0.1$



(b) $p_c = 0.5$

Figure 1.3: FPP and FNP in the collusion case

FPP and FNP in the collusion case: Figure 1.3 plots the FPP versus the logarithm of FNP for $p_c = 0.1$ (Figure 1.3(a)) and $p_c = 0.5$ (Figure 1.3(b)). Each curve corresponds to a different packet loss probability (p_{loss}), and each point on a curve corresponds to a different expected number of witness nodes (Λ). The highest point on each curve represents the case that $\Lambda = 0$ (data-path-based detection). As Λ increases, both the false positive and false negatives generally decrease, demonstrating the overall value of using a witness-based approach. The one exception is in the case of extremely high packet loss probability ($p_{loss} = 0.9$). False negatives slowly increase as Λ increase, unless witness density is high enough to successfully receive inconsistent evidence, which exposes a collusion attack.

1.5.3 Communication overhead

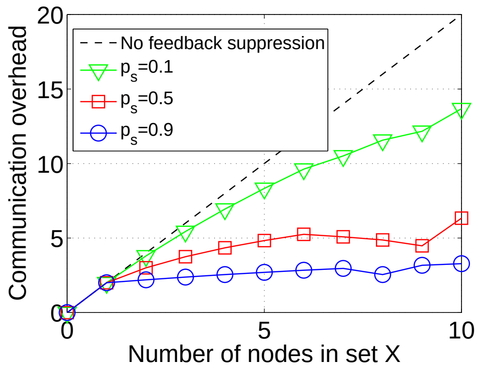


Figure 1.4: Communication overhead (when node $x[\in X = \{C\} \cup W]$ does not retransmit ED packets)

We next consider the reduced number of redundant ED and ED_ACK packets by feedback suppression. Recall that since feedback suppression is based on an ED_ACK packet from node A, it does not decrease the probability of evidence being successfully transmitted in the witness-based detection scheme. We define communication overhead as the expected number of ED and ED_ACK packets. Each solid curve in Figure 1.4 plots the communication overhead for different feedback suppression success probabilities (p_s). p_s is the probability that a node receives or overhears an ED_ACK packet. Derivations can be found in [80]. A dotted curve denotes the communication overhead without feedback suppression. Figure 1.4 shows that feedback suppression results in relatively low communication overhead, almost independent of the number of nodes in x over a wide range of values of p_s.

1.6 Conclusion

In this Chapter, we presented a witness-based detection scheme that verifies correct forwarding along data paths in wireless networks. Using observations from multiple witness nodes, our scheme either identifies a source of forwarding misbehavior (when there is no collusion among nodes) or exposes the presence of a misbehaving node (when malicious nodes collude with one another). Unlike existing schemes, the witness-based detection scheme detects misbehaving nodes without incurring significant delays, even in the presence of multiple adversaries. Under the assumption of reliable communication between nodes, we formally showed that the witness-based detection scheme does not produce false positives or false negatives, as long as there is at least one uncompromised witness node. Using an analytical model, we studied the performance of our scheme in a realistic wireless setting. Our analysis showed that witness-based detection can support low false positive and false negative rates even in the presence of highly lossy wireless links, without incurring a significant communication overhead.

CHAPTER 2

DISAMBIGUATION OF RESIDENTIAL WIRED AND WIRELESS ACCESS IN A FORENSIC SETTING

2.1 Introduction

As our second network forensic challenge, this Chapter considers the problem of remotely disambiguating wired and wireless access for identifying a target in a forensic scenario. Thousands of cases each year of child exploitation on p2p file sharing networks [47, 67] lead from an IP address to a home. A first step upon execution of a search warrant is to determine if the home's open Wi-Fi or the closed wired Ethernet was used for trafficking; in the latter case, a resident user is more likely to be the responsible party. Indeed, drive-by abuse of open Wi-Fi by criminals has been a documented practice for years [45, 61], but methods to distinguish such access are unavailable.

This Chapter thus investigates methods that use remotely measured traffic to disambiguate wired and wireless residential medium access. Importantly, we place our work in a practical forensic setting by *constraining our approaches to only use remotely gathered "plain view" data that can be gathered legally from p2p networks before a warrant or wiretap is required (in the US).* This constraint distinguishes our work from previous wired/wireless disambiguation research, which has assumed that measurements are taken from the target's gateway router, which is not only a much less challenging problem but impractical from a forensic setting since it violates the Wiretap Act [62]. Our goal is to provide information to investigators as they execute a search warrant inside a home. In addition to checking alibis and supplying information for a suspect's interview, our techniques are also useful for forensic triage. Backlogs of six months are typical for criminal forensics labs, and the easiest way to reduce the queue is to not add to it by eliminating computers from consideration (for example, those that have no wired interface) [50].

Our techniques work across the Internet by estimating the per-flow distribution of *inter-arrival times* of packets transmitted over different types of home access networks, as measured by an investigator at a remote Internet p2p client. Using a set of traces that we collected, we evaluate the ability of a number of classifiers to *remotely* distinguish wired from wireless access within the same house. We also develop a model of packet spacing for residential traffic sent via a cable modem through the Internet that illuminates and explains our classification results. We find that 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. Specifically, our analysis reveals the following:

- We use a simple decision tree classifier that uses remotely measured traces and identify **25th percentiles** and **entropy** of inter-arrival times distribution of the traces as classification features achieving a true positive rate (TPR) of 0.9 to 1.0 and false positive rate (FPR) of 0.0 to 0.1 in our studies. For Linux, we can precisely classify wired from wireless using 25th percentiles or entropy in accordance with a cable network's state. But for Windows, we can only depend on entropy as good classification features.

- High contention for a wireless channel locally at the target greatly affects classification accuracy, though this can be overcome.

- We evaluate the cases of both single and multiple p2p flows from the source, but we find that this distinction does not affect our results; only the individual upstream throughput of each flow has an impact on the classifier.

- Our classifier must be trained separately for significantly different upstream throughputs from the target; fortunately, this throughput is easily observable at the receiver. Such training can be performed when the search warrant is executed from within the house; there is no reason to train a general classifier ahead of the warrant for all houses.
- We also show that measurements from points "near" the target (i.e., in the same cable network) do not guarantee better classification results.

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

The remainder of this Chapter is organized as follows. In Section 2.2, we discuss the legal and practical issues that provide the background and motivation for the particular problem addressed in this Chapter. In Section 2.3, we define the problem setting, the classification problem, and the application, network, protocol and environmental factors impacting our work. In Section 2.4, we describe the experimental setting in which we obtained measurement traces. Section 2.5 discusses the properties of the classification features. In Section 2.6, we describe the classification algorithms that we use to distinguish wireless from wired access and then discuss our empirical evaluation of classification. In Section 2.7, we discuss related past research in the network measurement community. In the last Section, we summarize our conclusions.

2.2 Investigative Method and Justification

The general criminal procedure for child pornography (CP) cases is as follows. Investigators search for content on p2p networks.

1. CP files offered in plain view by a peer, identified by IP address, are downloaded by investigators.

2. The download provides sufficient *probable cause* as part of an application for a magistrate-issued search warrant of the home associated with the IP address's billing records.

3. The warrant is issued, and once inside the home, a triage-style search begins for evidence associated with CP, which might not be the previously downloaded content. Users of the home's computers are interviewed.

4. Seized devices are sent to an off-site lab for detailed forensic examination.

5. Evidence found during search is then used to support a criminal trial for receipt, possession, or distribution of CP.

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

Our goal is to examine whether it is possible to remotely infer the target's access type. Our technique would be used as follows. During Step (1) above, investigators would keep a packet-level trace of the file download, which is already common practice. Using the packet-level trace, investigators identify a criminal's computer setting (such as operating systems, TCP parameters and p2p applications), and characterize the file-downloading-flow's throughput and concatenation rate, as we will see in Section 2.5. This information is not a part of the warrant application. During Step (3), the classifier is trained, which can be completed in minutes with a pre-configured program and a laptop with both wired and wireless interfaces. The pre-configured program regenerates the observed flow having equivalent throughput and concatenation rate in Step (1) via wired and wireless interfaces. For accurate classification, the laptop should be equivalently configured as a criminal's computer setting. The information is used on scene to inform triage and user interviews. We note that it would only reduce accuracy to pre-train a classifier from general Internet scenarios.

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

A number of legal issues restrict the initial process of gathering the data we use to infer a target's medium access type [17,41]. First, US law prohibits government search and seizure of evidence without a warrant if and only if the source of the data has a *reasonable expectation of privacy* (REP) [4]. US courts have found consistently that users of p2p file sharing networks have no REP when investigators are peers in the network; see *U.S. v. Breese, 2008 WL 1376269* and *U.S. v. Gabel, 2010 WL 3927697.* Collecting information at a user's gateway without a warrant is certainly illegal.

Second, prior to obtaining a warrant, law enforcement cannot use technology that is not in "general public use" to obtain information that would otherwise be unavailable. This restriction is a result of *Kyllo v. U.S., 533 U.S. 27 (2001).* For example, recently the court ruled that software designed for law enforcement to monitor activity on p2p networks does not violate 4th Amendment protections since if it follows the protocol as any peer on the network does. Similarly, in *Massachusetts v. Karch (2011)*, the court ruled that law enforcement programs that do not search the remote computer, but "merely gather and evaluate publicly available information with greater efficiency and with an eye toward obtaining evidence of criminal activity" do not violate *Kyllo*, even if the software itself is unavailable for general public use.

Related work, in Section 2.7, that has been motivated by network monitoring and measurement is also governed by several US federal laws. Sicker et al. [62] provide an excellent overview and discussion of these laws and their consequences for the network traffic measurement research community. Criminal investigations are not included in that analysis since they lack the *provider protection* motive, which is measurement with the aim of protecting the network infrastructure, e.g., detecting or characterizing network attacks. In monitoring settings, clients typically consent to monitoring by the provider as part of an acceptable use policy.

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

2.3 Problem Statement

Our problem setting is illustrated in Figure 2.1. As described in Section 2.2, we begin by assuming that investigators have already identified a peer, denoted as A

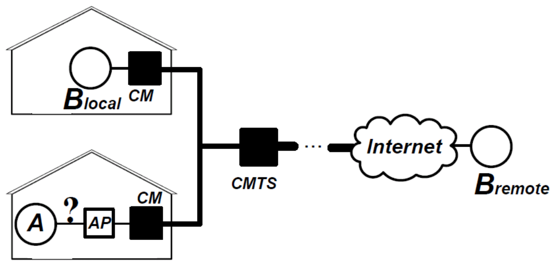


Figure 2.1: An illustration of our expected network topology.

in the figure, who is a *target* that uploads illegal content to the investigator. *Our challenge is to determine whether A is connected to the home AP via a wireless 802.11 network or via a wired Ethernet.* Investigators, denoted as B in the figure, can make this determination using only traces measured, from a remote location, either within the same cable network or in the larger Internet.

We assume the AP used by A is connected to the Internet via a cable modem (CM). The coordination system of a regional head-end, known as a Cable Modem Termination System (CMTS), regulates the use of upstream and downstream bandwidth based on A's level of contracted service with the cable network service provider. The CM communicates with the CMTS using the Data Over Cable Service Interface Specification (DOCSIS) [15, 21] protocol stack. In the downstream direction, the CMTS broadcasts data and control frames to a set of CMs. The upstream channel consists of a stream of time slots shared among CMs. Using the DOCSIS protocol, the CMTS replies to CM time-slots requests and grants time-slots to CM usings *MAP messages* every 2ms. Once a CM has acquired time slots from the CMTS, it usually transmits a TCP segment per DOCSIS frame. In the case of congestion, a CM can buffer multiple TCP segments and concatenate the segments in a DOCSIS frame after waiting for a longer time-slot-granting delay. The size of a concatenated frame is limited by a *maximum burst frame size* and TCP segments destined to different receivers can be concatenated in a frame. One manifestation of buffering at a CM has been recently noted in *bufferbloat* [28, 52].

We evaluate two locations from which the investigator B can legally make measurements. As discussed above, we assume that measurements cannot be made at A's location since that would violate the 4th Amendment or Wiretap Act protections. 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 Figure 2.1):

- 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} would be likely routed through a small number of cable network routers (and a final CM) before reaching B_{local}.

- 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 records the inter-arrival times of TCP data packets sent from A. We expect the TCP stream to be offered by A as part of a p2p file sharing application. In this case, the investigator would be typically 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.

2.3.1 Factors Affecting TCP segment spacing and burst size

Since we will use the inter-arrival times between segments to distinguish between wired and wireless access in the sender's home, let us next consider how the TCP and

DOCSIS protocols shape the time between transmission of A's TCP segments. TCP's sliding window algorithm typically results in *bursts* of packets that are sent back-to-back, i.e., with only short inter-departure times between back-to-back segments. These bursts are then separated by a relatively longer interval of time, while the sender waits for the receiver's ACK.

When the CM transmits segments, the inter-departure time between two segments can be different from those segments' inter-arrival time to the CM. As we will see shortly, these changes can be small or can be significant, and can depend on the level of congestion in the cable network. Since the segments' inter-arrival times to the CM follow their departure from the (wired or wireless) access network to the CM, and since our goal is to distinguish between wired and wireless access times based on these inter arrival times, we will want to focus on segments whose inter-departure time from the CM closely matches their inter-arrival time to the CM. In Section 2.5, we present key insights that allow us to identify segments whose inter-departure time is relatively unchanged from their inter-arrival time.

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

- **p2p application rate limit.** As a file sender, peer A is assumed to always have file data to send. In some cases, a p2p application may use a rate-limiting algorithm [63] to purposefully limit TCP throughput. In this case, TCP might not send segments fast enough to fill the congestion window. Such a rate-limited flow may have a larger number of *smaller* bursts (i.e., fewer segments with short inter-departure times) and a larger number of *longer* inter-departure times than an unconstrained TCP sender. But TCP's burst-followed-by-an-inter-burst-delay behavior - a feature we exploit in our classification - is still observed.

- **Multiple flows from A.** A p2p peer often exchanges data with multiple peers simultaneously. Since upstream bandwidth is shared among these multiple flows, each individual flow will experience a lower throughput than in a single flow scenario. This decreased throughput is evidenced in a decreased burst size and increased inter-burst spacing. We find, however, that for accurate classification, we only need determine (by measurement) the throughput of a target flow; the number of competing flows need not be known.

- **TCP send buffer size and Nagle's algorithm for OSes**. The TCP send buffer size and Nagle's algorithm play an important role in determining TCP's burst size. Linux has a large maximum send buffer size and disables Nagle's algorithm by default, and consequently can produce burst sizes adaptive to TCP's window size; conversely, Windows' burst size is often equal to its *very small* default send buffer size of 8 KB[1]. Windows buffers 8KB data and transmits the 8KB data immediately without waiting for the ACK. The buffering of data larger than Maximum Segment Size (MSS) bypasses Nagle's algorithm which is enabled by default for Windows. These TCP send buffer sizes can be overridden by p2p applications such as *eMule* and *ktorrent*, and cause a flow to have different burst size. (We verified these applications' behavior by examining their source code.)

- **Wireless channel contention.** Packets ready for departure from A must gain access to a wired or wireless medium in order to reach the local CM. Significant differences between PHY and MAC protocols of wired and wireless access networks result in distinguishable distributions of inter-frame arrival (and therefore inter-packet arrival) at the CM; these differences can survive through the Internet as we show in Section 2.5. These differences are easier to detect when local contention for the medium increases, as wireless MACs introduce much greater delays between frames during contention than Ethernet MACs.

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

2.4 Experimental Environment and Methodology

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

2.4.1 Experimental setting

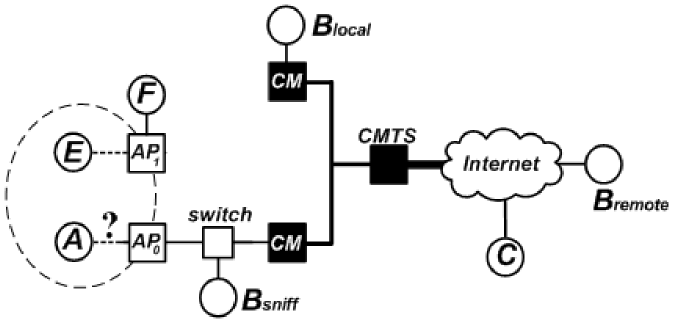


Figure 2.2: Our measurement network topology for experiments.

Figure 2.2 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* [53] to transmit TCP data at the maximum rate possible; hence the TCP transmit queue is never starved for data. At B_{sniff} we place a sniffer that captures frames before they are transmitted via the CM. We stress that B_{sniff} is used here *only for experimental research purposes here; as discussed above, our practical forensic setting would preclude making measurements at this point in practice.* Our classification results are performed using *only* traces gathered at either B_{local} or B_{remote}.

B_{remote} is located at the University of Massachusetts Amherst. A and B_{local} are located in a house in a town near UMass Amherst, using Comcast's residential cable network. B_{local} was measured to be 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 competes with traffic to B. Node C was located at Purdue

University. AP_0 is the link type we seek to classify. A's connection to AP_0 was either IEEE 802.11g with 54 Mbps or 1 Gbps Ethernet in our study. For 802.11g measurement, we located A less than 1m away from the AP_0 to obtain the wireless traces least distinguishable from wired traces. For emulating contention traffic for the wireless network, we set up an independent subnet near A as shown in Figure 2.2. 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 with 4 ticks per time-slot as upstream framing and 8,160 bytes as a maximum burst frame size. Thus, we can instantaneously see an upstream throughput of up to 10 Mbps, although the upstream capacity of a contract is 3 Mbps.

We varied the experimental environment as follows.

- **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.

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

- **Linux vs. Windows**. Our traffic source was either Ubuntu with Linux Kernel 2.6.22 or Windows Vista at node A. Linux Kernel 2.6.22 uses CUBIC [30] and Windows Vista uses CTCP [66]. In operating systems, the TCP send buffer size can be globally assigned by the kernel, or applications can individually override its maximum size using *SO_SNDBUF* socket option. Since Kernel 2.4, Linux takes 4 KB as minimum, 16 KB as default, and 4 MB as maximum, and TCP dynamically adjusts the size of the send buffer based on these three values. However, Windows Vista has a TCP send buffer size of 8 KB by default; the same 8 KB default is also found in Windows XP and Windows 7. We observed different inter-arrival time distributions for default or overridden send buffer sizes.

- **Wireless channel contention (0 Mbps or 10 Mbps)**. F continuously received 0 Mbps or 10 Mbps TCP traffic from E on the same channel as A and artificially generated 0 Mbps or 10 Mbps contention. In both cases, we characterized non-artificial wireless channel contention caused by actual in-range wireless transmitters as background traffic near A during our measurement. We used a *monitor mode* to capture all traffic generated with the same or overlap wireless channels across different SSIDs [49]. The measured background traffic was commonly less than 1 Mbps.

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

2.5 Properties of Classification Features

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

2.5.1 Model

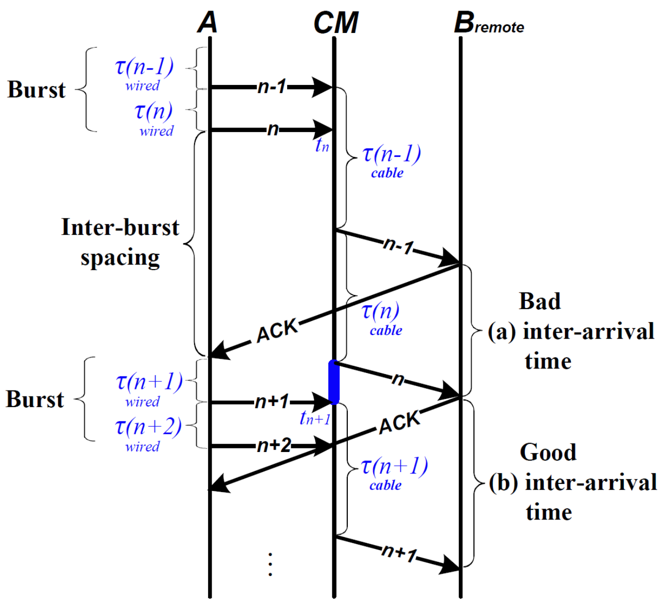


Figure 2.3: An example on a TCP flow's time-sequence diagram during the congestion avoidance phase.

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

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

Figure 2.3 shows a TCP flow's time-sequence diagram observed at A, at the CM, and at B_{remote}. A generates a series of short segment inter-departure times as a burst (sometimes referred to as a "flight" of segments) followed by a relatively longer inter-departure time. Let $\tau_{wired}(n)$ be segment n's access and transmission delays at a wired link. (τ_{wired} reflects the characteristics of the PHY and MAC protocols of wired access.) Let $\tau_{cable}(n)$ be the sum of time-slot-granting and transmission delays at the CM for segment n.

