MEASUREMENT-DRIVEN MODELING OF TRAFFIC DEMAND AND SHORT TERM FORECASTING IN LARGE WLANS

by

Elias Raftopoulos

A thesis submitted in fulfillment of the requirements
for the degree of Masters in Computer Science

University of Crete

2007

Approved by _____

Maria Papadopouli
Assistant Professor
Chairperson of Supervisory Committee

_____

Apostolos Traganitis
Professor

_____

Panagiotis Tsakalidis
Associate Professor

_____

Panos Trahanias
Professor
Chairperson of the Graduate Studies Committee

Date _____

ABSTRACT

Wireless Local Area Networks (WLANs) have seen enormous success in response to the growing demand for wireless access. Network operations, such as capacity planning, admission control, and load balancing, will become more relevant and will have to be properly engineered to match the WLAN constraints. In this context, it is of critical importance to understand the performance and workload of the wireless networks and develop wireless networks that are more robust, easier to manage and scale, and able to utilize scarce resources more efficiently. Thus, it is important to perform empirical studies to measure the phenomena of interest in real-life networks and formulate realistic models of user communication and association patterns. This can be beneficial in the administration and deployment of wireless infrastructures, protocol design for wireless applications and services, and their performance analysis. Moreover, the development of testbeds, tools, benchmarks, and models is of tremendous importance and can motivate further performance analysis and simulations.

In this work, we study a large infrastructure and model the traffic load at APs. We take advantage of the wireless infrastructure of the University of North Carolina (UNC) to draw large amounts of different types of measurement data. In the first stage of this research effort, we exploit the spatial and temporal resolution available in data traces in deriving models for traffic demand in the network. Our contributions have a strong methodological element. For example, in modeling traffic demand we adopt a hierarchical framework that is found to capture demand in various levels of spatial scales, ranging from individual buildings to groups of buildings and network-wide. Throughout this work, we make heavy use of statistical tools; clustering techniques help us address scalability concerns in traffic demand modeling. In the second stage, we focus on the AP-level and design forecasting algorithms to predict the traffic load in different time-scales. We then apply these forecasting algorithms on real traffic traces acquired from the most heavily utilized APs and evaluate their performance.

ACKNOWLEDGMENTS

This thesis is the result of a two-year working experience as graduate student at the research group for Telecommunications & Networks at the Institute of Computer Science (ICS) of the Foundation for Research and Technology Hellas (FO.R.T.H.). These years have been very instructive, giving rise to several interesting tasks more or less related to the research.

First of all I would like to thank Maria Papadopouli, my supervisor, not only for her encouragement during this work, but also for letting me proceed my own way until I needed some guidance, and for being right there with the thrusters when asked!

I want also to thank our colleagues and research collaborators, F.H. Campos, H. Shen, and M. Karaliopoulus, for their invaluable help. Without their support the completion of this thesis would not have been possible.

Thanks also to members of the NetLab Group for the friendship and energy expended in extra-curricular activities, encouragement and comic relief when things started to get crazy. Of these I want to give special thanks to M. Ploumidis, V. Lekakis, M. Pantelias, K. Flouri, M. Moudatsos, M.Spanakis, L. Kriara and G. Tzagkarakis.

This work was supported by a graduate fellowship from the ICS (FO.R.T.H.), which I also truly acknowledge for providing the necessary technical equipment.

Finally, I would like to thank my family, for providing me with the ideal environment to grow up in, and exhorting me in changing for the better. Their constant support and encouragement have really brought me here.

I have had a lot of fun. Thank you all (I hope that I've not forgotten someone).

TABLE OF CONTENTS

LIST OF FIGURES

Number *Page*

LIST OF TABLES

Number *Page*

1. Introduction

1.1 WLAN deployment

Wireless Local Area Networks (WLANs) have seen enormous success in response to the growing demand for wireless access. IEEE802.11 networks are becoming widely available in universities, corporations, and residential areas to provide wireless Internet access. Such networks are also increasingly being deployed in airports, hospitals, shopping centers, and other public areas. Popular applications and services from the wired networks shift in the wireless arena and new applications are increasingly being deployed. The proportion of wireless streaming audio and video traffic increased by 405% between 2001 and 2003/2004, P2P from 5.2% in 2001 to 19.3% in 2003/4, filesystems from 5.3% to 21.5%, and streaming from 0.9% to 4.6% between January 2006 and March 2006. Moreover, the rapid deployment of the IEEE802.11 infrastructures in various environments impacts the way users access the information and triggers new applications and services that in turn, generate a richer set of traces for analysis.

1.2 Motivation

Despite their already broad use, the evolution of WLANs towards larger, multi-service networks bears challenges that have not been thoroughly addressed by the research community. Standardization work is still ongoing in IETF (capwap group) and IEEE (802.11x groups) in an attempt to improve the management of the radio resources and provide the Quality of Service (QoS) real-time services require. Within this context network functions, such as capacity planning, access control, and load balancing, will become more relevant and will have to be properly engineered to match the WLAN constraints. For the design of adequate quality of service provision, and network monitoring mechanisms, it is important to analyze and interpret the traffic characteristics. Real-life measurement studies can be particularly beneficial in the development and analysis of such mechanisms. It is critical to understand the performance and workload of the wireless networks and develop wireless networks that are more robust, easier to manage and scale, and able to utilize scarce resources more efficiently.

1.3 Goals

The research goals of this work are twofold. In the first stage, we intend to analyze network traffic using different spatial and temporal scales in order to capture real-life phenomena, such as user communication and association patterns, and formulate realistic models of wireless traffic demand in a scalable manner that will be useful to network administrators, who wish perform capacity planning and network provisioning. Moreover, our models can be used to replace existing models which are widely used by researchers in simulation studies, benchmarks, and testbeds. The assumptions often made in these models fail to capture phenomena encountered in large wireless infrastructures and, therefore, can lead to unrealistic results.

In the second stage, we shall focus on traffic analysis in a very fine resolution, namely AP-level traffic modeling, and attempt to exploit the traffic characteristics and the diverse sources of data acquired to monitor different aspects of the network dynamics, in order to tailor simple short-term forecasting algorithms. These algorithms can be incorporated in access points to allow them to perform admission control and capacity planning in a real-time fashion. So, in this context, each AP predicts its traffic load for the next time interval (e.g., next hour or five minute interval) and uses its traffic load forecasts during admission control to not only better manage its traffic demand but also advice clients to associate with the appropriate APs to better utilize their local resources. Such predictions can be used to reduce the energy spending at the client side, improve the capacity utilization of wireless LANs, and better load balance the traffic. To the best of our knowledge, these research efforts are the first traffic forecasting studies on large IEEE802.11 infrastructures.

1.4 Challenges

Wireless network measurements feature higher complexity and, as a result, are less abundant than those in wired networks. Depending on how detailed view of the network is required at the spatial dimension, e.g., at an AP, APs co-located in a building or set of buildings, and the architecture of the network (single link-level subnet or multiple subnets), one needs to capture traffic at multiple physical locations. As IEEE 802.11 MAC-layer frame sniffers are not commonly available, researchers often have to build

custom equipment or resort to expensive commercial tools to capture the over-the-air traffic with the required level of detail. It comes as no surprise that only recently have traces from large-scale wireless infrastructures with statistically significant network usage been made available. Notably, the majority of the measurement studies, e.g., [4, 5, 6], make high-level observations about network dynamics in both the temporal and spatial domains without getting into the detail that modeling tasks require.