A **bad inter-arrival time**, as shown in Figure 2.3(a), occurs when the inter-segment spacing at the receiver has been completely re-shaped (from the original inter-segment spacing on arrival to the CM at A) by the CM; in this case, any difference in access times due to the wired or wireless nature of the access network at A would be lost. A bad inter-arrival time between two segments happens if *(i)* the latter segment is queued before being served at the CM, or *(ii)* the CM concatenates these two segments in a single DOCSIS frame. In Figure 2.3(a), segments n-1 and n show an example of case *(i)*. When segment n (the latter segment associated with an inter-arrival time) arrives at time t_n, the CM is serving segment n-1 and hence enqueues segment n. After segment n-1 has been transmitted, segment n is transmitted and experiences $\tau_{cable}(n)$ delay at the CM. Thus, the (bad) inter-arrival time (a) at B_{remote} equals $\tau_{cable}(n)$.

A **good inter-arrival time**, as shown in Figure 2.3(b), occurs when the inter-segment spacing is long enough to be less affected by the CM, thus preserving the difference between wired versus wireless access delay at A when the segments arrive at B_{remote}. A good inter-arrival time occurs if no segments are being transmitted (or are queued for transmission) at the CM when the second segment arrives. For instance, suppose that segment $n+1$ (as the latter segment associated with an inter-arrival time) arrives at t_{n+1} when the CM is empty, and is instantly transmitted without being queued. Then, a part of the inter-segment spacing (marked as a *"blue"* bar in Figure 2.3) is unchanged by the CM and the difference between wired versus wireless access networks would be preserved in the segment inter-arrival time at B_{remote}.

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

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

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

In our setting, 802.11g uses neither the RTS/CTS option nor CTS protection but supports frame-burst. Since 802.11g spends 322μs on transmitting a full-sized TCP segment without random-backoff and frame-burst [26], most short inter-depature times in a burst are less than 1ms at B_{sniff}. *We stress α would not be used during a forensic* *investigation, nor do we employ it in our classification procedure. However, we will find it useful to use α to explain our classifier results, as α characterizes the burstiness of the (unobservable) source.*

Throughput observed at B_{remote}. We calculate an average A-to-B_{remote} throughput observed at B_{remote} and denote it by T. We will see that a flow with a lower throughput is more likely to have more good inter-arrival times than a flow with a higher throughput, and thus is more likely to result in more accurate classification. Throughput is an important flow attribute to be considered in assessing the classification accuracy in Step (1) (described in Section 2.2). During Step (3), T would be used to generate a flow observed in Step (1).

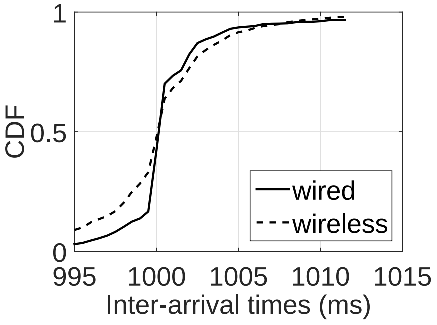
Concatenation rate observed at B_{remote}. A flow's concatenation rate (denoted by β) is the fraction of segment inter-arrival times at B_{remote} that indicate that these two segments were concatenated in a single DOCSIS frame by the CM. We calculate β using the dataset measured at B_{remote} as

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

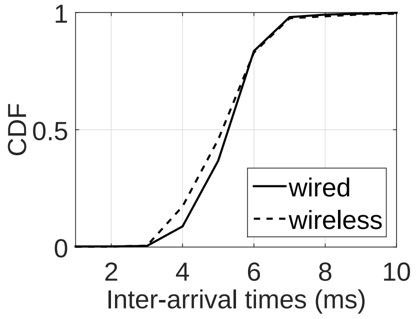
A receiver can easily identify concatenated TCP segments as those segments having an inter-arrival time of less than 1ms since the CM must wait for at least 2ms to be granted a time slot from the CMTS. This assumes that giving delays in the Internet backbone are small, which we would expect with Gigabit backbone links. As in the case of T, β would be used to generate a flow having equivalent values of T and β as observed in Step (1). As we will see, the value of β will be an important factor in deciding whether to use percentiles or entropy values of the inter-arrival time distribution observed at B_{remote} as classification features.

2.5.2 Percentiles and entropy of a distribution observed at B_{remote}

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



(a)



(b)

Figure 2.4: CDF of segment inter-arrival times at B_{remote}, wired versus wireless access at A. (a): $T = 80$ bps ($\alpha = 0.00$); (b): $T = 2.3$ Mbps ($\alpha = 0.50$). No concatenation ($\beta = 0.00$).

Percentiles. Figure 2.4(a) and (b) plot the CDF of segment inter-arrival times at B_{remote}, for the case of wired versus wireless access at A. Figure 2.4(a) is for the case of low throughput and low burstiness; these curves are for the case that the sender sends only one 10-byte segment per second. Since 1 sec is much longer than the 100ms RTT in our setting, the CM handles only a single segment at a time ($\alpha = 0$). Figure 2.4(b) is for the case of higher throughput and higher burstiness ($\alpha = 0.5$). Here, the sender transmits at the fastest possible rate using *iPerf*, but we observed that no concatenation occurred.

In Figure 2.4(a) and (b), the differences between the curves for wired and wireless access suggest those features of the segment inter-arrival time distribution that might best be used as a classifier to distinguish wired from wireless access. Specifically, we note that *the difference between wired versus wireless access networks is manifested most obviously in segment inter-arrival time percentiles lower than median.*

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

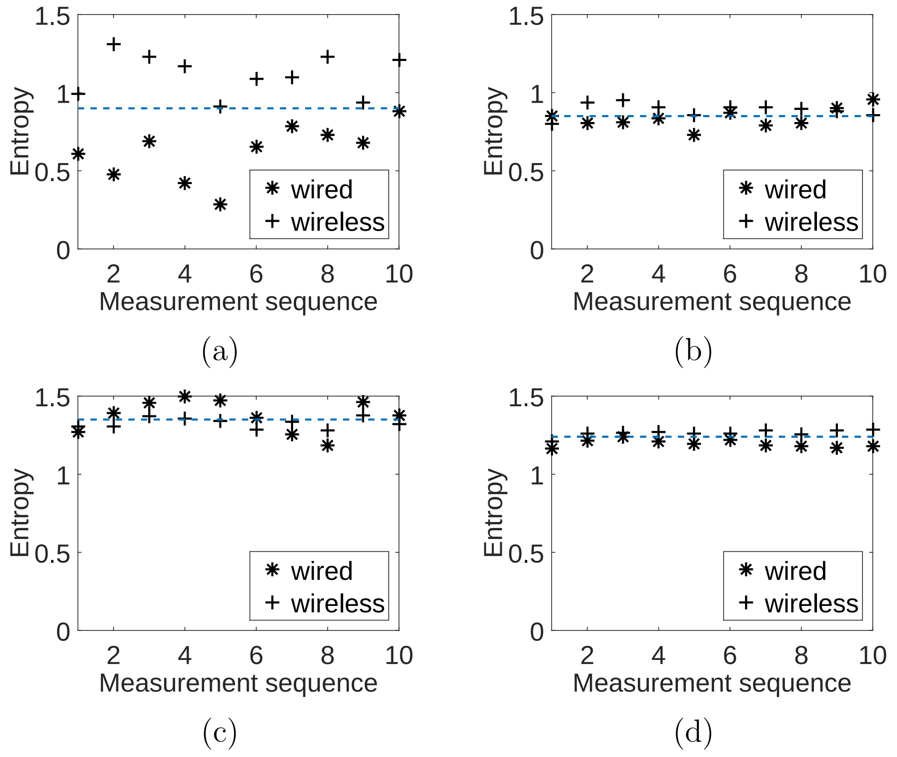


Figure 2.5: Entropy of segment inter-arrival times at B_{remote}, wired versus wireless access at A. (a): T = 80 bps, $\beta = 0.00$, $\alpha = 0.00$; (b): T = 2.3 Mbps,$\beta = 0.00$, $\alpha = 0.50$; (c): T = 2.5 Mbps, $\beta = 0.17$, $\alpha = 0.83$; (d): T = 3.5 Mbps, $\beta = 0.78$, $\alpha = 0.83$.

Entropy. Figure 2.5 shows the entropy of the inter-arrival time distribution at the receiver for ten wired and wireless access datasets with minimal and with high concatenation rates, respectively. These datasets were obtained as described in the previous Section. Figure 2.5(a) and 2.5(b) were generated in the same manner as

Figure 2.4(a) and 2.4(b), respectively. Figure 2.5(c) and 2.5(d) were generated via *iPerf* when a cable network experienced different amount of congestion.

A classifier using entropy would conceptually construct a horizontal line (representing a given entropy value) separating the wired points from wireless points in Figure 2.5. In an ideal case, all entropy values for wireless access would fall above the horizontal line and all the entropy values for wired access would fall below the horizontal line. The blue dotted lines in Figure 2.5 are the result of running the entropy-based classifier algorithms that we will discuss in the following Section. Two of these four graphs, Figure 2.5(a) and 2.5(d), show that entropy is a good classifier, but the two other graphs, Figure 2.5(b) and 2.5(c), show that entropy is not a good classifier. We conjecture the following:

- Figure 2.5(a) has small values of throughput, α and β, and indicates that there are few bad inter-arrival times. Thus, we conjecture that the inter-arrival time distributions are different enough that inter-arrival time entropy (which considers the entire distribution) is also sufficient (as is the lower percentile of inter-arrival time distributions) to distinguish wired from wireless access.

- In Figure 2.5(b), the increased throughput (see earlier discussion) results in larger bursts of arrivals (and likely queueing) at the sender, and consequently has fewer good inter-arrival times. Although entropy is not a good classifier in this case, we saw earlier that the lower percentile of the inter-arrival time distribution is a good classifier in this case.

- Figure 2.5(c) is a scenario similar to Figure 2.5(b), except with a non-negligible amount of concatenation, and thus the explanation is similar to that of Figure 2.5(b).

- In Figure 2.5(d), there are even fewer good inter-arrival times than in Figure 2.5(b). However, we conjecture that with a higher concatenation rate, many of the bad inter-arrival times are due to concatenation. Recall from our earlier discussion that concatenation leads to nearly constant inter-arrival times between segments at the receiver. Thus, when considering the *difference* in the entropy of the inter-arrival time distribution for the wired and wireless access cases, these identical, deterministic delays due to concatenation cancel each other out. This then leaves a proportionally larger fraction of good inter-arrival times that then contribute to the entropy difference of the wired and wireless access cases.

2.6 Evaluation of Classifier Performance

In this Section, we describe the classification algorithms we use to distinguish wireless from wired access at the sending host A and the experimental procedure for evaluating these classifiers using the traces described in Section 2.4. We present our empirical evaluation of *decision tree* (DT) classification and verify our conjectures above regarding the circumstances in which different classifiers would work well. We also discuss the relationship of the classification results with other factors discussed in Section 2.3.

2.6.1 Classification algorithms

The decision tree (DT) classification algorithm [59] builds a tree that predicts the output value based on several features as an input. Each interior node represents a feature, each branch descending from a feature node is one of the possible values for that feature, and each leaf is an output value. During a training phase, a DT is built by selecting the feature making the most difference to the classification at the root and testing a path to a leaf.

In the course of our research, we also ran logistic regression (LR), and support vector machine (SVM) classifiers. The LR classification produces linear decision boundaries between data using a logistic function when the predicted output has only two possible values. SVM projects data into a new space using a kernel function that seeks to create a clear gap between two possible output values and builds a hyperplane to classify data. We found that LR and SVM typically provided similar classification accuracy as DT and thus we do not report the results for LR and SVM here; see [79] for these details.

2.6.2 Experimental procedure

For each experimental trace (consisting of ten wired and ten wireless datasets taken in the same house using the methodology discussed in Section 2.4), we evaluated DT classification as follows. We trained and cross-validated the classifier using datasets of wired and wireless traffic *"without channel contention"* and investigated various features such as the 25th, 50th, 75th percentiles of the inter-arrival time distribution at the receiver, and the entropy of this distribution. We did not use cross-validation for the following two cases. *(i)* The 10 Mbps wireless access cases were evaluated using the trained model based on wired and wireless traffic without channel contention. We expect to see that the classifier trained with less wireless contention traffic performs well for classification when there is more contending traffic, since the gap between wireless and wired access features would only increase as the amount of interfering wireless traffic increases. *(ii)* A flow generated with competing flows was evaluated using the trained model based on a single rate-limited flow. During Step (3) in Section 2.2, it is not easy for an investigator to generate multiple flow cases (for training purposes). Moreover, an investigator cannot remotely distinguish a single rate-limited flow from a flow generated with multiple competing flows. We will see that a single rate-limited flow's training dataset can indeed be applied to evaluating the multiple flow case.

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

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

2.6.3 Evaluating classifiers for single full-rate flow cases

Let us begin our discussion of our DT classification results by considering what we found to the most difficult classification scenario: classifying a single full-rate flow (i.e., a flow whose sending rate is not constrained by the application, in this case, *iPerf*), when there is no artificially generated wireless channel traffic. Intuitively, we might expect this to be the most challenging case, since in this high throughput

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

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

Table 2.1: DT classification results for a single flow case.

scenario, there are a large number of long bursts and minimal inter-burst-delay. In our discussions, we distinguish between the OSes used at A (Linux, Windows), since sometimes our classification results will differ based on the OS type. Also, for the same send buffer size, we find that Windows and Linux can generate quite different values of α and β and that Windows consistently generates traffic with a non-negligible amount of concatenation. This result requires that an investigator identify the OS type and p2p application's send buffer configuration during Step (1). For a detailed discussion of OS-related issues, see [79].

Table 2.1 shows the classification results for a single full-rate flow with no concatenation (very low values of β) and higher concatenation, using the default send buffer size. For the six traces in the table, throughput saturated at a value less than the contracted upstream service rate of 3 Mbps.

Linux. Trace 1, a case with no concatenation, shows that the 25th-percentile classifier worked well but that the entropy classifier does not work well. Traces 2 and 3, cases with concatenation, show that the 25th-percentile classifier has a lower TPR

(.7) and that neither the 25th-percentile classifier nor the entropy classifier work well in both traces 2 and 3 (although entropy works well as a classifier for trace 2).

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

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

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

2.6.4 Other Scenarios

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

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

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

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

- **D4) Local measurement results.** The sender A transmits a *single full-rate flow* from B_{local} to the CM via either the wired or wireless access network.

We use the default send buffer size for Windows and Linux for above four cases. As we will see, D1, D2, and D3 cases show better, and more consistent classification performance than that of the full-rate single flow scenario.

	Linux	**Windows**
α, β	0.46, 0.00	0.81, 0.77
Features	**TPR**, **FPR**	**TPR**, **FPR**
25th-percentile	**1.0*, 0.0***	0.0, 0.0
entropy	1.0, 1.0	**1.0***, **0.0***

Table 2.2: DT classification results with **10 Mbps wireless cases**

1) 10 Mbps wireless channel contention: Table 2.2 shows the classification results for scenario D1 (10 Mbps wireless contention) using the classifiers trained by wired and wireless (without contention) access datatsets for Linux and Windows. The values of α, and β for 10 Mbps wireless datasets are shown in the table. The traces used for training purposes are indicated at the top of columns two and three. For Linux, the 25th-percentile classifier shows perfect performance in classifying 10 Mbps wireless access, a trace with no concatenation. Similarly, for Windows the entropy classifier also shows perfect performance for identifying 10 Mbps wireless datasets. We note that, as we expected, the 25th-percentile classifier does not perform well in this high concatenation case, and the entropy classifier does not perform well for the non-concatenation case.

	Linux		**Windows**	
	wired	wireless	wired	wireless
α, β T (Mbps)	0.00, 0.00 0.11(±0.00)	0.00, 0.00 0.11(±0.00)	0.81, 0.69 0.10(±0.00)	0.79, 0.68 0.10(±0.00)
Features	**TPR**	**FPR**	**TPR**	**FPR**
25th-percentile	**1.0***	**0.1***	0.1	0.1
entropy	**0.9***	**0.0***	**1.0***	**0.1***

Table 2.3: DT classification results, application-limited rates

2) Application-limited rates: Table 2.3 shows the classification results for Linux and Windows. Here, traffic was generated by transmitting one full-sized TCP segment every 100ms, thus mimicking the behavior of a rate-limited application. Comparing Tables 2.1 and 2.3, we see that classification of rate-limited flows results in more accurate classification than classification of full-rate flows for Linux and Windows. For Linux, both the 25th-percentile and entropy classifiers indeed provide acceptable classification. For Windows, a rate-limited flow consistently generates $\alpha \approx 0.8$. Thus, the entropy classifier works well but the 25th-percentile classifier does not work well.

	Linux	(Trace 7)	**Windows**	(Trace 8)	**Windows**	(Trace 9)
	wired	wireless	wired	wireless	wired	wireless
α β T (Mbps)	0.29 0.00 0.53 (±0.00)	0.28 0.00 0.54 (±0.01)	0.83 0.65 0.89 (±0.01)	0.81 0.65 0.86 (±0.03)	0.83 0.65 0.89 (±0.00)	0.81 0.65 0.85 (±0.03)
Features	**TPR**	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**
25th-percentile	**1.0***	**0.0***	0.0	0.0	0.0	0.0
entropy	0.0	0.0	**0.9***	**0.0***	**1.0***	**0.1***

Table 2.4: DT classification results for multiple flow cases.

3) Multiple flow cases: Traces 7 and 8, as shown in Table 2.4 show the cross-validated classification results for the multiple flow case. Comparing Tables 2.1 and 2.4, we see that classification of the multiple-flow scenario results in more accurate classification than classification of full-rate, single flows. We performed experiments with other multiple-flow traces and consistently found these observations across all of the experiments. Trace 9 column shows classification results for evaluating Trace 8 when we use a classifier trained based on Trace 9. Trace 9 is a rate-limited flow and has equivalent values of β and T with Trace 8, as shown in Trace 9 column. In this case, we see good classifier performance.

4) Local measurement results: We next 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 $\beta = 0.00$. At B_{local}, the classification result using 25th percentile degrades as TPR=0.7, and the entropy classifier does not perform well. 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.

2.7 Related Work

Several past studies have addressed the problem of classifying a sender's access network type using traffic measurements. Wei et al. [74] classified sender network access types into 802.11b, Ethernet, and low-bandwidth access (e.g., dial-up, cable modem, ADSL) categories, using cooperatively transmitted back-to-back UDP packet pairs between sender and receiver. Like our work, Wei et al. took measurements of packet inter-arrival times at the receiver. However, unlike our work, they assume UDP packet pairs are sent by a *cooperative sender*; instead we perform classification without the sender's knowledge, using only (p2p application) TCP traffic, with the sender potentially engaging in multiple TCP sessions with multiple receivers.

In subsequent work, Wei et al. [72, 73] monitored ACK packets exiting a university gateway and built a classifier for distinguishing between Ethernet and 802.11b/g traffic.

Gateway measurement is not possible in our forensic setting, as this would violate the Wiretap Act. In contrast, we are focused on measurements taken from outside the source's network domain. Additionally, we see our problem as more challenging: we expect bottleneck links in a heavily managed and shaped residential cable modem network to more often obscure distinctions between wired and wireless, as compared to a high-capacity university network.

More recently, Chen et al. [19] address the problem of identifying a suspect's mobile device despite being located behind a wireless AP/NAT router. However, they mark the traffic flow and sniff wireless traffic. This marking and monitoring of traffic broadcast without a warrant is a violation of the Wiretap Act. Moreover, courts require that applications to wiretap traffic meet a significantly higher standard than warrants issued to allow search and seizure of machines located in a residence.

2.8 Conclusions

This Chapter proposed legal methods that use remotely measured traffic to disambiguate wired and wireless residential medium access in a practical forensic setting, based on characteristics of the inter-arrival times in the wired and wireless access networks. We justified our method's legality based on US law and extensively considered the effect of unknown or hidden factors in a forensic setting (such as wireless channel contention, network stack parameters, and p2p application configuration) on classification performance. We identified 25th-percentile or entropy of inter-arrival times as the best performing features and figured out when and why these features worked reliably or poorly in diverse scenarios.

CHAPTER 3

MEASUREMENT AND MODELING STUDY OF USER TRANSITIONING AMONG NETWORKS

3.1 Introduction

Now we take a broader and global network-level view and turn our attention to challenges related to mobility management in a clean-state approach to designing a mobility-centric architecture. Before investigating architectural design issues in mobile environments, we first observe a study that measures and discusses user mobility among points of attachment (access networks) to the larger Internet.

"Mobility" in computer networks takes two distinct forms: physical (human) mobility among a network's access points and base stations, and *virtual* mobility of a user identity among the many access networks that make up the larger Internet. Physical human mobility has played a central role in the design and operation of mobile networks (including cellular, Wi-Fi, and mobile ad hoc networks) and their protocols for hand-off, intra-network routing and location management, and more. Consequently, numerous research studies have developed models of human physical mobility and used these models in the design and evaluation of mobile network protocols. Virtual mobility – mobility *among networks* – is a more recent concern of protocols such as Mobile-IP and new architectures such as XIA [32] and MobilityFirst [68], which aim to provide location independence (mobility transparency) by separating identifiers (names) from addresses or network locations. Here, the need to map a user's identity to his/her current network location (i.e., the access network to which the user is attached) via mobility registration and lookup/indirection protocols, are central concerns. Thus, a quantitative understanding of how a user identity transitions among access networks – the networks through which that identity is addressed and ultimately reached – is of great interest for mobility architecture and protocol design and analysis.

Recognizing the potential ambiguity between physical and virtual mobility, we will refer to a user identity moving among networks from a network-layer/addressing viewpoint as *transitioning* among networks. To appreciate this distinction, consider an individual, say Alice, often connected to the Internet via numerous different networks during the course of her day. She might begin her day reading email on a tablet, connected to the Internet via a residential DSL or cable network or a wide-area wireless network; she might later work a bit from home using a computer connected via Ethernet to her residential network and then later connect wirelessly via a smartphone to her wide-area wireless network service provider as she bikes or drives to work. At work, Alice connects via the company network, but also uses a smartphone. At the end of the day, her transitioning among networks is repeated in reverse. Together, these networks might be considered Alice's set of frequently used "home" networks. When traveling, Alice connects via a smartphone's wireless provider network and via airport, airplane, cafe, hotel and remote institutional networks. Indeed, we see that the identity that is "Alice" connects to the Internet via *many* different networks over time and is sometimes connected using different devices on different networks at the same time.

A user's transition between networks can occur in a number of different scenarios: *(i)* a user might detach from one network and attach to a new network (e.g., a user explicitly disassociating from one wireless network and then associating with a different wireless network); *(ii)* a user with multiple devices[1] might move his/her activity from a

[1]The use of multiple devices is increasing rapidly. The Pew Internet Research Project [56] notes that in addition to traditional Internet access via computers, 58% of Americans own a smartphone, with approximately 50% of these users using a smartphone as their primary Internet-connected device. 43% of Americans own a tablet, a thirteen-fold increase in ownership over four years.

device attached to one network to another device attached to a different network, or use both devices concurrently; we will refer this latter as a user being “contemporaneously connected” to two (or more) networks; *(iii)* a user with one device with multiple NICs may change the interface being used, or use multiple NICs on the single device contemporaneously (which we believe is relatively rare); *(iv)* a user may connect to a VPN, thus changing its network-visible IP address.

In this Chapter, we perform a measurement study of user-transitioning among networks and discuss insights and implications drawn from these measurements. Our study thus differs from previous mobility studies that have developed models of a single device moving between access points or base stations. Based on these measurements, we also develop and validate a parsimonious Markov chain model of canonical user transitioning among networks. Our measurement study, conducted using two sets of IMAP server logs (a year-long log of approximately 80 users, and a four-month log of a different population of more than 7,000 users) quantitatively characterizes network transitioning in terms of transition rates among networks, network residency time, degree of contemporaneous connection to multiple networks, and more. We find that users spend the majority of their time attached to a small number of access networks, and that a surprisingly large number of users access two networks contemporaneously. We also show that our Markov chain model of a canonical individual user, in spite of its many simplifying assumptions, can accurately predict aggregate transition rates, the degree of contemporaneous multi-homing, and other key network-transitioning performance metrics for an aggregate population.

Our measurements provide quantitative insight into the location management signaling overhead needed by modern and proposed name/address translation and location management protocols; our models provide the ability to design, dimension and analyze such systems. More generally, we believe that while physical mobility and the design of link-layer and intra-subnetwork handoff protocols are relatively well-understood, the behavior, modeling and measurement of users transitioning among networks and the design of protocols for managing that mobility at global scale are much less well-understood. This Chapter is an important step in deepening that understanding.

The remainder of this Chapter is structured as follows. In Section 3.2, we describe the information of our collected trace, and describe our measurement scenario and methodology. In Section 3.3, we then quantify various aspects of user transitioning and discuss insights drawn from these measurements. Section 3.4 presents and validates a parsimonious Markov chain model of canonical transitioning. In Section 4.7, we discuss related past research. Section 3.6 concludes this Chapter.

3.2 Measurement Methodology

In this Section, we first discuss the challenge of measuring user-transitioning at large scale and our decision to use IMAP logs to do so. We then provide details of the IMAP logs themselves and discuss the set of networks visited by users in our logs. We conclude this Section with a discussion of how we estimate user session lengths based on log data.

3.2.1 Why IMAP mail access logs?

Measuring user mobility between networks is itself a challenging task. Measuring network connectivity directly at the end user requires a population of users willing to install software on *each* of their network-connected devices (e.g., laptop, home/office desktops, tablet and/or smartphone), periodically monitoring/logging network connectivity on all interfaces on all devices, and then collecting measurement data. In addition to the difficulty of finding and managing such a user base, the task is technically complicated by concerns regarding battery drain for monitoring connectivity on mobile devices. For these reasons, a more centralized, server-based approach might seem preferable. In particular, since a client's connection to a server provides that client's IP address, the (possibly changing) access network used by each of the server's multiple clients can thus be easily logged at a server.

Yet there are also many challenges associated with server-side measurement of user transitioning among networks. Each server implements a *single* service/application and each user runs many services and applications. Monitoring all service and application servers is impossible - there are far too many servers, and many commonly-accessed servers are proprietary. Moreover, a user invoking multiple applications has a different "identity" in each application; correlating a user's identity on one application with his/her identity on another application is a difficult research problem [29]. From a practical viewpoint then, we ideally need a server application that *(i)* is frequently used by an online user, *(ii)* can be monitored at a non-proprietary server, and *(iii)* provides both a user "identity" (so that the same user can be tracked across multiple sessions) and the network address from which that identified user accesses that server.

Although no single application server meets this ideal, we believe that an IMAP server [22] is a compelling choice. Email checking, polling, and delivery all create entries in the IMAP server's log containing an associated client IP address, as well as a client's identifier (i.e., the email account); the email account typically remains the same across a user's devices. A user who accesses the IMAP server from a desktop while at work, and then from a mobile device while commuting, and then from a laptop at home will create IMAP logs evidencing transitions from office network to cellular provider network to home access network. Although many e-mail clients periodically and automatically access their IMAP server while online (providing a rich source of IMAP data), not all clients do so. Consequently, using IMAP logs to trace a user's transitioning among access networks may miss a network transition or underestimate the amount of time spent in a network. And email is indeed but one application (albeit a popular one). Thus, we can think of our results here informally as a lower bound on the actual amount of network-transitioning performed.

IMAP logs can be also used to discover a multi-homed user, or a user contemporaneously belonging to multiple networks via multiple devices. In the former case, if the user with a single device accesses the IMAP server using multiple NICs connected to different networks, the multi-homed IMAP accesses via these different client IP addresses (and networks) will be evidenced in the IMAP log. In the latter case, a user accessing the IMAP server from multiple devices (e.g., working and reading email on laptop or PC, while also having email pushed to a smartphone) within the same period of time will have IMAP accesses via multiple contemporaneous connections during this period of time evidenced in the IMAP logs.

3.2.2 IMAP log collection

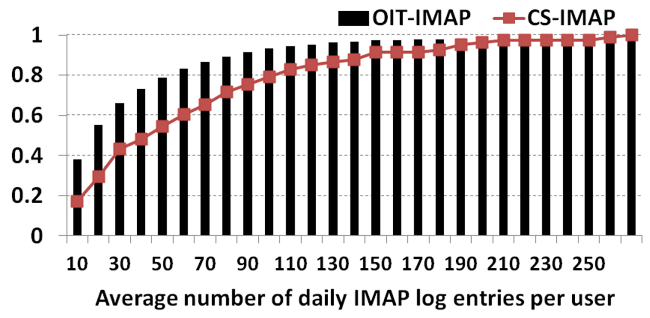


Figure 3.1: CDF of the average number of IMAP entries per day over all users.

For this study, we collected two sets of traces from IMAP servers located at the University of Massachusetts Amherst. The CS-IMAP set contains logs from IMAP servers in the Computer Science Department from Apr 14, 2013 to Feb 22, 2014; the CS-IMAP has a population of 81 users mostly consisting of CS faculty and staff members. The OIT-IMAP set contains approximately four months of logs from IMAP servers that supports a mail service for university students (primarily), faculty and staff that is separate from the CS mail service. The OIT-IMAP has a campus-wide user population of 7,137 users; these traces were taken from Dec 1, 2013 to Mar 25, 2014. The total number of CS-IMAP and OIT-IMAP log entries per user over the measurement ranged from 2 to 79,392, and from 1 to 1,490,473, respectively. Figure 3.1 plots the CDF of the average number of daily IMAP entries per user and shows that users in CS-IMAP (mostly faculty members) tend to access mail servers more frequently than OIT-IMAP (mostly students).

Each trace consists of a series of individual IMAP log entries stored by *syslog* [27], recording a user's e-mail activities, including signing into the mail server, checking the INBOX, deleting messages, and unilateral server decisions to close (idle) connections. We processed only a user's sign-in logs which allowed us to extract the following pieces of information for each entry: *(i)* user's account ID - we consistently anonymized a user's account ID (email address) using SHA2-hashing for privacy purposes, *(ii)* timestamp - the time at which a user signs into the IMAP mail server to poll, check, or retrieve email, and *(iii)* a client-side IP address - this is the IP address of user's (client-side) device when accessing the IMAP server[2].

Given an IP address, we determined the user's IP prefix network, Autonomous system number (ASN), and network domain ownership via *whois* using *whois.cymru.com* [7]. Information at *whois.cymru.com* is updated every 4 hours from the regional registries including ARIN, RIPE, AFRINIC, APNIC, and LACNIC. The CS-IMAP set contains 1,405 unique IP prefixes and 387 unique ASNs, and the OIT-IMAP set contains 9,016 unique IP prefixes and 1,777 unique ASNs. The network information for two IP addresses in the CS-IMAP and 63 IP addresses in the OIT-IMAP was

[2]Users in the CS-IMAP set occasionally accessed mail via a departmental web-based server, rather than directly from a client email application. In this case, the user's logged IP address is recorded in the IMAP log as 127.0.0.1; we analyzed the server's web logs to determine the client address of the user browser associated with this IMAP access. Only 1.6% of all IMAP web-based log entries could not be identified due to missing web logs; those entries were excluded from our analysis. VPN access to the IMAP servers is not required. Anecdotally, we believe VPN access is used primarily for accessing library and other restricted campus resources.

unknown, but the number of IMAP logs generated from such unknown IP addresses was negligible; these entries were excluded from our analysis.

3.2.3 IMAP traces: network information

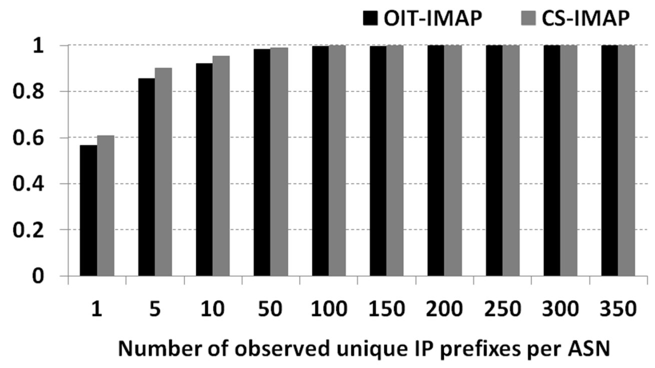


Figure 3.2: CDF of the number of observed IP prefixes associated with an ASN over all users.

Figure 3.2 shows the CDFs of the number of observed unique IP prefixes associated with an ASN over all users in the CS-IMAP and the OIT-IMAP sets. Figure 3.2 shows that approximately 61% and 57% of ASNs had only a single observed IP prefix in the CS-IMAP, the OIT-IMAP, respectively. In the traces, the following ASN and IP prefix information of frequently visited service providers have been observed (we will investigate the length of time a user is resident in an IP prefix or ASN network in Section 3.3). AT&T, Sprint, T-Mobile, and Verizon wireless are mobile access service providers. Comcast, Charter, Cox, Time Warner, and Cablevision are residential wired Internet service providers (e.g., cable and DSL access networks); the Hughes network supports a satellite Internet service used in rural communities lacking wired and cellular broadband service. The UMass Amherst network is part of the Five Colleges (AS1249) network. SAS in the CS-IMAP (a DSL and Wi-Fi service provider in France) and Unicom in the OIT-IMAP (a mobile service provider in China) were used for a non-negligible amount of time in our measurements.

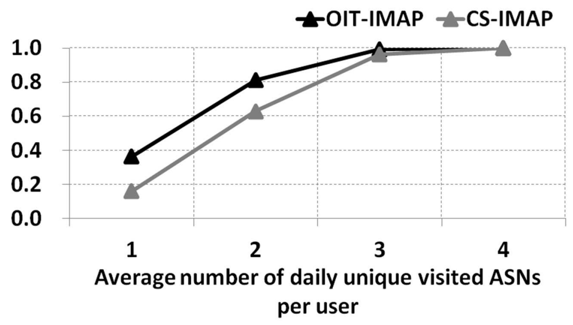


Figure 3.3: CDF of the number of unique ASNs visited daily per user over all users.

Figure 3.3 plots the CDF of the number of unique ASNs visited daily per user over all users, indicating that users in both OIT-IMAP and CS-IMAP access at most four unique ASNs in a day, but users belonging to CS-IMAP (mostly faculty members) access more ASNs than OIT-IMAP (mostly students).

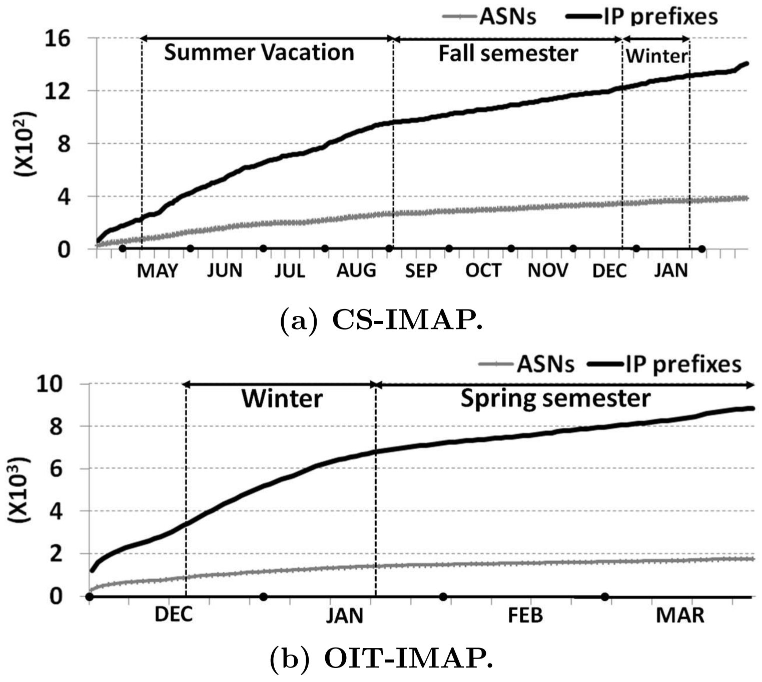


Figure 3.4: CS-IMAP. Cumulative number of unique ASNs accessed by all users over time.

Figures 3.4(a) and (b) plot the daily cumulative numbers of unique IP prefixes and ASNs accessed by all users over time. These figures indicate that the cumulative number of unique IP prefixes and ASNs each increase roughly linearly over time; the slopes of two curves during vacations (when users would be out of town more frequently) are steeper compared with the slope during the academic term. This constant increase in the daily number of new networks accessed (after the initial startup period) was initially surprising, as we had expected that users would generally access the same set of networks over time. We'll see later that a user typically does indeed spend most of the time in the same (relatively small) number of networks over time, but does visit new networks outside of this set of common networks at a roughly constant rate, resulting in the positive slope in Figure 3.4.

3.2.4 From IMAP log data to sessions

We use the notion of a *time window* to determine intervals of time during which a user is connected to a network.

Definition 1. Time is divided into consecutive *time windows,* each of length Δt. A **session** is defined as a series of consecutive time windows, each of which has one or more IMAP log entries from the *same* network (distinguished by either its IP prefix or ASN).

By Definition 1, two IMAP log entries in the same time window that have different IP addresses but the same IP prefix (or the same ASN) would be regarded as belonging to the same session. Our measurements indicate that a user may be also connected to *more than one network* during a window of time.

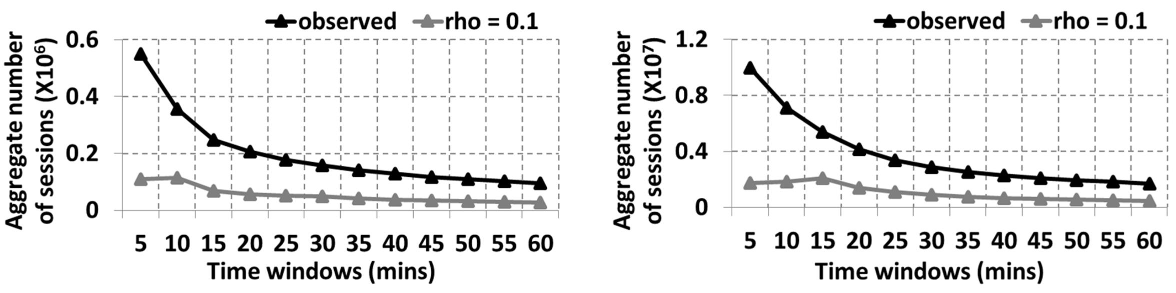
Definition 2. Given time window of length Δt, a **multi-sessioned** time window for a user is one in which that user has IMAP entries from two or more different networks (as distinguished by their IP prefix or ASN).

Choosing a value for Δt for session identification via Definition 1. If we choose a small time window value, this would break a user's single session into multiple distinct sessions separated by empty Δts having no IMAP logs entries. If a user was indeed connected during these empty Δt intervals, then we would overestimate the amount of user network-transitioning. Conversely, if the time window is too large, intervals of time during which the user disconnects and then reconnects to that same network would be coalesced into a single session, thus underestimating the amount of user transitioning. This dilemma is often faced when reconstructing user session behavior from discrete log entries [20, 54]. We choose the length of the time window Δt by observing the number of sessions as a function of Δt, as discussed below.

Another caveat for session identification using IMAP logs is that IMAP logs may not record all the network access behavior of a user. Inevitably, a session established for other purposes (not mail checks via IMAP) is not counted and thus the location tracking workload might be underestimated. Moreover, the best size of a time-window resulted from only IMAP logs might be misled; for instance, if a user sporadically checks mails but continues to use a network without being detached, then the best size of a time-window is estimated smaller than it should be, and thus the location tracking workload might be overestimated.

To investigate the challenge of choosing a value for Δt, let us define the Definition 3 below and observe the best size of a time-window for different values of ρ.

Definition 3. Given time window Δt that *(i)* contains no IMAP entries and *(ii)* fall between two time windows that contain IMAP entries, we define ρ as the probability that a user "indeed" remains connected to the network for that time window. Since we didn't know the ground truth of whether or not a user remains connected while generating no IMAP entries, we will use ρ to consider the hypothetical case that the user is connected in some fraction (ρ) of the time windows, even though no IMAP entries are generated.



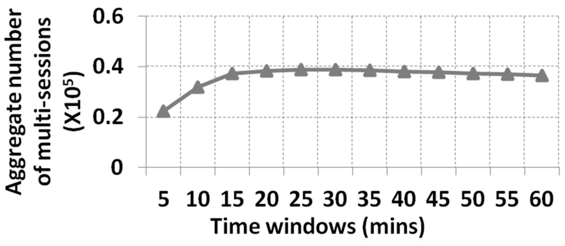
(a) CS-IMAP. **(b) OIT-IMAP.**

Figure 3.5: Aggregate number of sessions over all users.

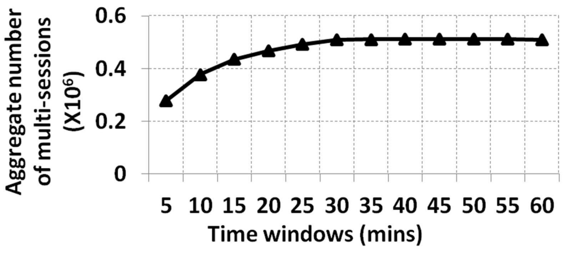
Figure 3.5 plots the total number of all users' ASN-based sessions[3] as a function of a time-window length. The black curve in Figure 3.5(a) shows that the number of sessions in CS-IMAP (as they were observed from our traces) initially decreases sharply with increasing values of Δt, and then, at around a time-window length of 15 minutes, begins decreasing more slowly. Figure 3.5(a)'s red curve plots that the hypothetical number of sessions with $\rho = 0.1$ for different time-window sizes in CS-IMAP. The red curve is significantly lower than the black curve in the inital region, and then shows a knee of the curve at 15 minutes; this pattern was also found for different values of ρ. Similarly, the knees of the curves in OIT-IMAP appears at approximately 20 minutes as shown in Figure 3.5(b). We also noted that approximately 97% of the time intervals between a user's two consecutive IMAP log entries in CS-IMAP were less than or equal to 15 minutes, and approximately 82% of the time intervals between a user's two consecutive IMAP log entries in OIT-IMAP were less than or equal to 20 minutes.