1.5 Related work

Empirical and performance analysis studies indicate dramatically low performance of real-time constrained applications over wireless LANs [1], and large handoff delays [2, 3]. The overhead of scanning for nearby APs is routinely over 250 ms, far longer than what can be tolerated by highly interactive applications, such as voice telephony. Mobile users experience frequent loss of connectivity and high end-to-end delays when they access the wireless Internet. While in several cases over-provisioning in wired networks is acceptable, it can become problematic in the wireless domain due to interference, regulatory, and environmental reasons. Furthermore, wireless clients have more vulnerabilities than their wired counterparts. Their energy limitations, dynamic characteristics, and mobility of the wireless clients impose additional constraints and the bandwidth utilization at an AP can impact their performance substantially.

Recently, there have been several empirical-based measurement studies about the traffic load [7, 8, 9, 10, 11, 12, 13], user access [14, 9, 15, 16, 17, 18, 19, 20], handoff [2, 3], delay and packet losses in the MAC [21] and TCP connections [22], link quality and routing [23, 24, 25]. Measurements on IEEE802.11-based mesh networks have also received a lot of attention [25, 26, 24, 27, 28, 39, 30, 31, 32, 33, 29]. However, while there is a rich literature characterizing traffic in wired networks [34, 35, 36], there are only a few studies available that examined wireless traffic load and even fewer studies on short-term wireless traffic forecasting.

1.6 Contributions

In this work, we study a large wireless infrastructure and model the traffic load in different levels of spatial aggregation, spanning from single access

points to the entire network. We take advantage of the large wireless infrastructure of UNC to draw large amounts of different types of measurement data. We then exploit both the spatial and temporal resolution available in data traces to derive models for traffic demand. Our contributions have a strong methodological element. For example, in modeling traffic demand we adopt a hierarchical framework that is found to capture demand in various levels of spatial scales, ranging from individual buildings to groups of buildings and network-wide. Throughout this work, we make heavy use of statistical tools; clustering techniques help us address scalability concerns in traffic demand modeling.

Shifting to a finer level, the AP-level, we design forecasting algorithms to predict the traffic load in different time-scales and apply these forecasting algorithms on real traffic traces acquired from the most heavily utilized APs to evaluate their performance. We show that hourly predictions of the traffic load at an AP have very large prediction error due to the high variability. We also focus on finer time scales and (a) present and evaluate a number of novel forecasting algorithms, such as the adaptive moving-average and flow-based algorithms, (b) integrate different types of information in the prediction algorithms (e.g., snmp and tcp-based), (c) observe dramatically improvement in the forecasting accuracy. We show that the time granularity and recent traffic history have dominant impact on the prediction accuracy. That is, the finer the time granularity (5-minute compared to hourly intervals) and more recent the historical traffic data is, the larger their impact on the prediction error.

1.7 Roadmap

In Section 2 we present the infrastructure studied, define the main entities of our simulation study, namely the wireless session and the network flow, introduce the main mechanisms used to capture network data, and present the methodology used in order to recover missing values, which can found in our data due to hardware failures, transient phenomena, or protocol characteristics. In Section 3 we present our modeling results, in terms of flow and session related characteristics, and discuss our hierarchical framework, which aims to capture traffic characteristics in different levels of spatial aggregation. In Section 4 we evaluate the different modeling alternatives and compare how the spatial detail affects the model's

capability to capture the traffic demand. Sections 3 and 4 are heavily based on the paper "**On scalable measurement-driven modelling of traffic demand in large Wlans**.", (published in the *IEEE Workshop on Local and Metropolitan Area Networks 2007*), by Merkouris Karaliopoulos, Maria Papadopouli, Elias Raftopoulos and Haipeng Shen. In Section 5 we shift to the AP-level and analyze traffic in finer time scales. Based on this analysis, we formulate short-term traffic load forecasting algorithms, using diverse sources of data, in an attempt to capture the multiple network dynamics. We evaluate these algorithms in the most heavily used access points and discuss our major findings. This Section is based on the following two publications, "**Evaluation of short-term traffic forecasting algorithms in wireless networks.",** (published in the *2nd Conference on Next Generation Internet Design and Engineering*), by Maria Papadopouli, Elias Raftopoulos, and Haipeng Shen, and "**Short-term traffic forecasting in a campus-wide wireless network",** (published in the *16th Annual IEEE International Symposium on Personal Indoor and Mobile Radio Communications)*, by Maria Papadopouli, Haipeng Shen, Elias Raftopoulos, Manolis Ploumidis, and Felix Hernandez-Campos. Finally, we present our main contributions and draw our conclusions.

2. Network infrastructure and measurement data

2.1 Introduction

In this section we present the campus network studied and describe the different methods used to capture user interaction with the wireless infrastructure. Users are mobile and often run applications with real time constraints whereas at the same time they roam seamlessly through different basic service sets (BSSs). This type of activity is shown in Figure 2.1.1 where we can see a typical user pattern. User B associates with AP1, he roams to AP2 and AP3 where he terminates his interaction and disconnects. Three types of data are used to capture user interaction, namely, syslog data, snmp data, and flow-related data. These types of information are studied thoroughly in the following sections.

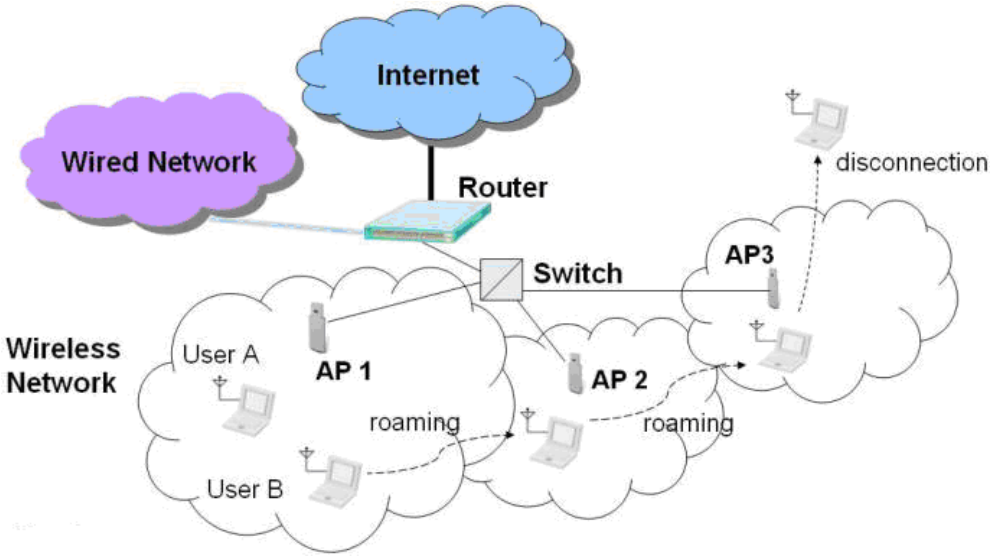

Figure 2.1.1. User interaction with the wireless infrastructure.