A similar analysis can be applied to the case of a user being contemporaneously connected to multiple networks. Figure 3.6 plots the total number of all users' ASN-based multi-sessioned time windows for different time-window sizes. Figure 3.6 shows that the number of multi-sessioned time windows in CS-IMAP increases until a window

[3]A comparison of using IP prefix vs. ASN distinctions to identify the number and length of sessions indicates that there is not a significant difference between IP prefix-based and ASN-based session lengths. Thus we only show ASN results.



(a) CS-IMAP.



(b) OIT-IMAP.

Figure 3.6: Aggregate number of multi-sessioned time-slots over all users.

length of 15 minutes and then flattens out and the knee of the curve appears at 20 minutes, the same knee location found in the Figure 3.5. Thus, a user who has been connected to multiple networks is likely to be completely offline for an amount of time greater than the time interval length at the knee. We will thus choose **15 minutes** in CS-IMAP and **20 minutes** in OIT-IMAP to be the length of the time window and identify user sessions accordingly via Definition 1.

3.3 Measurement Analysis and Findings

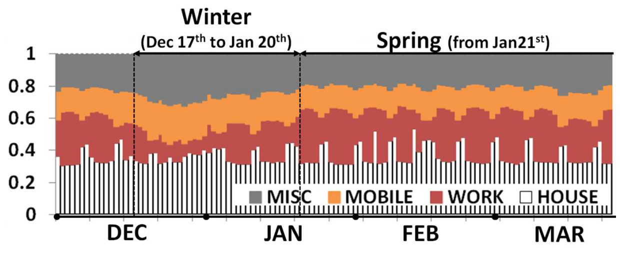
In this Section, we present and discuss our measurement results regarding user residence time in various networks and multi-sessioned behavior. The insights and implications drawn from these results form the foundation for our Markov chain modeling of user network transitioning in Section 3.4.

3.3.1 Network residence time

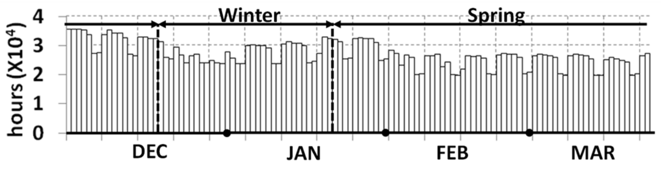
House	Comcast (AS7015, AS7922, AS33651, AS33668), Charter (AS20115), Cox (AS22773), Hughes (AS6621), Time Warner Cable (AS11351), Cablevision (AS6128)
Work	Five colleges AS (AS1249)
Mobile	Verizon (AS22394, AS701, AS6167), AT&T (AS20057, AS7018), T-Mobile (AS21928), Sprint (AS3651)

Table 3.1: House, work, and mobile categorization of a user's home networks.

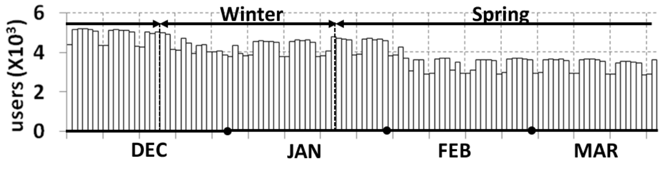
Let us first consider the *aggregate* network residence time over all users spent in various networks. Table 3.1 defines house, work, and mobile networks whose constituent ASNs are registered in U.S. and are accessed by users for more than 0.5% of aggregate network residence time. The MISC category, which includes all other network domains observed in our logs, may thus include rarely-used residential wired service provider or mobile access provider ASNs that account for negligible fractions of network residence time. Broadly, we may consider the house/work/mobile networks as a user's "home" networks and the remaining MISC networks as a user's "visited" networks.



(a) Daily fractions of network residence times.



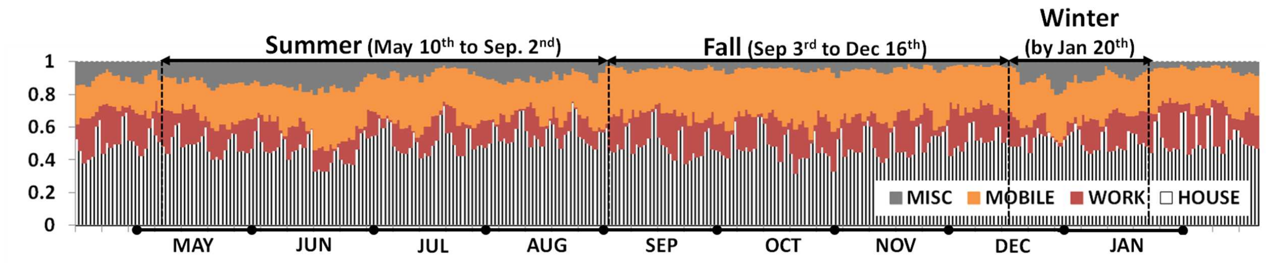
(b) Daily total network residence time.



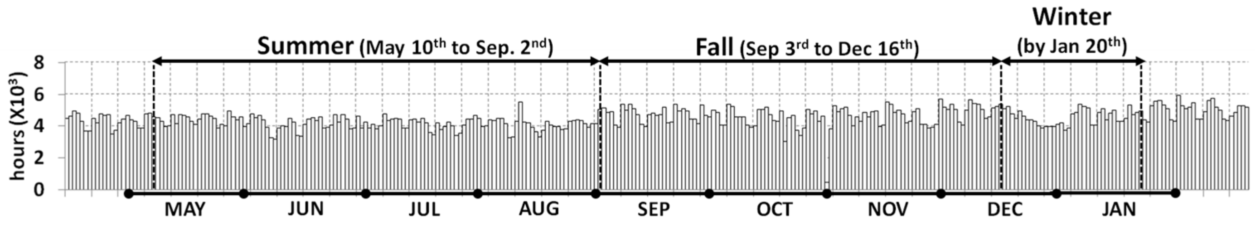
(c) Daily user population.

Figure 3.7: OIT-IMAP. Time series plot of network residence time over all users.

Figures 3.7(a), 3.8(a) plot the daily fraction of *aggregate* residence time spent in house, work, mobile and MISC ASNs over all users for OIT-IMAP, CS-IMAP respectively. Given that the house, work and mobile networks are collectively constituted by only 17 (as shown in Table 3.1) out of the 1,858 ASNs observed in CS-IMAP



(a) Daily fractions of network residence times.

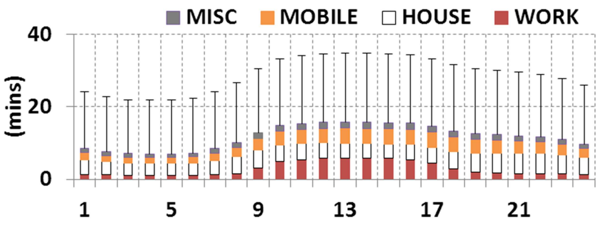


(b) Daily total network residence time.

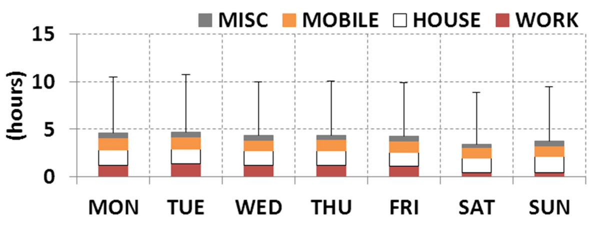
Figure 3.8: CS-IMAP. Time series plot on network residence time over all users.

and OIT-IMAP, Figures 3.7(a), 3.8(a) show that users spend the majority of their time (approximately 80% through a measurement period, and in particular more than 90% during fall semester in CS-IMAP) resident in only a small number of networks. We also observed that just two ASNs (Comcast AS7015, and Five colleges AS1249) account for more than half of the overall residency time in OIT-IMAP and CS-IMAP, and that the ten most common ASNs collectively account for approximately 85% (for OIT-IMAP) and 90% (for CS-IMAP) of the overall residency time, confirming the observation that the lion share of aggregate user time is spent in a relatively small number of networks.

Figures 3.7(a), 3.8(a) also show seasonality corresponding to the UMass Amherst academic calendar; a decrease in work network occupancy and a concomitant increase in MISC network occupancy during vacations; conversely, an increase in house network occupancy and work network occupancy but a decrease in MISC network occupancy during semesters. Not surprisingly, Figure 3.7(a), 3.8(a) also show per-week periodicity for house and work network residence times, with the percentage of time in work networks higher on workdays and less on weekend days, and the percentage of time in house networks higher on weekend days and less during workdays. Figures 3.7(b), 3.8(b) plot the daily total residence time spent in all networks over all users for CS-IMAP, OIT-IMAP, respectively and Figure 3.7(c) plots the daily population of users producing IMAP logs for OIT-IMAP, all showing similar periodic behavior.



(a) Hourly network residence time.



(b) Weekly network residence time.

Figure 3.9: OIT-IMAP. Box plot with whiskers with average and maximum for hourly and weekly network residence time over all users.

We also observe hourly and weekly patterns in the aggregate average and maximum for hourly and weekly network residence times (shown as box plots with whiskers with average and maximum in Figures 3.9(a) and 3.9(b)) over all users in OIT-IMAP. Figure 3.9(a) shows that users tend to be connected approximately 10 minutes on average and up to 35 minutes per hour. Network residence time during daytime is longer than during nighttime, with an increase of residence time in work networks during the day. Figure 3.9(b) shows that users are connected approximately 5 hours a day on average up to 10 hours per day. Network residence time during workdays is longer than during weekend days, with an increase of residence time in work networks during the week. Similar hourly and weekly results are also found in CS-IMAP.

Let us now turn our analysis from the aggregate to the individual, and investigate the fraction of an *individual* user's residence time spent in the single network in which it is most often resident, as well as in the three networks in which together it is most often resident. Figure 3.10 plots the distribution (over all users) of the fraction of time that a user in CS-IMAP spends resident in the network in which it is most often

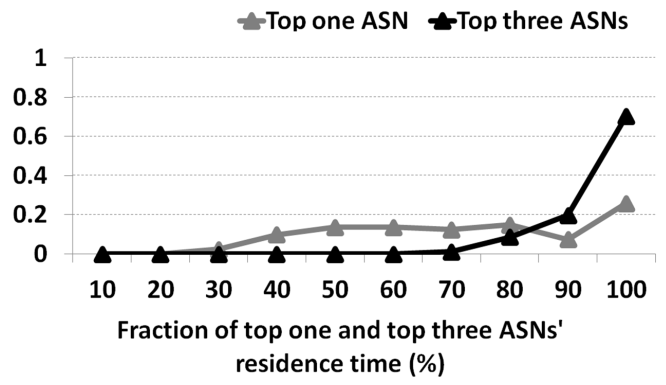


Figure 3.10: CS-IMAP. pdf of the fraction of the (three) longest residency ASNs' residence times to the total residence times.

resident (grey line with triangle points), and in the three networks in which together it is most often resident (black line with triangle points). The black curve indicates, for example, that approximately 75% of the users spend between 90% and 100% of their time in their top three networks, and that nearly 20% of the users spend between 80% and 90% of their time in their top three networks. Thus we see that individual users generally also spend the lion share of their residency time in just a few (e.g., three) networks. A much smaller fraction of the users spend their time in just one network - the gray curve indicates that roughly 25% of the users spend 90% to 100% of their time in their most commonly resident network. Similar results are also found in OIT-IMAP.

3.3.2 User's multi-sessioned behavior

Having considered a user's connectivity to individual networks, let us next examine a user's contemporaneous connection to two or more networks. In our measurements, we observe that 99% of the ASN-based multi-sessioned time windows in OIT-IMAP and 99.5% of the ASN-based multi-sessioned time windows in CS-IMAP consist of only two ASNs.

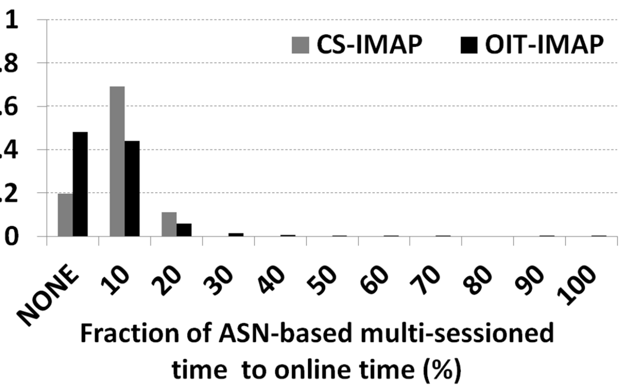


Figure 3.11: pdf of ASN-based multi-session time per user.

Figure 3.11 plots the fraction of users (y-axis) who spend a given fraction of their time (x-axis) connected to multiple networks in CS-IMAP and OIT-IMAP. Figure 3.11's gray bar indicates, for example, that 20% of the users in CS-IMAP were always connected to a single network (when online). Approximately 70% of the users spent less than 10% (but greater than 0%) of their time multi-sessioned and approximately 7% of users were multi-sessioned between 10 and 20% of their time online. Figure 3.11's black bar shows that approximately 50% of the users in OIT-IMAP were always connected to just a single network. Overall, however, we found the amount of multi-sessioned time to be much higher than we would have expected, suggesting that contemporaneous connectivity to multiple networks should not be considered "outlier" behavior.

A deeper investigation in the multi-sessioned time windows revealed three common scenarios, with the following potential causes of multi-sessions:

1. **Fixed and mobile networks.** 55% of multi-sessioned time windows in OIT-IMAP and 51% in CS-IMAP consisted of a fixed (residential or Five colleges) and a mobile network (as defined in Table 3.1's mobile category). *(i)* These scenarios could correspond to the cases of a user carrying multiple devices or a single device with multiple NICs being contemporaneously connected to different networks

(e.g., a laptop connected to a wired network and a smartphone connected to a cellular data network). *(ii)* Network transitions between fixed and mobile networks within a time window could also have resulted from a user's switching his/her devices.

2. **Fixed networks across different ISPs.** 17% of multi-sessioned time-slots in OIT-IMAP and 27% in CS-IMAP consisted of two fixed networks (residential and Five colleges) with little overlap in their physical footprints - the Five colleges network is generally confined to campus locations. *(i)* Contemporaneous access to these two networks in the same time window could have resulted from a user physically moving from one network to another (e.g., office to home or vice versa) or *(ii)* could also have resulted from emails being automatically requested by a user device in a different physical location that the user him/herself, or from VPN access to the Five colleges network via the residential network.

3. **Network transitions within the same ISP.** 6% of multi-sessions in OIT-IMAP and 4% in CS-IMAP show multiple networks access from two ASNs owned by a single service provider such as SAS, Verizon, AT&T and Comcast. This may correspond to the case of a user who is either physically moving and connecting to different 3G/4G or 802.11 base stations while in motion, or a stationary user connecting to different base stations within a time window.

Let us conclude this section by further dissecting the cases above to determine which multi-sessioned time windows might result from a user's transition between networks (e.g., as indicated by a series of IMAP log entries from one network followed by a series of IMAP log entries from another network during a time window) versus a user switching back and forth between networks in that time window. Let $S_{t_1}^{t_2}$ be a sequence of networks to which a user is connected from t_1 to t_2. For instance, if a user at t generates three consecutive IMAP log entries via network B followed by one IMAP log entry via network A, then $S_t^t = \{B, A\}$. We determine whether a user performs a network transition or is contemporaneously connected to multiple networks at multi-sessioned time window t based on the following proposition.

Proposition 4. Given a user's IMAP log entries over three consecutive time-slots from $t-1$ to $t+1$, a user is regarded as performing a network transition at multi-sessioned time-slot t if $S_t^t = S_{t-1}^{t+1}$.

For example, suppose that $S_{t-1}^{t-1} = \{A\}$, $S_t^t = \{A, B\}$, and $S_{t+1}^{t+1} = \{B\}$. Then we derive $S_{t-1}^{t+1} = \{A, B\}$, and thus $S_t^t = S_{t-1}^{t+1}$, implying a network transition during the time window. On the other hand, suppose that $S_{t-1}^{t-1} = \{A\}$, $S_t^t = \{A, B\}$, and $S_{t+1}^{t+1} = \{A\}$. In this case, $S_{t-1}^{t+1} = \{A, B, A\}$, and thus $S_t^t \neq S_{t-1}^{t+1}$, indicating the user does not perform a network transition at t; instead we interpret this as there being one session associated with network A from $t-1$ to $t+1$, contemporaneously existing with another session associated with network B during time window t.

Using Proposition 1, we observed that users performed network transitions in only 12% of multi-sessioned time windows in both OIT-IMAP and CS-IMAP, suggesting that a user is more likely to be using multiple networks contemporaneously during a multi-sessioned time window rather than being the process of transitioning between networks.

3.4 A parsimonious Markov Chain model

In this Section, we develop a parsimonious discrete-time Markov chain model of individual user transitioning among networks. This model can be used to design, analyze and provision protocols and services that support mobility (e.g., Mobile-IP home and foreign agents, or next generation services such as MobilityFirst's GNS [70]). A model of individual user behavior is particularly valuable, as it can be easily used to scale up evaluation workloads. After presenting our model, we validate how well performance measures determined via the aggregation of individual user-level models (in particular, signaling overhead due to user-transitioning between networks) match those determined from the traces.

3.4.1 Markov Chain Model of User-Centric Network Transitioning

We develop a parsimonious discrete-time Markov chain model of individual user network-transitioning. Our unit of discrete time is the time window discussed in Section 3.2. The Markov chain states encode enough state information to compute the cost of a user's signaling at each time-step.

- Let X_t be the number of *new* networks to which a user is attached at time t, with respect to time *t-1*. The first dimension of the Markov chain tracks the value of X_t, which will be used to quantitatively compute signaling overhead induced as a user transitions among networks, as we will discuss below.
- Let Y_t be the number of networks to which a user is attached at time t. The second dimension of the Markov chain tracks the value of Y_t, which will be used to quantitatively compute signaling overhead induced when a user detaches from a network, as we will discuss below.

X_t and Y_t may take value $\{0, 1, *\}$, where $*$ denotes two or more networks contemporaneously connected at t; for simplicity, we do not distinguish the case of more than two contemporaneous sessions from the case of exactly two such sessions, since approximately 99% of multi-sessioned time windows consist of only two network domains in our traces, as discussed in Section 3.3. Our model can be easily extended to cover the more general case. Our Markov model thus consists of six states,

$$\{(0,0), (0,1), (1,1), (0,*), (1,*), (*,*)\}.$$

The model has a stochastic transition probability matrix $P = [p_{ij}]$ where $p_{ij} = Pr\{(X_t, Y_t) = j | (X_{t-1}, Y_{t-1}) = i\}$ and $\sum_j p_{ij} = 1$. These transition probabilities will be determined empirically from our traces.

The overall signaling cost from the user to a network-wide mobility management service (e.g., a Mobile-IP home agent, or the MobilityFirst GNS) on a state transition at $t-1$ to t, denoted by CO_t, is computed as follows. Let $\mathcal{A}$ be the signaling cost generated when a user joins a new network, and let $\mathcal{D}$ be the signaling cost generated when a user departs from a network. (For simplicity, we will not consider signaling costs in the reverse direction from the management service to the user, although these can be easily included in the model.)

- **Explicit detach**. In the case that network detachment is explicitly signaled, CO_t is computed by

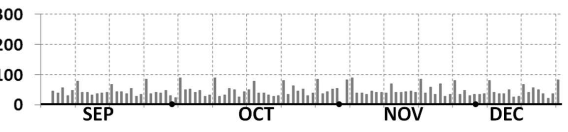
$$CO_t \;=\; \mathcal{A} \cdot X_t + \mathcal{D} \cdot (Y_{t-1} - (Y_t - X_t))$$

- **Implicit detach.** In the case that network detachment is implicitly signaled by attachment to a new network, CO_t is computed by

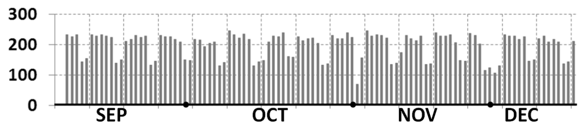
$$CO_t \;=\; \mathcal{A} \cdot X_t.$$

3.4.2 Trace properties

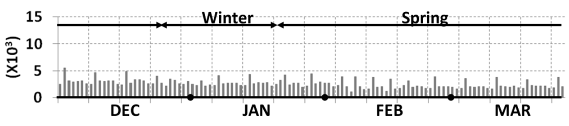
We investigate the properties of our CS-IMAP and OIT-IMAP traces. We first extract subtraces from the CS-IMAP and the OIT-IMAP traces and bisect each subtrace into the *training phase (also called phase 1)* and the *validation phase (also called phase 2),* which will be used in model parameter estimation and model validation, respectively.



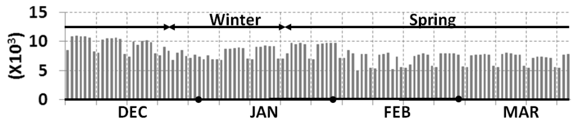
(a) CS-Fall. X_t.