2.2 UNC campus

The measurement data for our project are drawn from the UNC campus wireless network. UNC began the deployment of its wireless infrastructure in 1999, providing coverage for nearly every building in the 729-acre campus, encompassing a diverse academic environment which includes university departments, programs, administration, activities, and residential

buildings. During the project, the network has seen substantial growth; the 488 APs by October 2004 had become 574 in April 2005 and 741 by May 2006, making it one of the largest of its kind. Almost all of them belong to the Cisco Aironet series [37]. The network APs are spread over more than 240 buildings in the 729-acre campus, including student residence halls, academic buildings, sport halls, and libraries, and a few off-campus administrative offices. They provide wireless access to 26,000 students, 3,000 faculty and 9,000 staff members. Of the 26,000 students, 61% are undergraduates and more than 75% of them own a laptop with wireless access capabilities. Such laptops and other devices allowing the campus population to interact with the wireless network are called clients. Wireless clients arrive at the network, associate to one or more APs for some period of time, and leave the infrastructure. Table 1 shows the evolution of the wireless infrastructure and the significant increase of APs and wireless clients.

Tracing period	Clients	APs
February 10 –April 27, 2003	7,694	232
17-24, October 2004	8,880	459
2-9, March 2005	9,049	532
13-20, April 2005	9,881	574
September29 –November,2005	14,712	574

Table 1. The evolution of the wireless infrastructure in terms of number of APs and clients.

Information access can be achieved based on this infrastructure. The most integral part of the wireless architecture is the access aoint. A wireless access point (AP) (or base station) is a dual-homed device, a gateway with a radio transmitter and receiver that provides Internet access to hosts in its wireless range. A typical AP is connected to a wired network and can relay data between devices on each side. Within the range of the AP, the wireless end-user has a full network connection with the benefit of mobility. Many APs can be connected together to create larger networks that allows "roaming" between them; APs relay packets between each other, so that a packet can be delivered to its final destination, a roaming client. In

contrast, in ad hoc networks, devices operate in a self-organizing, autonomous manner.

2.3 Data collection

Three types of data have been used to track the interaction of clients with the wireless network infrastructure: Syslog messages, Simple Network Management Protocol (SNMP) data, and packet header traces. The first two types of data have been collected almost continuously from the wireless network, whereas the packet header traces come from two different eight-day monitoring periods spaced one year apart, i.e., April 13-20 2005 and Apr 28-May 5 2006.

A. Syslog data

Syslog is a protocol that allows a machine to send event notification messages across IP networks to event message collectors – also known as Syslog Servers or Syslog Daemons. In other words, a machine or a device can be configured in such a way that it generates a Syslog Message and forwards it to a specific Syslog Daemon (Server). Syslog messages are based on the User Datagram Protocol (UDP) type of Internet Protocol (IP) communications. Syslog messages are received on UDP port 514. Syslog message text is generally no more than 1024 bytes in length. Since the UDP type of communication is connectionless, the sending or receiving host has no knowledge receipt for retransmission. If a UDP packet gets lost due to congestion on the network or due to resource unavailability, it will simply get lost. Syslog is typically used for computer system management and security auditing. While it has a number of shortcomings, syslog is supported by a wide variety of devices and receivers across multiple platforms. Because of this, syslog can be used to integrate log data from many different types of systems into a central repository. Syslog is now standardized within the Syslog working group of the IETF.

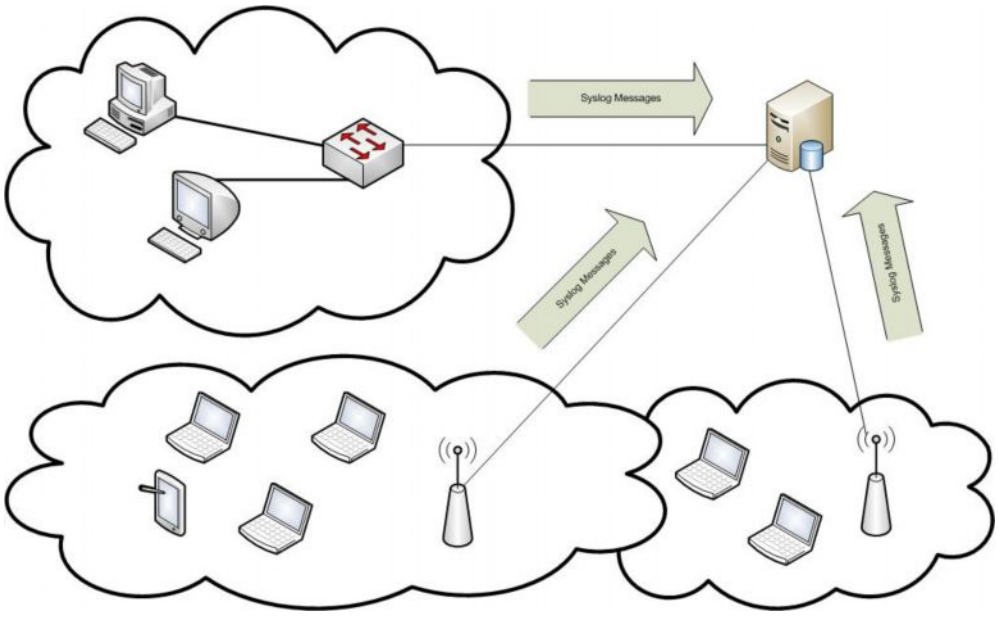

Figure 2.3.1. Syslog data polling procedure.

Syslog messages are event-based. They are triggered by different types of events at the IEEE 802.11x MAC layer, including the (re)association/disassociation of a client with/from an AP, its authentication/deauthentication with/from the network, and a client connection reset. All network APs are configured to report syslog events to a server, which is continuously operational. The majority of APs on campus were configured to send trace data via syslog messages to a syslog server in the computer science department.

There are seven types of events that trigger an AP to transmit a syslog message. These messages and their corresponding events are interpreted as follows:

Authenticated: A card must authenticate itself before using the network. Since a card still has to associate with an AP before sending and receiving data, we ignore any authenticated messages.

Associated: After it authenticates itself, a card associates with an AP. Any data transmitted to and from the network is transmitted by the AP.

Reassociated: A card may reassociate itself with a new AP (usually due to higher signal strength) or the current AP. After a reassociation with an AP, any data transmitted to and from the network is transmitted by that AP.

Roamed: After a reassociation occurs, the old AP and sometimes the AP with which the card has just reassociated send a roamed message. Since we still receive the reassociated message, we can ignore this message as well.

Reset: When a card's connection is reset, a reset message is sent. In our trace, cards with a reset message are only involved in reset messages. We believe this to be an artifact of us not having logs from all of the APs, and therefore ignore any reset messages.

Dissasociated: When a card wishes to disconnect from the AP, it disassociates itself. We ignore any disassociated messages for a card if the previous message for that card was a disassociated or deauthenticated message.

Deauthenticated: When a card is no longer part of the network a deauthenticated message is sent. It is not unusual to see repeated deauthenticated messages for the same card, with no other type of events for that card in between. We ignore any deauthenticated messages for a card if the previous message for that card was a disassociated or a deauthenticated message. A disconnection message describes either a disassociated or deauthenticated message.

B) SNMP data

SNMP is an Internet-standard protocol for managing devices on IP networks. Many kinds of devices support SNMP, including routers, switches, servers, workstations, printers, modem racks, and uninterruptible power supplies (UPSs). It is an application layer protocol that facilitates the exchange of management information between network devices and is part of the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite. SNMP enables network administrators to manage network performance, find and solve network problems, and plan for network growth. SNMP forms part of the internet protocol suite as defined by the Internet Engineering Task Force (IETF) and is used by network management systems to monitor network-attached devices for conditions that warrant administrative attention. It consists of a set of standards for network management, including an Application Layer protocol, a database schema, and a set of data objects. SNMP exposes management data in the form of variables on the managed systems, which describe the system configuration. These variables can then be queried (and sometimes set) by managing applications.

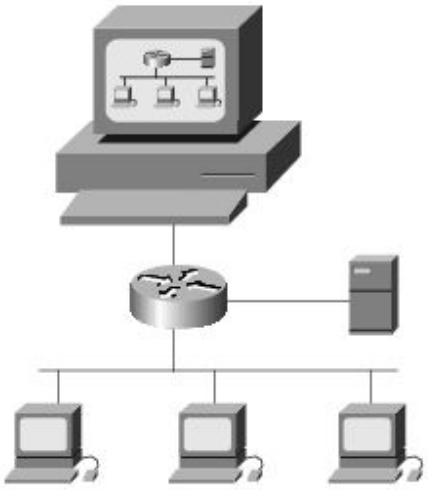

Figure 2.3.2. SNMP facilitates the exchange of network information between devices.

An SNMP-managed network consists of three key components: managed devices, agents, and network-management systems (NMSs). A managed device is a network node that contains an SNMP agent and that resides on a managed network. Managed devices collect and store management information and make this information available to NMSs using SNMP. Managed devices, sometimes called network elements, can be routers and access servers, switches and bridges, hubs, computer hosts, or printers. An agent is a network-management software module that resides in a managed device. An agent has local knowledge of management information and translates that information into a form compatible with SNMP. An NMS executes applications that monitor and control managed devices. NMSs provide the bulk of the processing and memory resources required for network management. One or more NMSs must exist on any managed network.

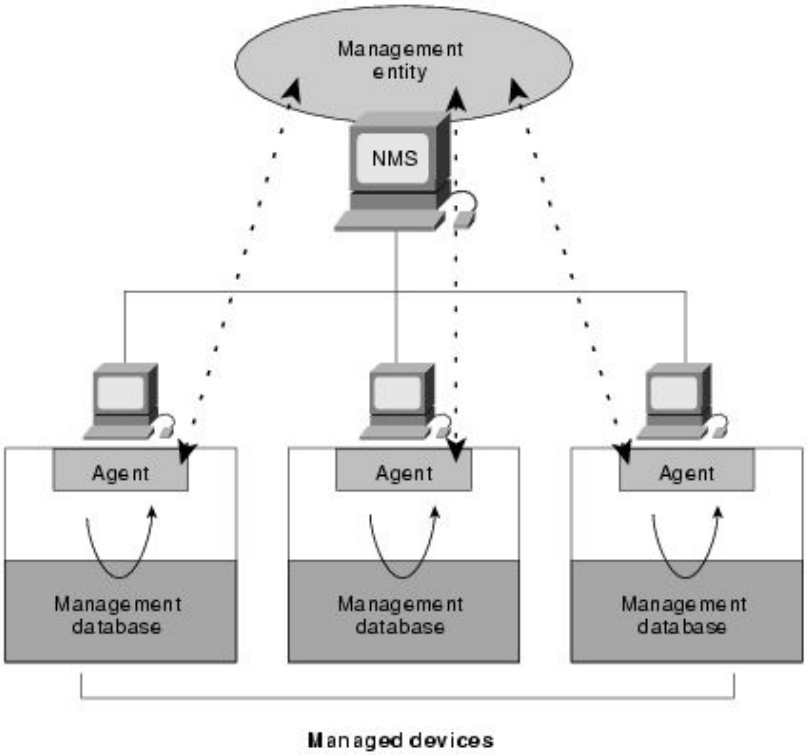

Figure 2.3.3. A SNMP-Managed Network Consists of Managed Devices, Agents, and NMSs.

In our infrastructure SNMP data become available periodically from each AP. We implemented a custom SNMP-polling system relying on a non-blocking SNMP library. APs are polled independently with a period of five minutes, so that delays incurring during the processing of SNMP polls by the slower APs do not affect the other APs. This eliminates any extra delays due to the slow processing of SNMP polls by some of the slower APs. The system ran in a multiprocessor system and the CPU utilization in each of the three processors we employed never exceeded 70%. The collection of SNMP data is a 24/7 process, which has been running almost without interruption since September 2004.

C) Packet header data

The bulk of the campus wireless network has a single aggregation point that connects to a gateway router. This router provides connectivity between the wireless network and the wired links, including all of the campus computing infrastructure and the Internet. Packet header traces

were collected with a high-precision monitoring card (Endace 4.3GE). The card was installed in a high-end FreeBSD server and captured all packets traversing the link between UNC and the Internet in both directions. The monitoring period was 178.2 hours in 2005 and 192 hours in 2006, yielding 175GB and 365GB of packet headers, respectively. During that period, the campus used primarily Cisco Aironet 350 802.11 APs, although some areas on campus are serviced by older APs from other manufacturers. As the syslog traces indicated, the infrastructure was accessed by 7,694 distinct wireless clients. The 37% of them made one or more HTTP requests during the tracing period February 26 through March 24, 2003. The sharp increase in the trace size indicates the significant growth of the wireless demand between these periods.

The HTTP traces were based on packet headers collected from the FreeBSD monitoring system described in the previous section. The tracing tool tcp-dump was employed to collect all TCP packets that have payloads that begin with the ASCII string "GET" followed by a space. The full frame was collected as a potential HTTP request. We did not restrict our collection to the standard HTTP port, allowing us to record HTTP requests sent to servers on non-standard ports, which include many common peer-to-peer file-sharing applications. The packet trace was then processed to extract the HTTP GET requests contained therein. From each packet, we kept the time of the packet's receipt (with one-second resolution), the hostname specified in the request's Host header, the Request-URI, and the hardware MAC address of the wireless IEEE 802.11 client. If all of these items were not available in a packet, we did not include the recorded packet in our recorded requests. Using these criteria, 8,358,048 requests for 2,437,736 unique URLs were traced and included in the analysis. By recording the traffic before it had passed through an IP router, we were able to capture the original MAC header as generated by the IEEE 802.11 clients for transmission to the gateway router.