(b) CS-Fall. Y_t.



(c) OIT-IMAP. X_t.



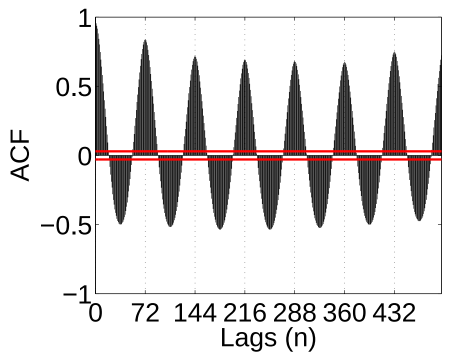
(d) OIT-IMAP. Y_t.

Figure 3.12: Time series plot of "daily" aggregate cost of X, Y over all users (using IP prefix distinction).

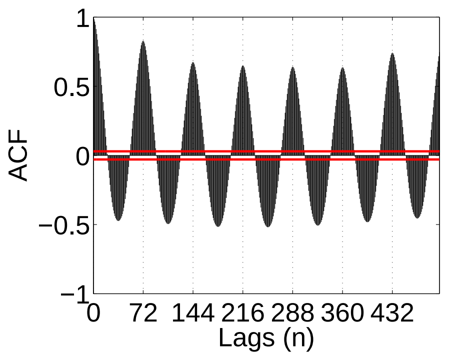
- **CS-Fall subtrace.** Figures 3.12(a), (b) show the time series plots of daily aggregate values of X_t and Y_t for 79 users during the Fall 2013 semester (using IP prefix distinction). The CS-Fall subtrace's *training phase* and *validation phase* consist of data from September 3rd to October 25th and from October 26th to December 16th, respectively.

- **OIT-Spring subtrace.** Figures 3.12(c), (d) show the time series plots of daily aggregate values of X_t and Y_t over 7,137 users in OIT-IMAP (using IP prefix distinction). Unlike the CS-Fall subtrace, Figure 3.12(d) shows a downward drift, particularly during the first half of the trace, likely resulting from the change in user population previously observed in Figure 3.7(c). Since our goal is to model the system in steady state, we thus only consider the subtrace during February and March for modeling, with the *training phase* and *validation phase* consisting of data from February and March, respectively. This subtrace has 5,793 users generating IMAP logs.

For each subtrace, we derive one set of aggregate values of X_t over all users, and another set of aggregate values of Y_t over all users (using IP prefix distinction), sampled at 15 minutes (for CS-Fall) or at 20 minutes (for OIT-Spring).



(a) X_t

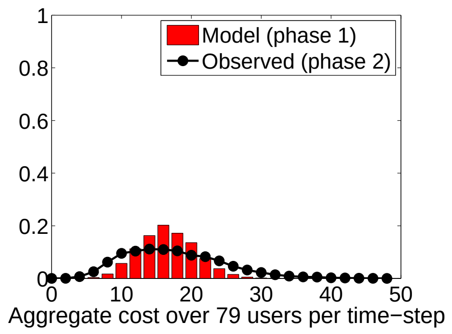


(b) Y_t

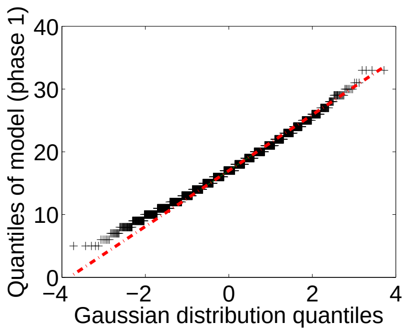
Figure 3.13: Autocorrelation function for X_t and Y_t at different time lags (n), OIT-Spring data.

Patterns of ACFs. The sample autocorrelation function (ACF) measures the degree of correlation between data at varying time lags (denoted by n), detects any trends and periodicity in a data series, and is also used to check the randomness of data. If random, the autocorrelation should be near zero for any and all time-lag separations. Figure 3.13 plots the ACFs of values of X_t and Y_t for the OIT-Spring subtrace. Figures 3.13(a) and (b) demonstrate that X_t and Y_t in the OIT-Spring subtrace have daily ($n = 72$) and weekly ($n = 504$) periodicity, and drop to near zero correlation at lag 20 so that X_t and Y_t are considered independent at around every seven hours (20·20 minutes). Similar periodicity and seven-hour independence results were also encountered in CS-Fall trace, but with lower amplitudes.

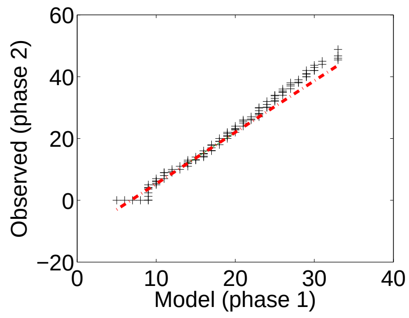
Testing for Stationarity. We check the subtraces themselves for stationarity using the *KPSS test* [46]. The KPSS assesses the null hypothesis that data is stationary over a range of time lags. The tests at the 1% significance level suggest that X_t and Y_t data in OIT-Spring are stationary for $n > 0$, but X_t data in CS-Fall is stationary for $n > 1$ and Y_t data in CS-Fall is stationary for $n > 4$.



(a) **pdf with bin size = 2.**



(b) **Gaussian distribution vs. Model**



(c) **Model vs. Observed**

Figure 3.14: CS-Fall. Aggregate cost over all users.

3.4.3 Model estimation and validation procedure

We use the observed relative transition rates during the *training phase* to estimate the transition probabilities of our Markov chain model. To determine how well our Markov chain model predicts user behavior we will compare signaling costs determined by the model with those found in data from the *validation phase*. We proceed as follows:

1. **Transition probabilities for the Markov Chain Model.** Using the *training phase* data, we derive the transition probabilities for our Markov Chain model of a canonical user by counting the number of times that U users move from state i to state j per time-step and then normalize these counts so that the sum of the transition counts out of each state equals 1. This gives us our empirical transition probability matrix, $\hat{P} = [\hat{P}_{ij}]$.

2. **Generating a sequence of synthetic transitions between states for a population of U users.** For each of the U users, we start from state (0, 0) and generate a next state using the transition probabilities $\hat{P}$. We repeat this process for ϕ time-steps (5,000) and then generate a sequence of length ϕ of state transitions made by the U users.

3. **Determining the signaling cost for U users.** For each time-step, we compute the aggregate signaling cost of the U users, using CO_t as in the previous subsection; for simplicity, we assume that users explicitly signal network detachment, with $A = D = 1$. Then we compute the distribution of signaling cost for the U users.

4. **Model validation.** Once the baseline distribution is built, we test how well our model predicts the number of signaling messages generated per time-step for the U users. To validate our model, we compare the model-predicted values (whose state transition probabilities were derived from *training phase* data) with the empirical distribution found in *validation phase.*

3.4.4 Prediction with aggregate user population

CS-Fall. Figure 3.14(a) plots the pdf of the model-predicted and the observed aggregate cost over all users for the CS-Fall data set. Figures 3.14(b), (c) show the Q-Q plot of the randomly generated, independent standard normal data ($\mathcal{N}(0,1)$) on x-axis versus the model cost data on y-axis, and the Q-Q plot of the model cost data on x-axis versus the observed cost data on y-axis, respectively; a data point (x,y) on the Q-Q plot corresponds to one of the quantiles of the distribution plotted on the y-axis against the same quantile of the distribution on the x-axis; the plot has a red reference line through the origin with slope 1; points denoted as + should lie roughly on this line if the x-axis and y-axis data come from the same distribution. The linearity evidenced in Figure 3.14(b) suggests that the data follows a Gaussian distribution with slightly positive skew. Figures 3.14(a), (c) confirm that the model cost and the observed cost datasets come from a Gaussian distribution and the model fits the observed data well, passing the chi-square goodness of fit test with 5% significance level.

Recall that our model for U users aggregates the results from U independent user-level models. Since the ACFs of empirical values of X_t and Y_t show both positive and negative correlation at different time lags in Figure 3.13, it is not surprising that signaling costs match the least well at the lower and upper extremes of the distributions in Figures 3.14(a), (c). If the tail distribution of cost is of interest (e.g., for provisioning system resources at the 95% workload maximum), interesting future work would be to develop a model that more accurately matches this tail behavior.

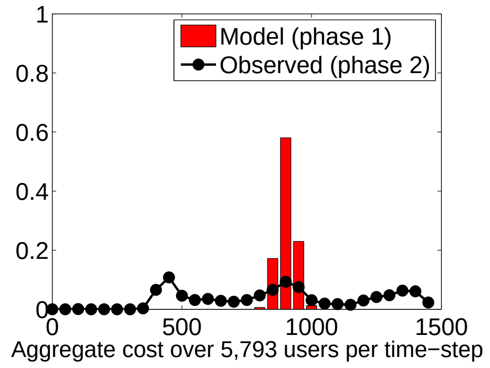


Figure 3.15: OIT-Spring. pdf of aggregate cost over 5,793 users.

OIT-Spring. Figure 3.15 plots the pdf of the model-predicted and the observed aggregate cost over all users for the OIT-Spring data set. Figure 3.15 shows that the Gaussian distribution of cost predicted by the aggregation of individual user models does not fit the observed multi-modal data, which shows three distinct peaks. Visually, Figure 3.15 suggests that costs might better be modeled as a *mixture* of Gaussian distributions. But what might each component of the mixture correspond to, and how many distributions should be mixed? To answer this question, we performed a clustering analysis.

3.4.5 Prediction with user clusters

Since a user's affiliation is not known in our OIT-IMAP traces, we partitioned the 5,793 users in OIT-Spring subtrace into K clusters based on their signaling cost, using Expectation Maximization (EM) clustering [12]. Let u_i be the average daily signaling cost during the OIT-Spring's *training phase* for user i, and let z_i be the latent variable for the user cluster assignment for user i. We assume that u_i follows a mixture of K Gaussian distributions, i.e., $u_i|(z_i = k) \sim \mathcal{N}(u_i|\mu_k, \sigma_k)$, with mixture weight $\tau_k = Pr[z_i = k]$ subject to $\sum_{k=1}^{K} \tau_k = 1$. EM clustering iteratively estimates $\theta = (\tau, \mu_1, \cdots, \mu_K, \sigma_1, \cdots, \sigma_K)$ while maximizing the following likelihood function until there is convergence of θ.

$$\begin{aligned} L(\theta|u, z) &= Pr[u, z|\theta] \\ &= \prod_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}(z_i = k) \cdot \tau_k \cdot \mathcal{N}(u_i|\mu_k, \sigma_k), \end{aligned}$$

where $\mathbb{1}$ is an indicator function.

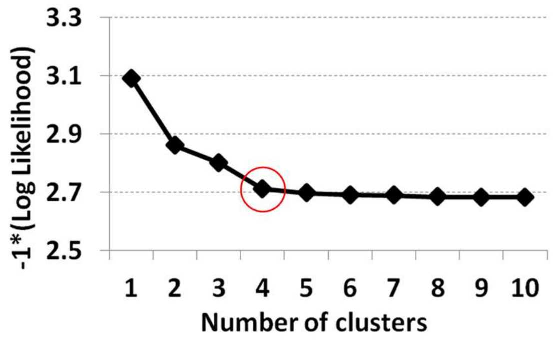
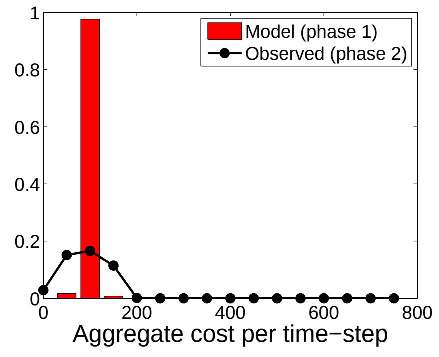


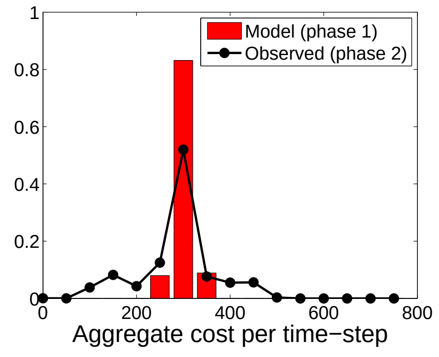
Figure 3.16: OIT-Spring. Log likelihood of cross-validation data for different numbers of clusters.

We use WEKA's EM clustering implementation [9, 75] which determines the best number of clusters using 10-fold cross-validation[4]. Figure 3.16 shows the negative log likelihood of the cross-validation data as a function of the number of clusters; the curve quickly decreases up to four clusters and then flattens out, suggesting that four clusters be used.

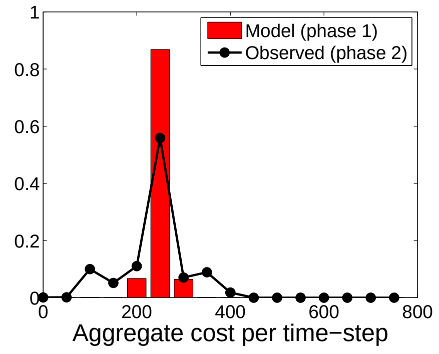
[4]In the 10-fold cross validation, the data is partitioned into ten folds. Each of the folds is then set aside at turn as a test set, a clustering model computed on the other nine training sets, and the value of the log likelihood calculated for the test set. These ten values are averaged for each alternative number of clusters. WEKA's EM algorithm iterates until the change in log likelihood falls below 10^{-6} or 100 iterations have elapsed by default.



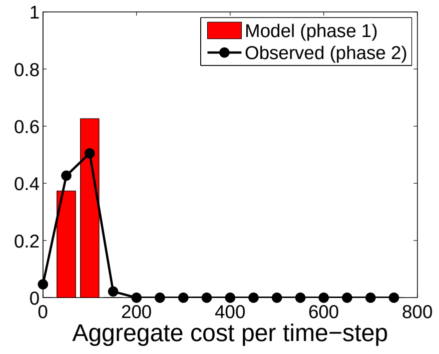
(a) light-user cluster



(b) mid#1-user cluster



(c) mid#2-user cluster



(d) heavy-user cluster

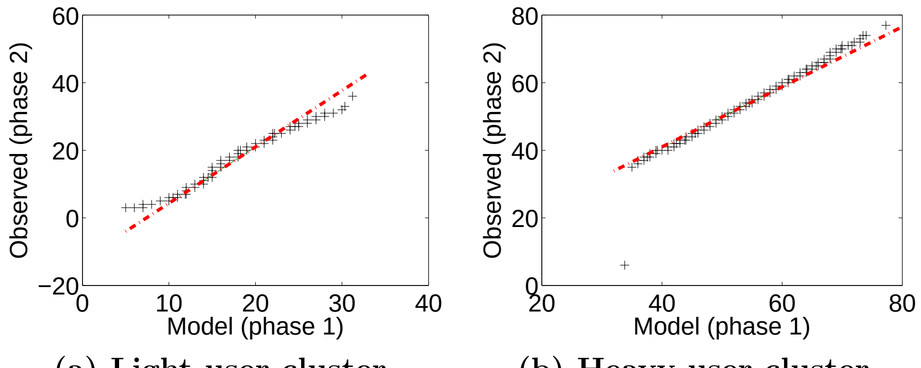
Figure 3.17: OIT-Spring. pdfs of aggregate cost over cluster users.

	# users	mean	std. dev.
Light-user cluster	870 (15%)	0.25	0.19
Mid#1-user cluster	2,274 (39%)	2.30	1.16
Mid#2-user cluster	1,928 (33%)	6.57	2.65
Heavy-user cluster	721 (12%)	13.62	6.23

Table 3.2: OIT-Spring. Four clusters resulting from the EM clustering.

Table 3.2 shows the resulting four clusters, labeled as light-user, mid#1-user, mid#2-user, and heavy-user clusters, according to the mean of u_i values of each cluster's constituent users. The second column shows the number (and the percentage) of users belonging to each cluster. The third and the fourth columns show the mean and the standard deviation of values of u_i in each cluster.

Figure 3.17 plots the pdf of the model-predicted and the observed aggregate cost over the users belonging to each cluster. Figure 3.17 shows that the cost distributions for the four-cluster model, with clustering based on signaling cost, are closer to their empirically observed distributions when compared with the single cluster (i.e., non-clustered) case. However, even the clustered models do not pass the chi-square goodness-of-fit test.



(a) Light-user cluster. **(b) Heavy-user cluster.**

Figure 3.18: OIT-Spring. Q-Q plots for aggregate cost over manually picked 100 users.

We thus next handpicked the light-user cluster to consist of the 100 users having the least signaling cost (a mean cost of 0.06) and a heavy-user cluster consisting of 100 users having the highest signaling cost (a mean cost of 41) in OIT-Spring's *training phase*. Figure 3.18 shows the Q-Q plots for aggregate costs for the light-user cluster and heavy-user cluster, and show a good fit, passing the chi-square goodness of fit test with the 5% significance level. These results suggest that proper clustering can improve model performance in predicting signaling costs, a topic we plan to pursue in future research.

state	(0, 0)	(0,1)	(0, *)	(1, *)	(*, *)
	86%, 87%	7%, 7%	6%, 6%	0%, 0%	0%, 0%

(a) Aggregate user population.

state	Light	Mid#1	Mid#2	Heavy
$(0, 0)$	85%, 87%	85%, 86%	85%, 87%	85%, 86%
$(0, 1)$	7%, 7%	8%, 7%	7%, 7%	8%, 7%
$(1, 1)$	6%, 6%	6%, 6%	6%, 6%	6%, 6%
$(0, *)$	0%, 0%	0%, 0%	0%, 0%	0%, 0%
$(1, *)$	0%, 0%	0%, 0%	0%, 0%	0%, 0%
$(*, *)$	0%, 0%	0%, 0%	0%, 0%	0%, 0%

(a) Clustered users.

Table 3.3: OIT-Spring. Model-based and empirically observed state occupancies.

Table 3.3 compares model-based and empirically-observed state occupancies of OIT-Spring, showing good agreement for both the aggregate population of users and for clustered users. Each entry of the table denotes the model-predicted value and the observed value. For example, as shown in Table 3.3(a), the model predicts that a user is offline (i.e., state (0, 0)) 86% of the time, while we empirically observe that a user is offline 87% of the time.

3.5 Related Work

Numerous studies have characterized physical human movement using empirical datasets and discussed the impact of physical user mobility patterns on network performance and design. Human mobility traces have been collected from diverse access networks such as WLAN [18, 34, 42], Bluetooth networks [18], and cellular networks [31, 37, 55]. Research using Wi-Fi access datasets has been done in a single, physically-scoped network domain, such as a campus or enterprise, thus focusing on user mobility within that limited physical domain. In this sense, cellular network data might more fully model human mobility (since users typically carry their cellular phones); such cellular data, however, is typically proprietary. But individual WiFi and cellular traces by definition only include data from an individual type of network, and have not considered contemporaneous residence within multiple networks nor transitions among networks. More generally, we believe there is an important distinction to be made between physical mobility and mobility among networks, as discussed in Section 3.1; our work is the first to characterize and model mobility among networks (which we have referred to as network transitioning).

[20, 31, 55] have related human mobility patterns to network resource use in Wi-Fi access points or cellular network base stations. [31, 55] have found that the extent of users' physical mobility is low and concentrated among a small number base stations, with infrequent visits to other base stations in that network. Those conclusions, however, are based on physical mobility within a single network.

3.6 Conclusion

In this Chapter, we performed a measurement study of user transitioning among networks and discussed insights and implications from the measurements. Our measurement study, conducted using two sets of IMAP server logs of populations of approximately 80 users and more than 7,000 users, characterized user network transitioning in terms of transition rates, network residency time, and degree of contemporaneously resident network domains. Based on these measurements, we also developed and validated a parsimonious discrete time Markov chain model of canonical user transitioning among networks. Our measurements and models provide quantitative insight into the location management signaling overhead needed by modern and proposed name/address translation and location management protocols; our models provide the ability to design, dimension and analyze such systems.

CHAPTER 4
GROUP MOBILITY INDIRECTION

4.1 Introduction

Group mobility - a group of users whose mobility among networks may be correlated - presents opportunities for efficiently handling the user-location information (e.g., the access network with which a given user currently resides) associated with the mobile users in that group. Such group mobility occurs when users travel together (e.g., in a vehicle), when users are engaged in social relationships (e.g., affiliation), or when users are regularly (and perhaps periodically) associated with a small set of networks. Intuitively, we expect that there may be significant savings in the workload associated with location-tracking protocols and name/location translation in location-independent architectures (e.g., GNS in MobilityFirst [69], home/foreign agents in IP networks [39], HLRs/VLRs in cellular networks [10], and SIP registrars [58]) if the network control plane updates information associated with the group as a single entity, rather than for each of the users individually.