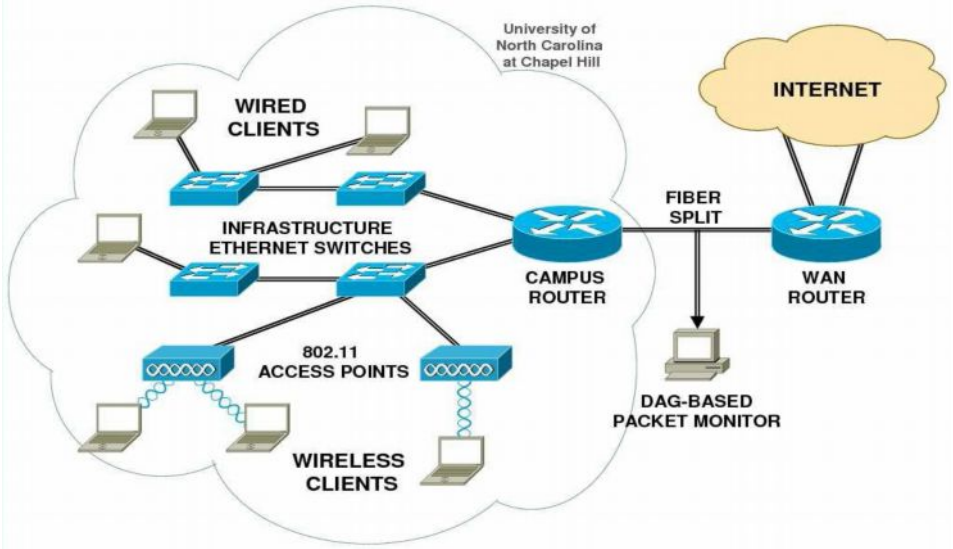

Figure 2.3.4. Campus-wide wireless infrastructure and packet monitor tool.

Our measurement data are summarized in Table 2. Each type of measurement data provides insight to different aspects of the clients' interaction with the network and proves relevant to different modeling tasks. In the following sections, we describe the use of traces in modeling the network traffic demand, and the way clients move in the network Moreover, we discuss their advantages and weaknesses in the respective context. In separate experiments, we have also captured intra-WLAN traffic from a finite number of APs. Doing this throughout the network would demand monitoring in many physical locations, which is impractical. In results presented subsequently packet header traces are drawn from the UNC-Internet link.

Data traces	Monitoring period	APs seen	Clients seen	Size	Modeling task
Packet headers	Apr 13-20, 2005 Apr 28-May 5 2006	N/A	9777 1248	175GB 365GB	Traffic demand
SNMP pollings	Sep 29, 2004-Jun 30, 2006	488-741		27GB	Traffic demand
Syslog messages	Jan 20, 2005-Mar 3, 2006 Apr 3-Jun 30, 2006	488-640 640-741		3GB 630MB	Client roaming

Table 2. Analyzed measurement datasets from UNC campus and their use in modeling tasks.

3. Modeling Methodology

3.1 Introduction

In this section we process the large amount of different measurement data we have acquired from the wireless infrastructure of UNC. We then take advantage of the spatial and temporal resolution available in data traces in deriving models for traffic demand of users in this network The main contribution is that, in modeling traffic demand we adopt a hierarchical framework that is found to capture demand in various levels of spatial scales, ranging from individual buildings to groups of buildings and network-wide. Throughout this section, we make extended use of statistical tools; clustering techniques are used in order to address scalability concerns in traffic demand modeling.

3.2 Two-tier modeling approach

The modeling approach proposed in this work, is hierarchical in that it organizes the activity of network clients into wireless sessions and network flows. The wireless session can be viewed as an episode in the interaction of a client and the wireless infrastructure: a wireless client arrives at the network, associates to one or more APs for some period of time, and then leaves the infrastructure. It was preferred over modeling individual association-disassociation sequences, whose dynamics in wireless LANs can change dramatically due to small changes in the network layout, physical environment, or network/client equipment. Likewise, at lower level we work with network flows, such as TCP connections and UDP conversations, rather than with packets. Flows include well-separated collections of packets exchanged between a pair of Internet hosts, i.e., packets that share the same transport-layer "5-tuple".

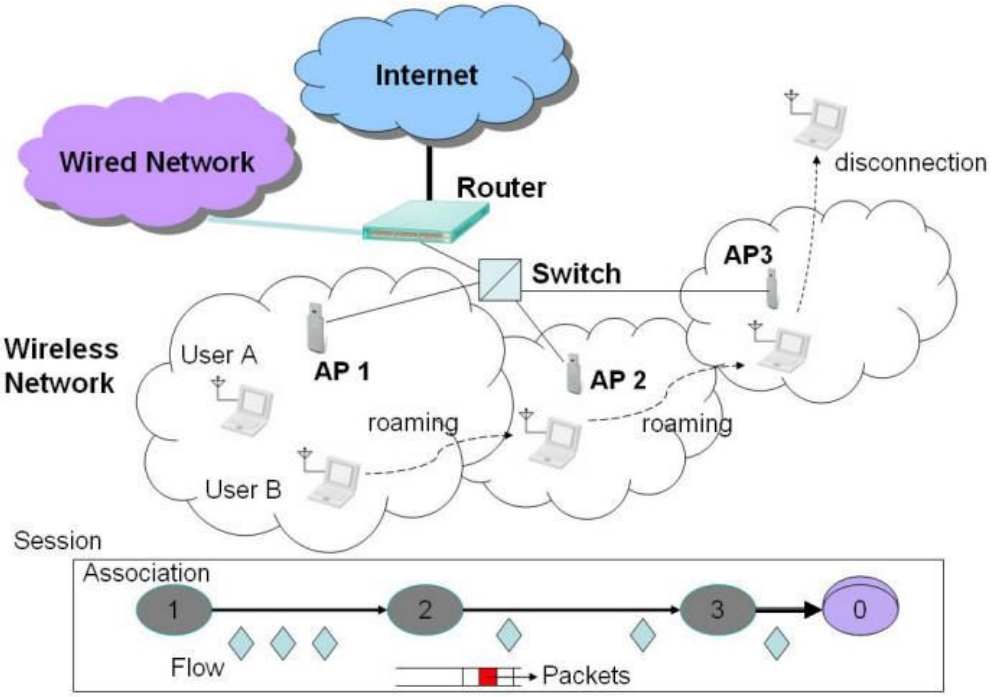

Figure 3.2.1. Sessions and flows.

Arguments related to scalability and reusability, which are particularly desirable properties in modeling, complicate the problem. Previous modeling studies have either attempted to model traffic demand over hourly intervals at the level of individual APs [38] or studied the problem at system-level deriving models for the aggregate network-wide traffic demand [39]. Clearly, both approaches have their strong and weak points. The second approach results in datasets that are amenable to statistical analysis and provides a concise summary of the traffic demand at system-level. However, it fails to capture the variation of this demand at finer spatial detail that may be required in the evaluation of system functions with focus on the AP-level (e.g., load balancing). While working at the AP-level achieves that, it fails in other respects: the approach does not scale for large wireless infrastructures since a large number of distributions has to be derived and simulated grows linearly with the number of APs, which is not desirable when studying large-scale wireless infrastructures, and data do not always lend to statistical analysis. Moreover, the modeling results are largely sensitive on the specific AP layout of a particular network and short-term variations of the radio propagation conditions.

So in the spatial domain we decided to work with buildings rather than APs. We view buildings as more robust entities for modeling the spatial variation of traffic demand. In fact, we could draw an analogy between flows-packets and buildings-APs. Much as packet-level dynamics are subject to network topology and instantaneous conditions, AP-level user activity is sensitive to radio propagation dynamics and environmental setting. One good example is the "ping-pong" effect, where a stationary user may be alternately associated with two, or even more, APs due to short-term radio signal propagation variations.