In this Chapter, we introduce the notion of "group-mobility indirection" in which a single group identifier, registered in the name/location translation service, references a group of users, and updating the network location of the single group identifier, rather than individually updating the location of all individual users, as the group's users moves. Our indirection technique differs from prior work (e.g., [16, 23]) that employed hierarchical techniques to define a static and rigid subnet structure associated with wireless users. Leveraging a logically centralized but globally distributed name/location resolution service [69], our approach provides for topology-independent group membership and handles diverse types of group mobility occurring at different levels of network granularity, such as mobility among access networks or among APs and BSs associated with different ISPs.

A central challenge for group-mobility indirection is to determine which (of the perhaps multiple) network locations to associate with the group, when users associated with a group identifier may be split among several networks. We design a group-mobility indirection architecture that separately tracks the group location associated with only one of the networks in which group members are resident, and then individually tracks the locations of users whose network location differs from that single group location. We propose two algorithms to determine the network location associated with the group location: *(i)* an event-based algorithm that elects a leader among group members and reactively associates the group location with the location of that leader as that leader moves, and *(ii)* a periodicity-based algorithm that periodically updates the group location with a predicted network location at a predicted time. We evaluate the performance of these algorithms by quantifying the reduction in location-tracking overhead (over the case of no group indirection) via a synthetic group-mobility model and via empirical traces of approximately 4000 mobile users. We discuss the relation of the best performing algorithm to the periodicity or predictability of user movements observed in these traces. Last, we cluster a set of approximately 4000 users in our empirical traces into multiple groups, each with a separate group identifier, and investigate performance as the number of groups increases. We find that the indirection technique decreases up to approximately 50% of location-tracking overhead by associating the 4000 users as a single group identifier, and decreases up to approximately 70% by dividing the 4000 users into only five clusters.

The remainder of this Chapter is structured as follows. In Section 4.2, we introduce the notion of group-mobility indirection and discuss its challenges. In Section 4.3, we describe our group-mobility indirection design. In Section 4.4, we describe the above two algorithms. In Section 4.5, we describe a generative group-mobility model and evaluate the performance of our group-mobility approaches using synthetic traces generated via this model. In Section 4.6, we empirically investigate the reduction in performance using real-world traces containing periodicity, and discuss the reduction in performance as the number of groups resulting from clustering increases. We then conclude this Chapter.

4.2 Group Mobility Indirection

In this Section, we begin by briefly describing the MobilityFirst architecture [69] in order to set the context for, and motivate the need for, group-mobility indirection. However, the concept of group-mobility indirection is more broadly applicable in current and future Internet architectures beyond MobilityFirst. Then, we introduce group-mobility indirection and discuss the challenges associated with individual nodes splitting from and rejoining the larger group.

4.2.1 The MobilityFirst architecture

The MobilityFirst architecture uses a globally unique ID (GUID) that names individual users, application identifiers, endpoints, content, and more. Similar to home/foreign agents in IP networks, and HLRs/VLRs in cellular networks, MobilityFirst's global name service (GNS) – a logically centralized but geographically distributed name and location translation – tracks a GUID's current network location(s), i.e., the access network(s) in which the GUID resides, and resolves the network location(s) of the GUID for a sender's location query. The GNS also allows a GUID to be associated with more than one network location, and can return one, some or all of those multiple locations associated with the GUID for a sender's location query on the GUID.

In order to track a GUID's network location(s), the location-tracking signaling between the GNS and a gateway router at the access network (or a mobility management agent/service for that access network), and between that gateway router[1] and a GUID happens as follows. A GUID joining or leaving an access network sends an attachment/detachment request to the gateway router in that access network. On the receipt of an attachment/detachment request, a gateway router transmits a **location-update message** to the GNS indicating the identity of the GUID that is attaching/detaching, as well as the identity of network involved. In the case of a detachment, the gateway router might not explicitly signal a location-update message to the GNS, depending on whether soft-state timeout of a GUID's residency information may occur. Then, this gateway-to-GNS signaling causes the GNS to update the GUID's location.

4.2.2 Group-mobility indirection

Group-mobility indirection is used to track the locations of a *group* of (mobile) GUIDs whose mobility among networks may be correlated. It does so by using a *single* identifier – the AGUID[2] – to represent the group's default location, updating the location of the single AGUID (i.e., the access network with which the AGUID is associated), rather than individually updating the location of all individual GUIDs, as members of the group move. Intuitively, there can be significant savings in signaling overhead if the individual GUIDs often move together as a group, and only the location of the single AGUID needs to be updated. We will consider two forms of mobility: *(i)* **network-aperiodic group mobility** in which a group of users have correlated

[1]This functionality could be implemented in a gateway router, or in mobility management service for that access network, as in the HLR/VLR in cellular networks or the home/foreign agent in Mobile IP. In the following we will use the term ''gateway router."

[2]The AGUID originates from an "affinity group" that represents group mobility in the context of the MobilityFirst architecture.

(but not necessarily identical) mobility but a sequence of associated networks does not contain periodicity in time, and *(ii)* **network-periodic group mobility** in which there is a sequence of networks associated with a group of users periodically or at the same times of a day.

Group-mobility indirection is conceptually similar to existing grouping approaches, such as multicast, in several respects. Group-mobility indirection references a group of GUIDs using a single group identifier, similar to a multicast IP address. Similar to IGMP's multicast router [25], a gateway router in group-mobility indirection is responsible for transmitting the connectivity of group members to the GNS on behalf of group members. However, group-mobility indirection has several fundamental differences from existing grouping.

(i) **The purpose of group-mobility indirection is to reduce the location tracking and name/location translation workload in the control plane**, including signaling overhead and name/location database updates. The main goal of many existing grouping approaches in wired (e.g., multicast, geocast) and mobile/wireless networks (e.g., MobiCast [65]) is to reduce communication overhead in the data plane when a group of users receive the same data; existing approaches have thus focused on building a communication-efficient data-forwarding tree for communication among the group of users. In contrast, the motivation for group-mobility indirection is to reduce the location management and name/location translation workload in the control plane. As we will see, these savings can be realized in cases where, without group-mobility indirection, numerous mobile users would generate a large burst of location-update messages in a short interval of time (and consequently multiple updates to the GNS) when they move to a new network. *Group-mobility indirection replaces such a burst of location-update messages for individual mobile users with a single update message, and single GNS update, for the single AGUID.*

(ii) **The challenge of group splitting.** The major complexities and challenges of group-mobility indirection occur when group nodes "split" from the group, resulting in different group members being located in different access networks at a given point in time. In this case, the GNS might maintain multiple network locations for an AGUID, or might associate the AGUID with just one network and track the location of group members not in this single (default) network separately. Additionally, group members may move from/to networks in a similar, correlated manner, but may not all do so at exactly the same time.

(iii) **Overhead in the data plane.** In addition to differences in control-plane signaling, there are important differences between multicast grouping and group-mobility indirection in the data plane as well. In a multicast session, a sender's goal is to reach all members in the multicast group, so multicast must necessarily and intentionally track the location of the multiple networks containing group members. However, associating an AGUID with multiple networks incurs a data-plane cost – a sender in a unicast session with one of an AGUID's group members wants to reach only one group member; delivering this sender's data to multiple networks associated with an AGUID results in unneeded transmission of data to networks other than the one in which the receiving group member is located.

4.3 Group-mobility-indirection design

In this Section, we describe the design for group-mobility indirection in the MobilityFirst architecture. At a high level, the GNS is designed to operate as a logically centralized controller that initiates group membership, and then interacts via gateway router(s) in access networks to manage location information of GUIDs in an access network. A broad goal is to maintain location information as precisely as in the case of no group indirection, where each GUID individually signals its location information changes and incurs the consequent increase in signaling overhead discussed above.

4.3.1 AGUID-membership establishment

We assume that AGUID membership – a pairing of an AGUID and its constituent GUIDs – is specified *a priori* based on a known shared context or is organized by cluster analysis using historical GUID network-association data (as we will discuss in Section 4.6). Once AGUID membership is initialized in the GNS, the GNS must additionally announce membership to either *(i)* gateway router(s) or *(ii)* the individual GUIDs associated with the AGUID. This enables a gateway router to determine that a given AGUID is associated with the GUID of a mobile user potentially in the access network managed by that gateway router.

4.3.2 Name/location resolution semantics at the GNS database

In this subsection, we describe how the GNS database manages an AGUID's network location(s), and resolves a query regarding the location(s) of the constituent GUIDs[3].

Given an AGUID and its constituent GUIDs, the GNS database maintains a mapping of a GUID to its associated AGUID and the locations associated with that AGUID and selected GUIDs as follows. *(i)* The GNS separately tracks the AGUID's (perhaps multiple) locations. In our work, we assume the GNS tracks **exactly one** out of the (perhaps multiple) locations for each AGUID (as we will discuss below); we call this network the **default network**. The default network may change over time as the GUIDs associated with that AGUID move from one network to another. *(ii)* The GNS may also maintain the individual location(s) of a GUID. If the GUID makes a move independent of the AGUID's default network, or if the GUID is simultaneously

[3]An AGUID is oblivious to senders, thus a sender which intends to transmit data to the GUID(s) associated with an AGUID cannot directly make a query on the AGUID's location(s).

associated with other network location(s) as well as with the AGUID's default network, then the GNS separately keeps the location(s) – different from the default network – as the GUID's individual location(s). The GNS keeps the GUID's individual location as **NULL** when the GUID is associated only with the AGUID's default network. We will also assume in our discussion below that the GUID explicitly signals that it is detaching from a network; a soft-state approach in which state times out when not refreshed could be implemented, but introduces additional complexity [60].

We maintain only a single default network for an AGUID for data-plane efficiency. If an AGUID were to be associated with multiple networks, then data destined for a GUID would be delivered to each of these (multiple) networks, resulting in the delivery of unneeded data to the networks in which the GUID was not resident. If GNS update signaling and database update costs are of particular concern (with data-plane costs being of lesser concern) then it might be preferable to associate multiple locations with an AGUID to minimize signaling and update costs. A precise quantification of these tradeoffs remains as future research.

The GNS resolves a query regarding the location(s) of the GUID associated with an AGUID as follows. If the GUID is associated with only one network at a time, the GNS employs a *most-specific-lookup* policy (similar in spirit to longest-prefix matching) in which the GUID's individual location has priority over its associated AGUID's default network. The GNS first looks up the GUID's individual location(s) and returns it as the GUID's location. Otherwise, if the GUID's individual location is NULL, the GNS returns the AGUID's default network as the GUID's location. On the other hand, if the GUID is simultaneously associated with more than one locations, the GNS intelligently returns the GUID's individual location(s), the AGUID's default network, or both.

4.3.3 Location-tracking signaling

In this subsection, we describe how the GNS updates the default network associated with an AGUID and the individual location(s) of the GUIDs associated with the AGUID, as the GUIDs move from one network to another.

Using the algorithms discussed in Section 4.4, the GNS chooses one out of (perhaps multiple) locations – where the GUIDs are resident or will be likely to be resident – to be associated with the AGUID's default network. When an AGUID's default network is associated with a new network, the GNS informs both the old default network and the new default network of this change using **ACK/NAK**, as we describe below. *Note that the gateway router in the network associated with the AGUID's default network reduces location-tracking signaling overhead by not sending location-update message(s) for each of the invidual GUID(s) associated with the AGUID.*

ACK. When an AGUID's default network changes, the GNS sends an ACK message to this new default network, indicating that the GUIDs associated with that AGUID and whose individual location state in the GNS indicates that they are in the default network are now implicitly recognized as being resident in that new default network. After the ACK, the gateway router in the new default network need **not** transmit location-update message(s) for GUID(s) associated with that AGUID that then later connect to this default network. Nor does the gateway router need to signal any information regarding GUIDs already resident in the default network, since changes in those GUIDs state can be done in the GNS itself. Thus no per-GUID signaling is needed when an AGUID associates with a new network

NAK. When an AGUID's default network changes, the GNS sends a NAK message to the old default network, indicating that it is no longer the default network for that AGUID. This means that any future arrivals of GUIDs associated with that AGUID must be explicitly signaled by the gateway router in the old default network.

Delayed NAK. It is possible to further minimize signaling with the use of delayed NAKs in the case the GUIDs associated with the same AGUID move from one network to another closely in time. Consider the scenario in which the AGUID has moved to another network, but the old default network has yet to be informed of this change (due to delayed NAK generation). Thinking that it is still the default network, this old default network will suppress sending any signaling messages for other GUIDs associated with that AGUID that leave the old network. (Note that those GUIDs will have their individual locations set when they enter a non-default network. In the case they enter the default network, their location remains NULL (i.e., that they are in the default network) with no signaling required).

rLIST. Upon the receipt of a NAK, the gateway router in the old default network sends the GNS a remaining list (**rLIST**) message containing a list of the GUID(s) who are still connected to the old default network after the change of the default network. The rLIST message thus allows the GNS to mark the old default network as the individual location(s) of the GUID(s) contained in the rLIST.

4.4 Determining the network associated with an AGUID

In this Section, we propose event-based and periodicity-based algorithms to determine the network associated with an AGUID's default network. In both algorithms, we assume that time is evenly slotted into consecutive time windows and thus the GNS attempts to decide an AGUID's default network at each time window.

4.4.1 Event-based algorithm for determining an AGUID's default network

In the event-based algorithm, at the start of a time window, the GNS elects a leader among the GUIDs associated with an AGUID. The GNS follows the location of the leader to determine the AGUID's default network, when the leader signals a move to a new network for the given time window. The event-based algorithm is reactive (as opposed to the periodicity-based algorithm below), since the default-network update is triggered by the location-update message indicating that the GUID elected as a leader moves to a new network. Clearly, the amount of reduced location-tracking signaling overhead will depend on how a leader is elected. We consider the following three leader-election strategies for electing a leader among the GUIDs associated with an AGUID:

First leader. In this strategy, the first GUID that connects to a new network at a given time window is chosen as a leader.

Random leader. In this strategy, a GUID is randomly chosen as a leader.

Majority leader. This strategy is based on the insight that a GUID that belonged to the subgroup containing the largest number of GUIDs resident in one network in the previous time window is more likely to belong to the largest subgroup at the next time window. Thus, at the start of a time window, one of the GUIDs who belonged to a largest subgroup in a previous time window is (randomly) chosen as a leader.

4.4.2 Periodicity-based algorithm for determining an AGUID's default network

Given a series of historical network-association data for the GUIDs associated with an AGUID, the periodicity-based algorithm seeks to exploit periodicity observed in this data to determine the default network. We begin our discussion of this algorithm with the following definitions:

Definition 4 (Period)**.** Let $X^i = \{X^i_t\}_{t=1}^T$, where X^i_t is the network location which GUID$_i$ is most frequently associated with during time window t. Sequence X^i is **periodic** if there exists $\tau \in \mathbb{Z}$ such that $X^i_{t+\tau} = X^i_t$. Then, the value of τ is a **period** of that sequence.

Definition 5 (Periodic sequence)**.** Given period τ, an AGUID's **periodic sequence** is defined as $Z = \{Z_t\}_{t=1}^{\tau}$ where Z_t is the network with which the AGUID's constituent GUIDs are most associated for all $t, (t+\tau), \cdots, (t + \lfloor \frac{T-t}{\tau} \rfloor \cdot \tau)$ time windows.

The periodicity-based algorithm selects the AGUID's default network at the start of a time window according to the identified periodic sequence. The periodicity-based algorithm is proactive, since the default-network update is performed without explicitly receiving a location-update message indicating that the GUID connects to a new network.

4.4.3 Optimal algorithm

Assuming that the GUIDs (associated with an AGUID) movements for next T time windows are known, the optimal algorithm performs a brute-force search that identifies a locally optimal sequence of networks associated with the AGUID's default network for next T time windows that produces the minimum amount of location-tracking signaling overhead. Trivially, the optimal algorithm returns an amount of location-tracking signaling overhead closer to a global optimum as T grows. The optimal algorithm, while impossible to implement in practice, provides an upper bound on the reduction in location-tracking signaling overhead for any feasible algorithm.

4.5 Evaluation of Event-based Group Indirection: Synthetic Model

In this Section, we build a synthetic group-mobility model in which each of the GUIDs associated with a single AGUID moves as a group or independently in discrete time. We describe how the overall amount of signaling overhead for the mobile GUIDs is quantified and then evaluate the performance of our group-mobility approaches by quantifying signaling overhead reduction using the synthetic group mobility model.

4.5.1 Synthetic group-mobility model

In our synthetic discrete-time group-mobility model, n GUIDs associated with a single AGUID split from and rejoin the AGUID at each time window, while moving among m networks. For simplicity, we assume that a GUID remains online at all times and is associated with just one network at a time; thus a GUID moving into a new network may simultaneously incurs two location-update messages: indicating its connection with a new network and its disconnect from an old network, respectively.

The mobility of GUID$_i$ $[i = 1, \cdots, n]$ is modeled using a two-state Markov chain; the Markov chains of the n GUIDs are independent of each other. In the `split` mode state, a GUID moves independent of other GUIDs; in the `group` mode state, a GUID moves together with the AGUID. α_i is the probability that GUID$_i$, given it is in a `split` mode in a time window, remains in a `split` mode at the next time window. β_i is the probability that GUID$_i$, given it is in a `group` mode in a time window, remains in a `group` mode at the next time window. Define $\alpha = E[\alpha_i]$, and $\beta = E[\beta_i]$. Then, we define the AGUID's **group coherence** by

$$\mathcal{C} = \frac{\beta}{\alpha + \beta}.$$

$\mathcal{C}$ is the average fraction of GUIDs in a `group` mode and thus characterizes the degree of location correlation of the AGUID's constituent GUIDs on average.

In our model, after the n GUIDs determine their modes, each of the GUIDs in a `split` mode and the AGUID (whose mobility will affect the rest of the GUIDs, who are in a `group` mode) determine their network locations for the next time window as follows. First, each of the GUIDs in a `split` mode and the AGUID separately decide whether to stay in their current network or move: with probability γ, a GUID or an AGUID stays in its current network. Alternatively, if a GUID in a `split` mode (or the AGUID) moves to a new network, GUID i chooses the next location out of m networks based on individual-transition matrix $\mathcal{S}^i = [\mathcal{S}^i_{ab}]$ (or group-transition matrix $\mathcal{G} = [\mathcal{G}_{ab}]$). Here, element $\mathcal{S}^i_{ab}$ denotes the transition probability that GUID i moves from network N_a to network N_b (subject to $\sum_b \mathcal{S}^i_{ab} = 1$). $\mathcal{G}_{ab}$ is the transition probability that the GUIDs in a `group` mode transition together from N_a to N_b (subject to $\sum_b \mathcal{G}_{ab} = 1$). Our model can be easily extended to accommodate multiple AGUIDs by having multiple group-transition matrices.

4.5.2 Quantifying the amount of signaling overhead

```
function 1: Amount of signaling overhead
input  : Series of $n$ GUIDs mobility among networks
output : Overall amount of signaling overhead using
         group-mobility indirection (denoted by $\mathcal{U}$)

1  for each time window do
2      if GNS uses a periodicity-based algorithm then
3          if GNS changes the AGUID's default network then
4              $\mathcal{U} \leftarrow \mathcal{U} + 2$;          ▷ ACK/NAK;
5      for each GUID do
6          while The GUID connects with a network, disconnects
           with a network, or stays connected with a network in
           the time window do
7              if GNS uses an event-based algorithm and the
               GUID is a leader then
8                  if The GUID connects to a new network then
9                      $\mathcal{U} \leftarrow \mathcal{U} + 3$;   ▷ ACK/NAK/connection;
10                 else if The GUID disconnects with a network
                   then
11                     $\mathcal{U} \leftarrow \mathcal{U} + 0$;
12                 continue;
13             if The GUID connects to a network then
14                 if The network is associated with the
                   AGUID's default network then
15                     $\mathcal{U} \leftarrow \mathcal{U} + 0$;          ▷ NONE;
16                 else
17                     $\mathcal{U} \leftarrow \mathcal{U} + 1$;        ▷ connection;
18             if The GUID disconnects with a network then
19                 if The network is associated with the
                   AGUID's default network then
20                     $\mathcal{U} \leftarrow \mathcal{U} + 0$;          ▷ NONE;
21                 else
22                     $\mathcal{U} \leftarrow \mathcal{U} + 1$;      ▷ disconnection;
23             if The GUID stays connected with a network then
24                 if The network is not associated with the
                   AGUID's default network then
25                     $\mathcal{U} \leftarrow \mathcal{U} + 1$;           ▷ rLIST;
```

Given a series of n GUIDs mobility among networks over T time windows, we must determine the amount of signaling overhead incurred as a result of these moves using group-mobility indirection. The overall amount of signaling overhead in the case of no group indirection is solely calculated by counting the total number of location-update messages incurred whenever the GUIDs move to a new network.

Function 1 indicates how the overall amount of signaling overhead is computed under group-mobility indirection. First, lines 2-12 count signaling resulting from an AGUID's default-network change. In the case of a periodicity-based algorithm (as shown in lines 2-4), the two ACK/NAK messages are counted as signaling overhead. In the case of an event-based algorithm (as shown in lines 7-12), the ACK/NAK messages and a location-update message for the leader's connection to a new network are counted, when the leader moves to a new network. At each time window, we must additionally account for the signaling for each GUID that connects to a network, disconnects from a network, or stays connected in a network; location-update message(s) and an rLIST message must also be counted as part of signaling overhead, depending on whether the GUID is associated with its AGUID's default network in that time window. Lines 13-17 show the number of location-update messages incurred when the GUID connects to a network; lines 18-22 show the number of location-update messages incurred when the GUID disconnects from a network; lines 23-25 count an rLIST message if the GUIDs remains in its AGUID's old default network but its AGUID's default network changes.

4.5.3 Numerical results

We evaluate and discuss the reduction in signaling overhead using group-mobility indirection for our event-based algorithms, under different leader-election strategies. We consider different degrees of group coherence, $\mathcal{C}$, for an AGUID, generating eleven sets of synthetic traces whose degrees of group coherence are $\mathcal{C} = 0, 0.1, \cdots, 1$, as we describe below. Our analysis consists of 1000 GUIDs associated with an AGUID, with 20 networks and 10,000 time windows. We generate synthetic traces of GUID and AGUID mobility as follows:

1) For each degree of group coherence, we arbitrarily assign the values of α and β that satisfy the given group coherence degree by generating α between 0 and 1 and then picking $\beta = 1 - \alpha$. For a given set of α, β values, we then randomly generate a pair of α_i, β_i values (associated with each of the i GUID$_i$, $i = 1, \cdots, 1000$) constrained to be between 0 and 1 using *exponential random variate* [38] as follows. Assuming that α_i follows exponential(α), exponential random variate generates α_i value using function $-\frac{\ln U(0,1)}{\alpha}$, where $U(0,1)$ is the uniform distribution on the unit interval (0, 1); here, we adopt the acceptance-rejection method [1] that rejects a returned value of the function if the value is not between 0 and 1 and tries again to generate a value until the value is between 0 and 1. The value of β_i is produced similarly.

2) For each degree of group coherence and for each of the 1000 pairs of α_i and β_i values in the Step 1), we generate a series of 10,000 time-window movements of GUID$_i$ (for all i). Recall that the next state of the Markov chain model for i is determined by the values of α_i and β_i. The next network location is determined by matrices $\mathcal{S}^i$ and $\mathcal{G}$. Here, we assume that a GUID equally likely moves to one of $m-1$ networks (other than a current network location), and thus define matrices $\mathcal{S}^i$ and $\mathcal{G}$ as follows.

$$\mathcal{S}^i_{ab} = \quad \mathcal{G}_{ab} = \quad \begin{cases} \gamma & \text{, if } a = b \\ \frac{1-\gamma}{m-1} & \text{, otherwise} \end{cases}$$

3) We run the above Steps 1) and 2) 100 times.

For each synthetic trace (each associated with a different degree of group coherence) we use function 1 to calculate the overall amount of signaling overhead under each of the proposed leader election strategies. Function 1 assumes the best case scenario in which an ACK is delivered before GUIDs (other than a leader) connect to a new default network, and that a NAK is delivered after GUIDs disconnect from an old default network.

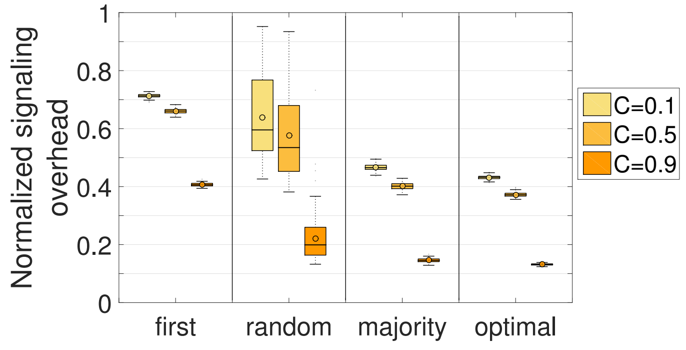
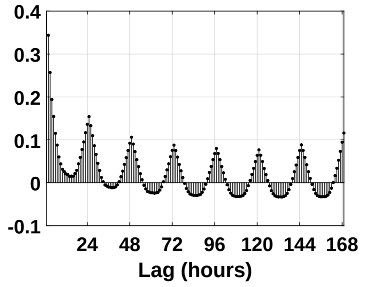


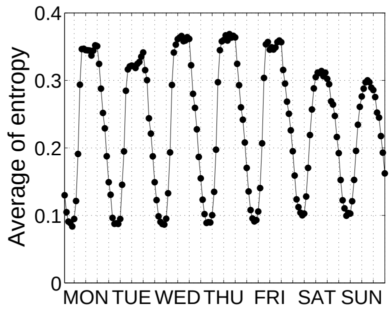
Figure 4.1: Normalized signaling overhead ($\gamma = 0.5$): event-based algorithm for three leader strategies versus no group indirection

Figure 4.1 shows a box plot with whiskers (maximum, 75th percentile, median, 25th percentile, minimum as lines, and mean as a circle) for 100 independent runs. The y-axis is the ratio of the amount of signaling overhead when group-mobility indirection is used (for a given leader election strategy) and the amount of signaling overhead when group-mobility indirection is not used. Three different degrees of group coherence are considered: 0.1, 0.5, and 0.9. The first three columns show normalized signaling overhead using our event-based algorithm with first-leader, random-leader, and majority-leader strategies, respectively. The last column shows an optimum of normalized signaling overhead using an optimal algorithm where $T = 1$. Figure 4.1 shows that the higher the degree of group coherence, the more reduced signaling. Figure 4.1 also shows that all leader-election strategies for the event-based approach to group indirection are useful – all normalized signaling overhead values are smaller than 1. The event-based algorithm with a majority leader, as shown on the third column, achieves the best performance, with a reduction of signalizing overhead that is close to the optimal.

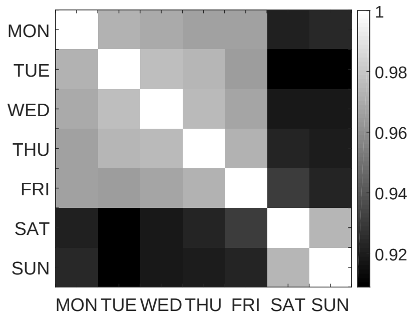
We have also investigated an additional group mobility model based on a two-dimensional grid in which a GUID makes either a dependent or independent (of



(a) Correlogram



(b) Entropy of locations for each hour of the week



(c) Similarities of network locations between different days of the week

Figure 4.2: Periodicity in sequences of users' network locations: autocorrelation, entropy, and correlation

other GUIDs in its current cell) decision to stay in its current network of move. An independent GUID decides whether to move up, down, left, or right from its current cell. Dependent group movement follows a gravity model in which the probability to move up, down, left or right is proportional to the number of group-members that are strictly in a row above or below, or in a column to the left or right of that group's current cell. We observed quantitatively similar results to the above Figure 4.1.

4.6 Evaluation: Clustering and Empirical investigation

In this Section, we use measured traces of actual user mobility to evaluate the performance of our two group-mobility approaches. Since our periodicity-based algorithm seeks to exploit predictable correlation in user mobility, we begin by describing our measurement traces, and then investigate the existence of multiple groups and periodicity in these traces. Then, we cluster a set of GUIDs in the traces into multiple AGUIDs, and evaluate the performance of our group-mobility approaches by again (as in the previous Section) quantifying signaling overhead reduction.

4.6.1 Empirical traces

We used a subset of IMAP mail logs [76] collected from University of Massachusetts Amherst IMAP servers that support an email retrieval service for campus users, in order to reconstruct a user's mobility among networks. The subset contains 3660 UMass-affiliated users; these users were associated with access networks in 1152 different autonomous systems (ASes). As in [76], our results characterize a user's (perhaps minimally observed only via mail activities) mobility among AS-level networks.

From the IMAP mail logs, we reconstruct a user's session as follows. Each IMAP log entry, collected by *syslog*, consists of three attributes: *(i)* a timestamp when a log entry is recorded, *(ii)* a user's mail access activity (e.g., connection establishment, sign-in, and mailbox access), and *(iii)* a *process ID* assigned when a successful session starts and persistently kept until that session ends[4]. Then, given a series of logs having the same process ID, we identify a set of (successfully established) associations between a user and various access networks. The session start time is determined from the time of an entry in the connection-establishment log, followed by sign-in and an entry in the authentication log (as in [22]). The time of a last log entry recorded before another connection-establishment log or before a unilateral server decision to close an idle connection[5] determine that session's end time. A sign-in log entry recorded between the connection-establishment log entry and the last log entry identifies the user and that user's access network.

Having described our traces, we next investigate the existence of multiple groups and periodicity in these traces.

Number of potential AGUIDs in the empirical traces. We observed that approximately 25% of users accessed only one AS; approximately 71% of users accessed

[4]When a user accesses an IMAP server, a process id is assigned by the user's device or the IMAP server [22]. In session reconstruction, we loosely assume that a process id is not reused until a session associated with the process ID ends.

[5]An IMAP log entry rarely contains a user's explicit disconnection from a mail server.

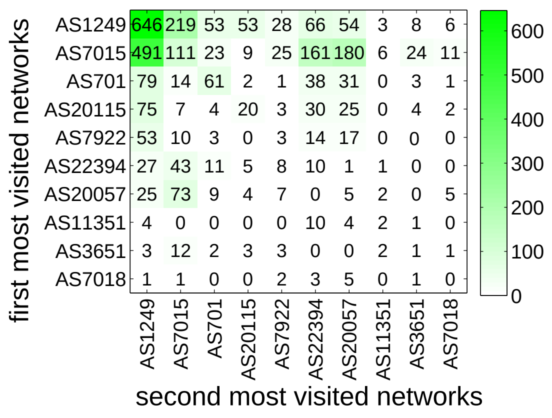


Figure 4.3: User frequency over pairs of first and second most frequently associated networks.

more than one ASes and produced more than 70% of their sessions in their "top two" ASes. We also observed that all users produced approximately 90% of their sessions from ten ASes out of the 1152 ASes observed in the traces: AS1249 (Five colleges network including UMass Amherst), AS7015 (Comcast), AS701 (Verizon), AS20115 (Charter communication), AS7922 (Comcast), AS22394 (Verizon), AS20057 (AT&T), AS11351 (Time Warner Cable), AS3651 (Sprint), and AS7018 (AT&T). Thus, the empirical traces *do* potentially contain multiple groups of users with correlated network associations, some of which consist of users mostly associated with a single AS, suggesting that their default network would rarely change over time. Additional groups might consist of users most frequently associated with two particular ASes, with their default networks being (periodically) associated with either of the two ASes. Figure 4.3 plots the asymmetric heatmap of the number of users over each pair of the ten most popular ASes, and shows that 491 users, for instance, were most frequently associated with AS7015 (first) and AS1249 (second). Thus a potential AGUID might be for users transitioning between networks AS7015 and AS1249.

Periodicity in the empirical traces. Before assessing the daily and weekly periodicity of network associations in the traces, we characterize the networks with which user i's is associated in each time window as follows. Let T be the total number of time windows (e.g., 1 hour) in a measurement period of the traces, and recall that there are m networks. Let $\mathcal{A}^i = [\mathcal{A}^i_{tj}]$ be a $T \times m$ matrix whose element $\mathcal{A}^i_{tj}$ is the network association probability that user i connects to network j in time window t, subject to $\sum_j \mathcal{A}^i_{tj} = 1$. We also define $X^i = \{X^i_t\}^T_{t=1}$, where $X^i_t = \arg\max_j \left(\mathcal{A}^i_{tj}\right)$; thus X^i denotes a sequence of user i's most associated network during each time window t. Given the traces, we calculate $\mathcal{A}^i_{tj}$ as the number of times that user i connects to network N_j at t, divided by the total number of times that user i connects to any network(s) at t.

With a set of nominal data $X = (X^1, X^2, \cdots, X^{3660})$, we investigate the periodicity of an individual user's network associations using the autocorrelation function (ACF) [71]. The ACF measures the degree of similarity among time-series data at varying time lags, and detects periodicity in the data and its period. Consider that each sequence X^i consists of one or more network locations. For each of X^i's constituent network(s), we convert X^i to a boolean series by taking the network as 1 and the rest of networks as 0s. Then we derive the ACF of the boolean series and identify the period of the series associated with that network. Figure 4.2(a) plots the average values of autocorrelations of all boolean series over all users as a function of time lags. Figure 4.2(a) shows peaks at a time lag of every 24 hours, and thus identifies 24 hours or multiples of 24 hours as periods in users network associations.

Variation among the networks to which an individual user connects during each hour of the week. We use the Shannon entropy to characterize the variability of the networks with which user i is associated at each hour, h, of the week:

$$H^i_h \quad = \quad -\sum_j \left(\mathcal{A}^i_{hj} \cdot \log \mathcal{A}^i_{hj}\right).$$

The lower the entropy value, the lower the variability of the data. Figure 4.2(b) plots the average entropy values for each hour of the week over all users. For all days, the hours roughly from 11pm to 7am have the lowest entropy (and thus exhibit more similarity in these time periods); this is when most users are at home. The hours roughly from 9am to 6pm have the highest entropy; this is when most users are at work thus are more likely associated with diverse networks. Also, weekend days show lower entropy values than week days. In summary, Figure 4.2(b) suggests that the network with which a user is associated from 11pm-to-7am is likely to be more predictable than the associated network from 9am-to-6pm.

Similarity in an individual user's networks during different days of the week. The correlation [3] measures the degree of similarity between two variables; the reflective correlation is a variant of the correlation in which the data are not centered around their mean values. The sample reflective correlation of user i's networks between two different days of the week d_1, d_2 is defined by:

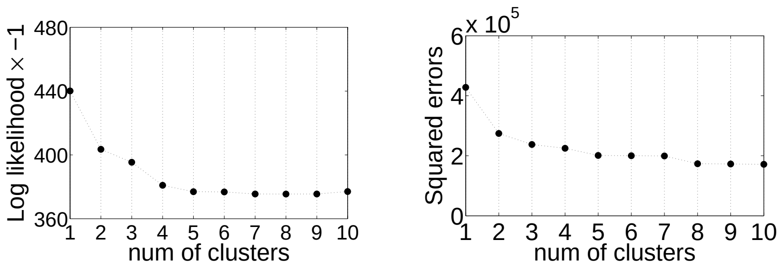
$$rr^i_{d_1 d_2} = \frac{\sum_j \left(\mathcal{A}^i_{d_1 j} \cdot \mathcal{A}^i_{d_2 j}\right)}{\sqrt{\sum_j (\mathcal{A}^i_{d_1 j})^2 \cdot \sum_j (\mathcal{A}^i_{d_2 j})^2}}.$$

The closer the correlation is 1, the stronger the correlation between the variables. Figure 4.2(c) plots the average correlation values between different days of the week over all users, and shows that the networks associated with users during weekend days are slightly different from those during weekdays. Thus, a periodic sequence of networks associated with an AGUID has a natural "one week" (i.e., 168 hours) period (see Definition 1).

4.6.2 User clustering for AGUID membership identification

We bisect the traces set into a *training set* (5 weeks of logs from Jan 21st to Feb 24th) and a *validation set* (4 weeks of logs from Feb 25th to Mar 24th). The training set was used to cluster 3660 users into multiple AGUIDs; the validation set was used to evaluate the performance (reduction in signaling overhead) of our two approaches using the resultant AGUIDs.

We use the training set to perform user clustering as follows. We characterize user i by a one-week periodic sequence that consists of a series of that user's most frequently associated network for each hour of the week, denoted by $\tilde{X}^i = \{\tilde{X}^i_h\}_{h=1}^{168}$, where $\tilde{X}^i_h = \arg\max_j(\mathcal{A}^i_{hj})$[6]. Given a set of nominal data $\tilde{X} = (\tilde{X}^1, \cdots, \tilde{X}^{3660})$, we apply clustering algorithms to partition the 3660 users into K clusters based on the similarity of their one-week sequences. We considered Expectation Maximization (EM) and K-means clustering algorithms, and adopted WEKA's nominal data clustering implementation [75]. EM determines the probabilistic description (in terms of the frequency counts for each associated network) of each cluster as the set of users that maximizes the log-likelihood of the data. K-means describes each cluster as the set of users that minimizes the distances from the centroids of the cluster.



(a) EM (b) K-means

Figure 4.4: Best number of clusters.

When, then, is the best number of clusters? Figure 4.4(a) plots the average log-likelihood values of 10-fold cross-validation data in EM clustering as a function of the number of clusters. Figure 4.4(b) plots the sum-of-squared errors in K-means clustering (with K-means++ [11] that carefully chooses the initial values of K-means clustering) as a function of the number of clusters. Both figures show that the curve quickly decreases up to five clusters and flattens out, suggesting five clusters be used.

[6]If user i gets offline at a given hour of the week h – no IMAP log, then X^i_h is set a last network which user i is associated with.

	EM		K-means	
	# of users	networks	# of users	networks
[1]	1570	AS7015 (87%), AS1249 (13%)	1510	AS1249 (100%)
[2]	931	AS1249 (100%)	916	AS7015 (100%)
[3]	651	AS20115 (81%), AS1249 (19%)	613	AS7015 (73%), **AS1249 (27%)**
[4]	317	AS701 (93%), AS1249 (7%)	365	AS701 (95%), AS1249 (5%)
[5]	191	AS7922 (88%), AS1249 (12%)	238	AS20115 (96%) AS1249 (4%)

Table 4.1: Five clusters

Table 4.1 shows the resultant five clusters using EM and K-means clustering. The first column shows cluster IDs. The next two columns show EM clustering results. The last two columns show K-means clustering results. One of the two columns shows the number of users belonging to each cluster, and the other shows the network(s) that constitute a one-week periodic sequence of each cluster with the percent of such network's occupancy in that sequence.

As a baseline, while considering that 3660 users spent their most of time in one or two ASes, it is also interesting to manually partitioning a set of users on the basis of the network with which they are most frequently associated. This manual clustering results in 161 clusters as follows. Seven clusters consist of more than 100 users, and their single most frequently-associated ASes. The AS and number of users in such a clustering are AS1249 (1184 users), AS7015 (1150 users), AS701 (266 users), AS20115(185 users), AS20057 (145 users), AS22394 (127 users), and AS7922 (119 users). 90 clusters contain only one user. The remaining clusters contain between 2 and 100 users.

4.6.3 Performance Results: reducing the normalized signaling overhead

Let us now quantitatively compare the performance of our event-based and periodicity-based algorithms under three different approaches for determining the

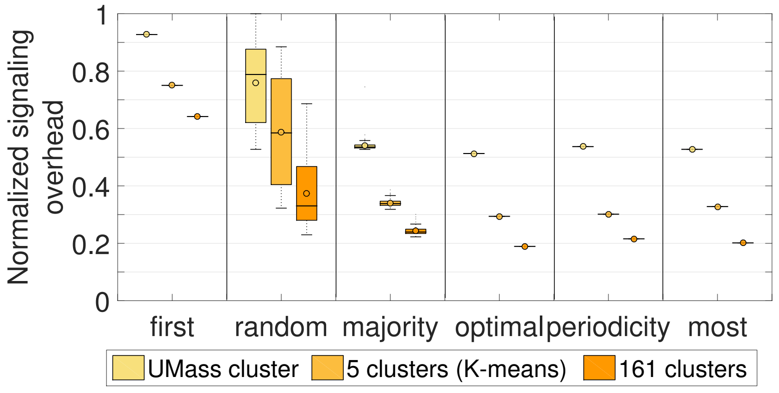


Figure 4.5: Aggregate normalized signaling overhead over all clusters: diverse algorithms for determining an AGUID's default network versus no group indirection

members of the AGUID: *(i)* K-means clustering (with K=5), as discussed above; we present K-means results only since it shows better performance (a decreased amount of signaling) than that achieved by EM, *(ii)* manual clustering based on most-frequently-associated network, which resulted in 161 custers, as discussed above, *(iii)* a single cluster in which all 3660 users are associated with one AGUID. For each of clustering results, Figure 4.5 plots a box plot with whiskers on one or 100 runs (as explained below) of aggregate normalized signaling overhead over all clusters. We calculated aggregate normalized signaling overhead using function 1 (as in Section 4.5) for each algorithm.

The first three columns show the normalized signaling overhead for the event-based first-leader, random-leader, and majority-leader strategies, respectively. The fourth column shows an optimal algorithm (where $T = 1$). The fifth column shows a periodicity-based algorithm in which each cluster's default network is periodically updated in accordance with its one-week periodic sequence, derived from the training set using Definition 2. The last column considers the case where each cluster's default network is *not* changed in time; we study this case to determine the marginal utility of adjustively changing each cluster's default network over time, as in the results in the fifth column. We evaluated normalized signaling overhead of the given traces only once in an event-based algorithm with a first leader, an optimal algorithm, a periodicity-based algorithm, and the last column. However, we evaluated applying random-leader and majority-leader strategies 100 times each, whose normalized signaling overhead can change every run.