The notable advantage of working with buildings is that many of our findings for the aggregate network traffic demand also hold for the per-building traffic demand. Session arrivals, for example, can still be modeled by time-varying Poisson processes, as reported in [40]. Moreover, buildings of the same type are found to share largely similar patterns in their hourly session arrival rates, despite discrepancies in volume. Likewise, the distributions listed in Table 3, BiPareto for number of flows per session and flow sizes and Lognormal for flow interarrivals within session, hold not only at the building level but also at intermediate levels of spatial aggregation, such as over all buildings of the same building type. Furthermore, at building level client sessions appear more stationary than when considering APs. Building-roaming sessions, during which a WLAN client visits more than one building, account for less than 10% of the overall sessions. Therefore, simulating the traffic workload consists of simulating sessions and the flows started inside them, leaving packet-level and association dynamics to underlying mechanisms that are independent of our model.

Modeled variable	Model	Probability Density Function (PDF)	Parameters
Session arrival	Time-varying Poisson with rate	N: #sessions between t_1 and t_2 $\lambda = \int_{t1}^{t2} \lambda(t)dt, \Pr(N=n) = \frac{e^{-\lambda}\lambda^n}{n!}, n = 0,1,\ldots$	Hourly rate: 44(min), 1132(max), 294(median)
Flow interarrival/session	Lognormal	$p(x) = \frac{1}{\sqrt{2\pi}x\sigma}\exp\left[-\frac{(\ln x-\mu)^2}{2\sigma^2}\right]$	$\mu = -1.3674, \sigma = 2.785$
Flow number/session	BiPareto	$p(x) = k^{\beta}(1+c)^{\beta-\alpha}x^{-(\alpha+1)}(x+kc)^{\alpha-\beta-1}$ $(\beta x + \alpha kc), x \geq k$	$\alpha = 0.06, \beta = 1.72,$ $c = 284.79, k = 1$
Flow size	BiPareto	Same as above	$\alpha = 0.00, \beta = 0.91,$ $c = 5.20, k = 179$

Table 3. Summary of models for network-wide traffic demand variable.

The actual processes/variables we model are four, namely session arrival, flow inter-arrival, flow number and flow size. In the following sections, one of the questions we address is what level of modeling the traffic demand variation in the spatial dimension yields the best trade-off between modeling efficiency and scalability. Generally, for each traffic variable listed in Table 3, the spatial detail of modeling could be the building, building type, or in the extreme case the network as a whole. Although the building type is an intuitive basis for grouping buildings, it is not the best one. In fact, in the following section, we use clustering techniques to come up with alternative groupings of buildings of higher utility for our modeling task.

Building type	Academic	Administrative	Athletic	Business	Clinical	Library	Residential	Social
Number	51	25	17	8	18	4	110	10

Table 4. Summary of building types.

Parametric models describe the traffic demand variables. When compared with empirical models, they provide better insight to the properties and dynamics of the modeled quantities. In parallel, they are more adequate in summarizing datasets and make their comparison straightforward. We then use formal and visual statistical analysis methods and tools, such as goodness-of-fit tests and quantile plots with simulation envelopes, to find the distributions best matching the modeled traffic variables. Since traffic

demand in large WLANs varies changes with time and in space, our modeling has considered various scales in both dimensions. The spatio-temporal variation actually manifests itself, though to a different extent, in all four traffic variables we have chosen in our two-level modeling approach. There are two notable results in this respect.

Firstly, we have found out that the same statistical distributions, though with different parameter sets, model our traffic variables during both the 2005 and 2006 monitoring periods. A time-varying Poisson process with constant rate over intervals of an hour captures the non-stationarity of session arrivals. The custom statistical test for testing the hypothesis of the time-varying Poisson process is described in [41]. The per-session number of flows and the flow sizes are well modeled by the BiPareto distribution [42]. BiPareto distributions have Pareto tails on both ends; since on a log-log plot a Pareto tail of the form x-a appears as a straight line with slope –a, the log-log plot of the BiPareto survival function results in two nearly linear regimes. This property of the distribution favors it over other tested distributions, including Weibull, Gamma, and Pareto, for the modeling of the per-session number of flows and flow sizes [43]. Likewise, the Lognormal distribution gives the best fit for the in-session flow interarrival durations.

Secondly, the same set of distributions has been found to match the modeled variables over different spatial scales. Whether we look at traffic in individual buildings or groups of buildings of the same type (e.g., Social, Residential, Academic), or all over the wireless network, the same four statistical distributions, each time with different parameter sets, match very well the variables we model. Table 5 lists the parameters of the distributions for two representative campus buildings with high demand and some sets of buildings with the same usage. The parameters for more buildings and the multi-scale analysis of the 2006 dataset are available at [44], whereas the distributions for the network-wide traffic during the April 2005 monitoring period are detailed in [43].

Modeled variable	Hourly session arrivals Poisson $\lambda(t)$, $\lambda(t) \in [\lambda_l, \lambda_h]$	Per-session flow number BiPareto(a,β,c,k)	Per-session flow interarrivals Lognormal(μ, σ)	Flow size BiPareto(a,β,c,k)
Network (2005)	[44,1132]	(0.06,1.72,284.79,1)	(-1.37,2.79)	(0,0.91,5.20,179)
Network (2006)	[75, 1171]	(0.09,1.49,585.4,1)	(-1.49,2.92)	(0,1.03,18.41,152)
Academic bldgs	[0, 421]	(0.11,2.17,713.85,1)	(-1.64,2.99)	(0,1.08,23.64,152)
Library bldgs	[0, 123]	(0.06,2.30,846.67,1)	(-1.74,2.98)	(0,1.18,24.94,154)
Social bldgs	[0, 139]	(0.00,0.08,718.69,1)	(-1.82,2.97)	(0,1.18,21.81,179)
McColl bldg	[0, 217]	(0.09,2.69,1026.37,1)	(-1.69,3.01)	(0,1.07,23.87,152)
H. James bldg	[0, 27]	(0.08,1.95,1357.35,1)	(-1.62,2.99)	(0,1.03,13.55,182)

Table 5. Summary model parameters per building type.

The methodological choices made attempt to strike a good trade-off between the two extreme approaches to traffic modeling that were outlined earlier, namely AP-level compared to network-level modeling. Not only do we model traffic workload in terms of sessions and flows, but we also look in more detail into the spatial dimension, using buildings as basic entities of traffic demand modeling. Major features of user activity, such as the traffic patterns they generate and their mobility within the wireless network, are studied at the building level. Heuristic and more formal clustering techniques are then applied to the data to group together buildings with similar traffic characteristics and achieve the scalability objective in our modeling.

Considerable effort is devoted to the validation of the modeling methodology. Synthesizing traffic based on the derived models and comparing with the trace data, helps us assess the reusability of system-wide models to smaller spatial scales, the accuracy-scalability trade off and the possible contributions of clustering techniques to its resolution. Moreover, the availability of datasets from two different monitoring periods, spaced one year apart, lets us identify modeling elements that are time-persistent.

The contributions in the modeling aspect of this section are summarized in the following:

- A hierarchical framework for modeling traffic workload both system-wide and at finer spatial scales (i.e., at building level and over

groups of buildings). We find that the same set of parametric distributions models our session- and flow-related traffic variables at various spatial scales and over two different monitoring periods.

- A novel methodology for scalable modeling of the spatial variation of traffic demand in large wireless networks drawing on heuristic and statistical clustering techniques.