Figure 4.5 shows that our periodicity-based algorithm (Column 5) yields the most significant reduction in signaling overhead among all of our algorithms, achieving performance close to the optimal. Interestingly, the approach of adaptively changing the predicted default network on a hourly basis for the group provides only a small gain over the case where the cluster's default network is not changed in time. In the course of evaluating an individual cluster's performance, K-means cluster [3] shows non-negligible occupancy 27% of its second most frequently associated network (as shown Table 4.1). Using the cluster [3]'s one-week periodic sequence (associating AS1249 for weekdays 9am-to-6pm and AS7015 for the remained hours of weekdays and all weekend days), a periodicity-based algorithm achieves approximately 33% reduced normalized signaling, over fixing its first most frequently associated network AS7015.

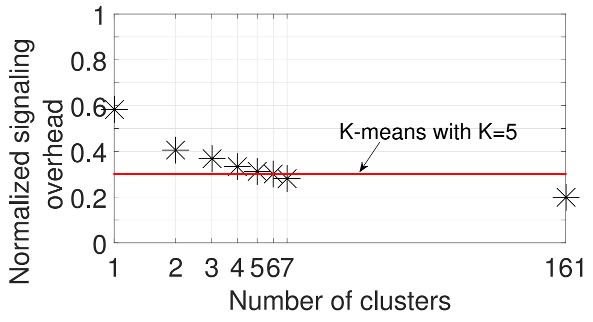


Figure 4.6: Relation of the number of AGUIDs (resulted from manual clustering) to normalized signaling overhead.

As expected, comparing the cases of 1, 5 and 161 clusters, Figure 4.5 shows that having more clusters results in an increased reduction in signaling overhead (i.e., improved performance). Figure 4.6 plots the aggregate normalized signaling overhead in a periodicity-based algorithm over all clusters when only x out of the 161 clusters (in increasing order of the number of their constituent GUIDs) are used; the red solid line denotes aggregate normalized signaling of K-means with K=5 using a periodicity-based algorithm. For instance, star-shaped black dot on $x = 6$ denotes aggregate signaling when the users associated with six largest clusters (listed above) are kept track of by each cluster but the rest of users not associated with these six clusters are individually tracked. Figure 4.6 shows that six manually partitioned clusters, consisting of more than 100 users, achieve the reduction performance close to K-means five clusters.

4.7 Related Work

Group mobility has been the subject of a number of past research studies. In mobile ad hoc networks and delay tolerant networks, group mobility – a group of mobile objects that are typically physically proximate – has been theoretically modeled (e.g., [13,51]) and also empirically investigated using real-world traces (e.g., [36]). Data mining researchers [43] have also performed trajectory clustering to detect a cluster of objects moving physically close to each other for a long time interval. However, from a network-architecture and -protocol perspective, group mobility mechanisms and approaches (as defined in Section 4.1) have been less well-explored.

Several networking research efforts have employed hierarchy techniques to efficiently handle a large burst of location-update signaling overhead resulted from the similarity of a *logically static* group of users' movement patterns. The IETF Network Mobility (NEMO) Working Group [23] developed a protocol that defines a group of mobile users as a logical subnet where a mobile gateway router – moving together with the group – is only directly connected to the Internet and takes responsible for updating a new location of the group on behalf of the users associated with the group, as the group moves. On the other hand, [2,16] clustered a set of BSs or APs in a wireless infrastructure as a logical subnet based on the similarity of a group of users' BS or

AP association patterns, so that user mobility within such subnet does not generate location-update overhead to inform a home agent of a new location. However, such static hierarchical solutions require a mobile gateway router installed for each group, or are only useful when the history of users who move among only a few APs or BSs is available.

4.8 Conclusion

In this Chapter, we introduced the concept of group-mobility indirection and proposed several algorithms to determine the network associated with a group's location. We evaluated the reduction in the location-tracking (signaling) overhead for these algorithms via a synthetic group-mobility model and via empirical traces of approximately 4000 mobile users. We found that an event-based algorithm that elects one group member as a leader and reactively associates the group location with this leader's location significantly reduces location-tracking overhead, as long as a group of users move together frequently enough. For the case that the sequence of networks associated with a group of users has periodicity, we showed that an algorithm that periodically updates the group location with a predicted network location at a predicted time reduces location-tracking overhead more than the event-based algorithm. We also showed the gain in reducing the location-tracking signaling as the number of groups increases.

CHAPTER 5
CONCLUSION

5.1 Thesis Summary

In this thesis, we have considered a number of forensics and mobility management challenges that arise in wireless/mobile network environments, using diverse research techniques such as measurement studies, modeling, machine learning algorithms, architecture and protocol designs, and algorithms. The first half of this thesis addressed two network forensic challenges that arise due to the broadcast nature of wireless communications. In Chapter 1, we proposed a forwarding misbehavior attack detection mechanism and analytically evaluated our mechanism's detection accuracy in diverse threat scenarios, in the case of lossy links. In Chapter 2, we presented a legal method to remotely disambiguate a sender's network access type (wired versus wireless) via supervised learning classification and discussed classification performance using traffic generated in controlled measurement environments. The second half of this thesis delved into a clean-state approach to designing a future Internet architecture that considers mobility as a first-order property. In Chapter 3, we presented and discussed a measurement and modeling study of use-transitioning among points of attachment to today's Internet; this work provides insights and implications regarding control-plane workloads for existing proposed and future mobility management architectures. In the last Chapter, we designed an indirection technique to efficiently handle a group of users moving together, and discussed the reduction in control-plane workloads using synthetic traces (generated by our generative group mobility model) and empirical traces.

5.2 Future Work

The research in this thesis can be extended in several directions:

- In Chapter 1, our witness-based detection scheme, which was explored in a static wireless environment, can be extended by considering a scenario in which nodes on a data path and witness nodes are mobile. It would be interesting to investigate the impact of node mobility on the detection accuracy of the scheme as well as other possible threats on data forwarding in such mobile scenario.

- In Chapter 2, the problem of locating a user inside a building using remotely measured traces in our forensic scenario by distinguishing wired access from wireless access can be extended to consider diverse properties of wireless links, such as a distance between an AP and a user, physical obstructions, and the level of wireless network interference.

- In Chapter 3, the problem of a (multi-homed) user's transitioning among access networks, which was considered from the perspective of the control plane, can be extended to investigate the impact of such user mobility on the data plane, such as session continuity across devices or access networks.

- In Chapter 4, the analysis of group-mobility indirection can be extended to consider the gain of latency reduction between the GNS and a user, and among the distributed GNSes for user-location information synchronization, compared with individually updating user-location information, as a group of users move. It would also be interesting to investigate the performance impact of the level of network granularity (e.g., APs, BSes) in establishing a group.

More broadly, the dramatic growth of mobile users makes it more important to consider mobility scenarios in network and security research problems. Such an emphasis on mobility requires the measurement and analysis of mobility (whether virtual or physical) on network architectures, system performance, and network planning/dimensioning in the control and data plane. Statistical analytics of network data and patterns in mobile environments will provide new opportunities to improve network algorithms and protocols to work more robustly and intelligently in diversely changing wireless and mobile network environments.

BIBLIOGRAPHY

[1] *acceptance-rejection method,* `http://planetmath.org/acceptancerejectionmethod`.

[2] Cisco wireless lan controller configuration guide, release 7.0.

[3] *Pearson product-moment correlation coefficient,* `https://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient`. Wikipedia.

[4] *Computer Crime Law*. Thomson/West, 2006.

[5] Google glass, 2013.

[6] The smart watch review, `http://www.thesmartwatchreview.com/`, 2013.

[7] Team cymru research nfp, ip to asn mapping, `http://www.team-cymru.org/Services/ip-to-asn.html`, 2013.

[8] Cisco visual networking index: Global mobile data traffic forecast update, 2013 Feb.

[9] *StattSoft: Electronic Statistics Textbook*. http://www.statsoft.com/textbook/cluster-analysis/, Junho de 2012.

[10] 3GPP. 3gpp specifications, `http://www.3gpp.org/specifications`.

[11] avid Arthur, and Vassilvitskii, Sergei. k-means++: the advantages of carefull seeding. *ACM-SIAM symposium on Discrete algorithms 2007*.

[12] Bishop, Christopher M. *Pattern Recognition and Machine Learning*. Springer, 2006.

[13] Borrel, Vincent, de Amorim, Marcelo Dias, and Fdida, Serge. A preferential attachment gathering mobility model. *IEEE Communications Letters 2005*.

[14] Buchegger, Sonja, and Le Boudec, Jean-Yves. Performance analysis of the confidant protocol. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing* (New York, NY, USA, 2002), MobiHoc '02, ACM, pp. 226–236.

[15] CableLabs. Data-Over-Cable Service Interface Specifications (DOCSIS).

[16] Caceres, Ramon, and Padmanabhan, Venkata N. Fast and scalable handoffs for wireless internetworks. *ACM MOBICOM 1996*.

[17] Casey, E. *Digital evidence and computer crime: forensic science, computers and the Internet.* Academic Pr, 2004.

[18] Chaintreau, A., Hui, Pan, Crowcroft, J., Diot, C., Gass, R., and Scott, J. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Trans. Mobile Computing 6*, 6 (2007), 606–620.

[19] Chen, Yingjie, Liu, Zhongli, Liu, Benyuan, Fu, Xinwen, and Zhao, Wei. Identifying Mobiles Hiding behind Wireless Routers. In *Proc. IEEE INFOCOM* (Apr 2011), pp. 2651–2659.

[20] Chen, Yung-Chih, Kurose, Jim, and Towsley, Don. A mixed queueing network model of mobility in a campus wireless network. In *IEEE INFOCOM* (2012), pp. 2656–2660.

[21] Cisco.com. Understanding Data Throughput in a DOCSIS World.

[22] Crispin, M. Internet Message Access Protocol (IMAP). RFC 3501, Mar. 2003.

[23] Devarapalli, Vijay, Wakikawa, Ryuji, Petrescu, Alexandru, and Thubert, Pascal. Network mobility (nemo) basic support protocol. RFC 3963, January 2005.

[24] Farinacci, D., Fuller, V., Meyer, D., and Lewis, D. RFC 6830: The Locator/ID Separation Protocol (LISP), Jan. 2013.

[25] Fenner, W. Internet Group Management Protocol, Version 2. RFC 2236, November 1997.

[26] Gast, Matthew. *802.11 Wireless Networks: The Definitive Guide Creating and Administering Wireless Networks.* O'Reilly Media, 2002.

[27] Gerhards, R. The Syslog Protocol. RFC 5424, March 2009.

[28] Gettys, Jim, and Nichols, Kathleen. Bufferbloat: Dark buffers in the internet. *Queue 9*, 11 (Nov. 2011), 40:40–40:54.

[29] Goga, Oana, Lei, Howard, Parthasarathi, Sree Hari Krishnan, Friedland, Gerald, Sommer, Robin, and Teixeira, Renata. Exploiting innocuous activity for correlating users across sites. In *WWW '13*, pp. 447–458.

[30] Ha, Sangtae, Rhee, Injong, and Xu, Lisong. Cubic: a new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev. 42*, 5 (July 2008), 64–74.

[31] Halepovic, Emir, and Williamson, Carey. Characterizing and modeling user mobility in a cellular data network. In *in ACM PE-WASUN* (2005).

[32] Han, Dongsu, Anand, Ashok, Dogar, Fahad, Li, Boyan, Lim, Hyeontaek, Machado, Michel, Mukundan, Arvind, Wu, Wenfei, Akella, Aditya, Andersen, David G., Byers, John W., Seshan, Srinivasan, and Steenkiste, Peter. XIA: Efficient support for evolvable internetworking. *USENIX NSDI 2012*.

[33] He, Qi, Wu, Dapeng, and Khosla, Pradeep. Sori: a secure and objective reputation-based incentive scheme for ad-hoc networks. In *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE* (2004), vol. 2, pp. 825–830 Vol.2.

[34] Hsu, Wei-jen, Dutta, Debojyoti, and Helmy, Ahmed. Structural analysis of user association patterns in university campus wireless lans. *IEEE Trans. Mobile Computing 11*, 11 (Nov. 2012), 1734–1748.

[35] Huan, Qiang, Avramopoulos, I.C., Kobayashi, Hisashi, and Liu, B. Secure data forwarding in wireless ad hoc networks. In *IEEE ICC 2005* (2005), vol. 5, pp. 3525–3531 Vol. 5.

[36] Hui, Pan, Yoneki, Eiko, Chan, Shu Yan, and Crowcroft, Jon. Distributed community detection in delay tolerant networks. *ACM/IEEE international workshop on Mobility in the evolving Internet architecture 2007*.

[37] Isaacman, Sibren, Becker, Richard, Cáceres, Ramón, Martonosi, Margaret, Rowland, James, Varshavsky, Alexander, and Willinger, Walter. Human mobility modeling at metropolitan scales. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services* (New York, NY, USA, 2012), MobiSys '12, ACM, pp. 239–252.

[38] J. Brockwell, Peter, and Richard, A. Davis. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* Springer, 2002.

[39] Johnson, D., Perkins, C., and Arkko, J. Mobility Support in IPv6. RFC 3775, June 2004.

[40] Johnson, D., Perkins, C., and Arkko, J. RFC 3775: Mobility Support in IPv6, June 2004.

[41] Kerr, O.S. *Computer Crime Law.* Thomson/West, 2006.

[42] Kim, Minkyong, Kotz, David, and Kim, Songkuk. Extracting a mobility model from real user traces. In *IIEEE NFOCOM* (Barcelona, Spain, April 2006), IEEE Computer Society Press.

[43] Kisilevich, Slava, Mansmann, Florian, Nanni, Mirco, and Rinzivillo, Salvatore. *Spatio-temporal clustering.* Springer, 2009.

[44] Komninos, Nikos, Vergados, Dimitris, and Douligeris, Christos. Detecting unauthorized and compromised nodes in mobile ad hoc networks. *Ad Hoc Networks 5*, 3 (2007), 289 – 298.

[45] Kravets, David. 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.

[46] Kwiatkowski, Denis, Phillips, Peter C. B., Schmidt, Peter, and Shin, Yongcheol. Testing the null hypothesis of stationarity against the alternative of a unit root : How sure are we that economic time series have a unit root? *Journal of Econometrics 54*, 1-3 (00 1992), 159–178.

[47] Liberatore, Marc, Erdely, Robert, Kerle, Thomas, Levine, Brian Neil, and Shields, Clay. Forensic Investigation of Peer-to-Peer File Sharing Networks. In *Proc. DFRWS* (August 2010).

[48] Marti, Sergio, Giuli, T. J., Lai, Kevin, and Baker, Mary. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking* (New York, NY, USA, 2000), MobiCom '00, ACM, pp. 255–265.

[49] Microsoft.com. Network Monitor 3.4, 2011.

[50] Mislan, Richard P., Casey, Eoghan, and Kessler, Gary C. The growing need for on-scene triage of mobile devices. *Digital Investigation 6*, 3-4 (2010), 112–124.

[51] Musolesi, Mirco, and Mascolo, Cecilia. Designing mobility models based on social network theory. *ACM SIGMOBILE MC2R 2007*.

[52] Nichols, Kathleen, and Jacobson, Van. Controlling queue delay. *Queue 10*, 5 (May 2012), 20:20–20:34.

[53] Openmaniak.com. IPERF - The Easy Tutorial. `http://openmaniak.com/iperf.php`.

[54] Padhye, Jitendra, and Kurose, James F. Continuous-media courseware server: A study of client interactions. *IEEE Internet Computing 3*, 2 (1999), 65–73.

[55] Paul, U., Subramanian, A.P., Buddhikot, M.M., and Das, S.R. Understanding traffic dynamics in cellular data networks. In *IEEE INFOCOM* (2011), pp. 882–890.

[56] Project, Pew Internet Research. Pew internet research project, `http://www.pewinternet.org/`, 2014.

[57] Ramanujan, R., Kudige, S., Takkella, S., Nguyen, T., and Adelstein, F. Intrusion-resistant ad hoc wireless networks. In *MILCOM 2002. Proceedings* (2002), vol. 2, pp. 890–894 vol.2.

[58] Rosenberg, Jonathan, Schulzrinne, Henning, Camarillo, Gonzalo, Johnston, Alan, Peterson, Jon, Sparks, Robert, Handley, Mark, and Schooler, Eve. Sip: session initiation protocol. RFC 3261, June 2002.

[59] Russell, S.J., and Norvig, P. *Artificial intelligence: a modern approach.* Prentice Hall.

[60] Sharma, Puneet, Estrin, Deborah, Floyd, Sally, and Jacobson, Van. Scalable timers for soft state protocols. *IEEE INFOCOM 1997*.

[61] Shore, Randy. Pedophiles exploiting wireless loopholes. The Vancouver Sun, http://www.canada.com/vancouversun/news/story.html?id=cff3073b-ceea-4ba4-877f-d020715358e9, February 13 2007.

[62] Sicker, D.C., Ohm, P., and Grunwald, D. Legal issues surrounding monitoring during network research. In *Proc. ACM IMC* (Oct. 2007), pp. 141–148.

[63] Siekkinen, M., Collange, D., Urvoy-Keller, G., and Biersack, E. Performance limitations of ADSL users. In *Proc. PAM Conference* (2007), pp. 145–154.

[64] Sterne, D., Balasubramanyam, P., Carman, D., Wilson, B., Talpade, R., Ko, C., Balupari, R., y. Tseng, C, Bowen, T., Levitt, K., and Rowe, J. A general cooperative intrusion detection architecture for manets. In *In IWIA 05: Proceedings of the Third IEEE International Workshop on Information Assurance (IWIA05* (2005), IEEE Computer Society, pp. 57–70.

[65] Tan, Cheng Lin, and Pink, Stephen. Mobicast: A multicast scheme for wireless networks, 2000.

[66] Tan, Kun, and Song, Jingmin. A compound tcp approach for high-speed and long distance networks. In *In Proc. IEEE INFOCOM* (2006).

[67] U.S. General Accounting Office. File-Sharing Programs – Child Pornography Is Readily Accessible over Peer-to-Peer Networks. GAO-03-537T. Statement Before Congress of Linda D. Koontz, March 2003.

[68] Venkataramani, A., Kurose, J., Raychaudhuri, D., Nagaraja, K., Mao, M., and Banerjee, S. Mobilityfirst: A mobility-centric and trustworthy internet architecture. *ACM CCR* (2014).

[69] Venkataramani, Arun, Kurose, James F., Raychaudhuri, Dipankar, Nagaraja, Kiran, Mao, Morley, and Banerjee, Suman. Mobilityfirst: A mobility-centric and trustworthy internet architecture. *ACM SIGCOMM CCR 2014*.

[70] Venkataramani, Arun, Sharma, Abhigyan, Tie, Xiaozheng, Uppal, Hardeep, Westbrook, David, Kurose, Jim, and Raychaudhuri, Dipankar. Design requirements of a global name service for a mobility-centric, trustworthy internetwork. In *IEEE COMSNETS* (2013).

[71] Vlachos, Michail, Yu, Philip, and Castelli, Vittorio. On periodicity detection and structural periodic similarity. *SIAM International Conference on Data Mining 2005*.

[72] Wei, W., Jaiswal, S., Kurose, J., and Towsley, D. Identifying 802.11 Traffic from Passive Measurements Using Iterative Bayesian Inference. In *Proc. IEEE INFOCOM* (April 2006).

[73] Wei, W., Suh, K., Wang, B., Gu, Y., Kurose, J., and Towsley, D. Passive online rogue access point detection using sequential hypothesis testing with TCP ACK-pairs. In *Proc. ACM IMC* (Oct. 2007), pp. 365–378.

[74] Wei, W., Wang, B., Zhang, C., Kurose, J., and Towsley, D. Classification of access network types. In *Proc. IEEE INFOCOM*, pp. 1060–1071.

[75] WEKA. `http://weka.sourceforge.net`.

[76] Yang, Sookhyun, Kurose, Jim, Heimlicher, Simon, and Venkataramani, Arun. Measurement and modeling study of user transitioning among networks. *IEEE INFOCOM 2015*.

[77] Yang, Sookhyun, Kurose, Jim, and Levine, Brian Neil. Disambiguation of residential wired and wireless access in a forensic setting. In *Proc. IEEE INFOCOM Mini-Conference 2013*.

[78] Yang, Sookhyun, Kurose, Jim, and Levine, Brian Neil. Measurement and modeling study of user transitioning among networks. In *Proc. IEEE INFOCOM 2015*.

[79] Yang, Sookhyun, Kurose, Jim, and Levine, Brian Neil. Disambiguation of residential wired and wireless access in a forensic setting. *Tech. Rep. UM-CS-2011-032, U. of Massachusetts Amherst* (2011).

[80] Yang, Sookhyun, Vasudevan, Sudarshan, and Kurose, Jim. Witness-based detection of forwarding misbehavior in wireless networks. In *U of Massachusetts Amherst, Department of Computer Science, Tech. Rep. UM-CS-2009-001* (2009).

[81] Yang, Sookhyun, Vasudevan, Sudarshan, and Kurose, Jim. Witness-based detection of forwarding misbehaviors in wireless networks. In *Wireless Mesh Networks (WIMESH 2010), 2010 Fifth IEEE Workshop on* (2010), IEEE, pp. 1–6.