Validation of our modeling approach assessing the model accuracy and scalability. Our results suggest that the two approaches are complementary in that they result in clusters with high purity in different traffic variables.

3.3 Enhancing scalability with clustering

Working at system-level presents two extreme cases in modeling traffic demand. The first approach gives a good insight to the user activity patterns and forms a valuable input for the network design and dimensioning. However, the averaging it introduces is not desirable for many system functions working at smaller spatial scales. For example, load balancing algorithms usually consider the traffic load of a set of APs that are in close proximity when making their decisions. Evaluation of these functions requires traffic demand models of finer detail in the spatial dimension. On the other hand, AP-level modeling, whilst giving the maximum possible accuracy, has the caveats and does not scale well. The number of distributions that have to be derived and simulated grows linearly with the number of APs, which is not desirable when studying large-scale wireless infrastructures. Furthermore, for many APs the available amounts of data are not adequate to allow statistical analysis and derivation of meaningful modeling results.

One way to resolve the trade-off between the two extreme approaches, as mentioned in previous sections, is to consider buildings. The use of buildings reduces complexity from 0(N) to 0(M), where N is the number of network APs and M is the number of buildings. However, this is often not adequate. For example, the UNC campus hosts more than 200 buildings. Therefore, we consider clustering as a way to group buildings and reduce the number of models we have to implement and use in the simulator. Clustering may be carried out with respect to one or more of the four traffic variables we consider in our model; its ultimate aim is to use the same cluster-level set of variables for a group of buildings instead of a

separate one for each building. Our hierarchical modeling framework evolves around individual buildings at the finest and the entire system at the coarsest detail. As mentioned in the introduction, both approaches have weaknesses. We address them by enhancing this framework with an intermediate level of detail, namely clusters of buildings that exhibit similar behavior with respect to the traffic variables we model.

Moreover, the main modeled entities in our two-level model are sessions, where the traffic non-stationarity is captured via the time-varying Poisson processes. Most of the buildings exhibit distinct session arrival characteristics during specific hours of the day. It seems like a natural selection to use the mean hourly session arrival timeslots as the input variable. Hence, we apply clustering techniques at the session level with the aim to come up with groups of buildings featuring similar variation of session arrivals in time. The ultimate aim of our building clustering is to use the same cluster-level hourly session arrival rate time series, hereafter called cluster profile or signature, for a group of buildings instead of a separate time series for each building, the building profile. Therefore, our clustering needs to take into account the size displacement between different building profiles. This requirement cannot be satisfied if we consider heuristic ways to group buildings, e.g., the building type as defined in Section 3.3, which result in building groups with high similarity in shape but large size displacements. To get clusters of buildings with the desired properties we combine clustering with dimension reduction techniques.

Clustering is a division of data into groups of similar objects. Each group, called cluster, consists of objects that exhibit similarity between themselves and dissimilarity to objects of other groups. Data representation based on fewer clusters is a coarse-grained approach and it is unavoidable to lose some high resolution details, but simplification is achieved making modeling a much easier task. However, the accuracy and reliability of a classification method will not be efficient if highly correlated variables are included in the analyzed dataset. The dimensionality of the underlying model in our data is the number of independent variables used. One important aspect of the data pre-processing before applying the actual clustering methodology, is to reduce dimensionality sacrificing the lowest possible accuracy. Principal components analysis is a quantitatively rigorous method for achieving this goal. The method involves a mathematical

procedure that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components. The objective is to reduce the dimensionality of the dataset but retain most of the original variability in the data. All the principal components, which are a linear combination of the original variables, are orthogonal to each other, so there is no redundant information. These components form an orthogonal basis, which constitutes the space of the data where the clustering procedure will be performed in the following steps. The first component accounts for the largest possible amount of variance. The second component, formed from the variance remaining after that associated with the first component has been extracted, accounts for the second largest amount of variance, etc. Therefore, the aim is to reduce the dimensionality of the original data, whilst preserving most of the variation in data in the first few Principal Components(PCs). Geometrically PCs can be viewed as dimensions in a reduced space where each dimension is perpendicular to each other. In our case the input variables are the hourly session arrivals.

Hence, we apply clustering techniques at the session level with the aim to come up with groups of buildings featuring similar variation of session arrivals in time. Specifically, we work with the 2006 trace resulting in a time series of 192 hourly session arrival rates for each building. The integration of time in the original variables will be revealing in the case of temporal phenomena, since our model will be able to capture characteristics related to the human behavior which are translated to periodic patterns in the data. Moreover, clustering based on a single metric, namely the mean session arrival rate, will produce results that are easier to examine and interpret. The time series are the features or attributes of the data matrix X input to clustering. The matrix has 250 rows, one for each campus building. Whereas the clustering algorithm is the same in all cases, we consider three alternatives for reducing the dimensionality of our dataset and bring out the size element. They rely on Principal Component Analysis (PCA) and Singular Value Decomposition (SVD).

A. Data dimensionality reduction via PCA and SVD

In the first one, centering is applied to the data matrix by subtracting the column means of *X*, which correspond to the average building session

arrival rates. The mean subtracted from each dimension is the average across each dimension producing a dataset whose mean is zero. No scaling is performed at this point since we want to preserve the different scales among different buildings. Even if the session arrival pattern found in two different buildings is the same, we would expect these two buildings to fall into different groups if the session arrival rate's scale is significantly different. This effect would be ignored if the data were scaled preserving only the trend. We then perform PCA. The full set of principal components produced by principle components analysis, is as large as the original set of variables. However, if we keep the lower order principal components and ignore the higher order ones we keep the most important aspects of the data while discarding redundant information, thus performing a dimension reduction. To decide how many PCs to extract, we employ the scree plot, which plots the percentage of variance in data that is explained by PC *i*.

The other two alternatives do not apply any centering on the data matrix. In the first approach, we take the original data matrix X and standardize the series for each building by dividing over the scale factor, calculated as the square-root of the sum of squares of session arrivals over the 192 hours. We then apply SVD to the standardized matrix to obtain a number of left PCs equal to the one that came out of the PCA technique described earlier. The scale factor vector is then used as an additional dimension, in addition to the PCs that came out of SVD, to be passed to the actual clustering. The second alternative is similar to the first one, only now the additional dimension is the vector of average numbers of session arrivals over the whole tracing period for the 250 buildings.

In the rest of the section we present in more detail the clustering results for the PCA-based clustering. We then give the results that came out of the two SVD-based clustering alternatives and compare them on the basis of validity indices assessing the compactness of clusters and their degree of separation.

B) Building Clustering

The first thing to do before proceeding with the actual clustering is the determination of the number of PCs to use. Figure 3.2.1 shows the scree

plot for the first seven PCs. The decision of when to stop extracting principle components basically depends on when there is only very little variability left. The nature of this decision is arbitrary, however, a reasonable approach is to locate a "knee" in the plot and discard all the components that are above that knee. The knee in this case, is located at PC3 indicating that only the first three components can adequately capture most of the variability in our dataset (approximately 90%).

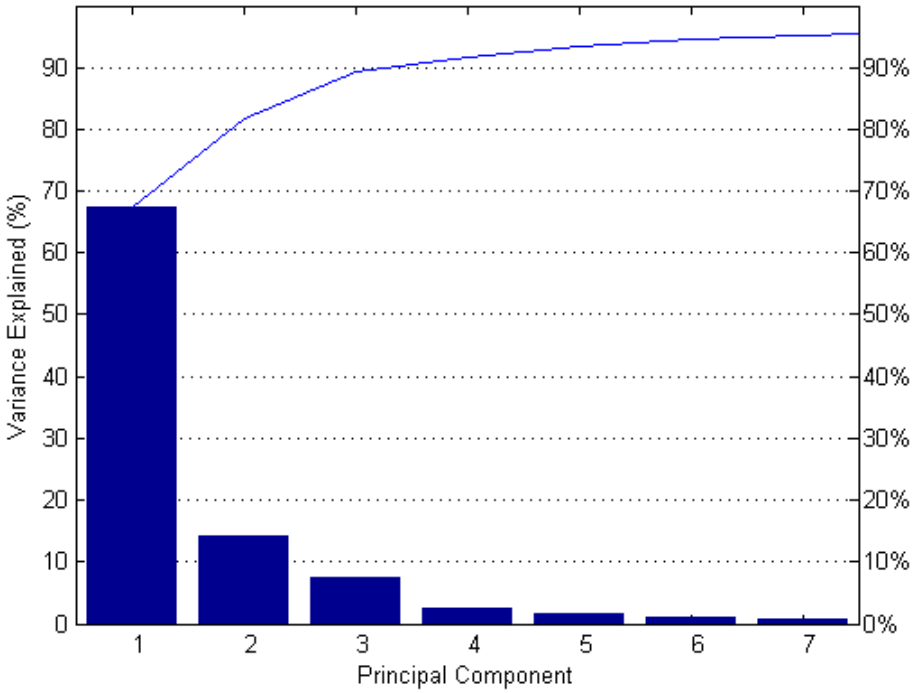

Figure 3.3.1. Percentage of variance explained by principle components.

The plots of the first three loading vectors in Figure 3.3.2 suggest that PC1 is highly correlated with the mean session arrival rate, PC2 captures the difference between day (6am-7pm) and night, whereas PC3 expresses the difference between the first 12 hours of day (12pm -11am) and the last 12 hours of day.

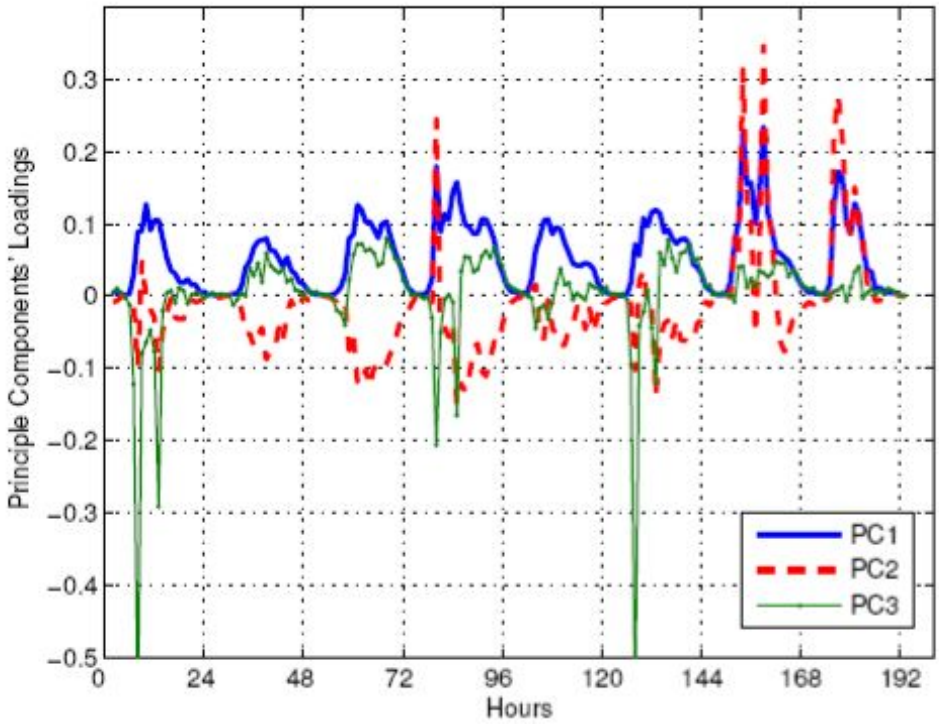

Figure 3.3.2. Percentage of variance explained by principle components.

Clustering is subsequently carried out on the projected space, i.e., the space spanned by the orthonormal PC1-PC3. Our goal is to partition the given data set into groups (clusters) such that the data points in a cluster are more similar to each other than points in different clusters. Thus, the main concern is to reveal the organization of patterns into sensible groups, allowing us to discover similarities and differences, derive useful conclusions about them, and discover distribution of patterns and interesting correlations in the dataset, thus making modelling easier. We employ agglomerative hierarchical clustering. Hierarchical clustering is an agglomerative schema allowing us to investigate groupings in the data, simultaneously over a variety of different scales, by creating a cluster tree. The cluster tree, which is called a dendrogram, is not a single group of clusters, but rather a multilevel hierarchy, where clusters at one level are joined into a single cluster at a higher level of the hierarchy. The most straightforward way to express alikeness between different buildings is to calculate their distance in the projected space, consisting of the first three principle components. The distance metric used is the standardized Euclidean distance and the unweighted pair-group method using

arithmetic averages (UPGMA) respectively, which proved to be adequate choices for our data. Dissimilarity between buildings and clusters are measured by the standardized Euclidean distance. They are both popular choices for clustering tasks [45].

Moreover, a distance metric has to be introduced in order to measure the degree of association between a data point and a cluster or between two clusters. This type of distance measure is called linkage and the most prominent selection for our case is average linkage. The size of the link expresses the degree of similarity between two connected components. The goal of assigning buildings to specific groups is equivalent to partitioning the cluster tree to disjoint segments, each one corresponding to a different cluster. The optimal level for this cut can be inferred by comparing the length of each link between cluster pairs in the cluster tree with those links at the same level of hierarchy. If they are similar, then the data clustered at this level show great alikeness, and therefore a high level of consistency, otherwise the links appear to be inconsistent. The existence of this kind of inconsistency, denoting a possible cut on the dendogram, is expressed by the inconcistency coefficient. The higher the value of this coefficient, the less associated the buildings connected by the link are. Leaf nodes have an inconsistency coefficient equal to zero. The depth used for the coefficient's calculation, denoting the number of levels of the cluster tree included in the calculation, is in this case two. In order to determine the cut-off threshold for the inconsistency coefficient one constraint has to be accepted, namely that the set of resulting clusters is the minimal one.

The results are shown in Figure 3.3.3 where the projected scores are visualized in a 3-dimensional space consisting of the first three principle components. An additional processing step we take after clustering is to exclude from subsequent analysis buildings with few session arrivals. Those buildings actually group together in the same cluster. To assist visualization of results and since the major challenges to system engineering come from the heavy-load buildings, we use heuristics to filter them out. After trial and error, the rule we set for filtering is to exclude those clusters, in which all buildings have mean session arrival less than 50, and the average of their maximum hourly session arrival rate does not

surpass a threshold equal to 5. The remaining 74 buildings are shown in Figure 3.3.3.

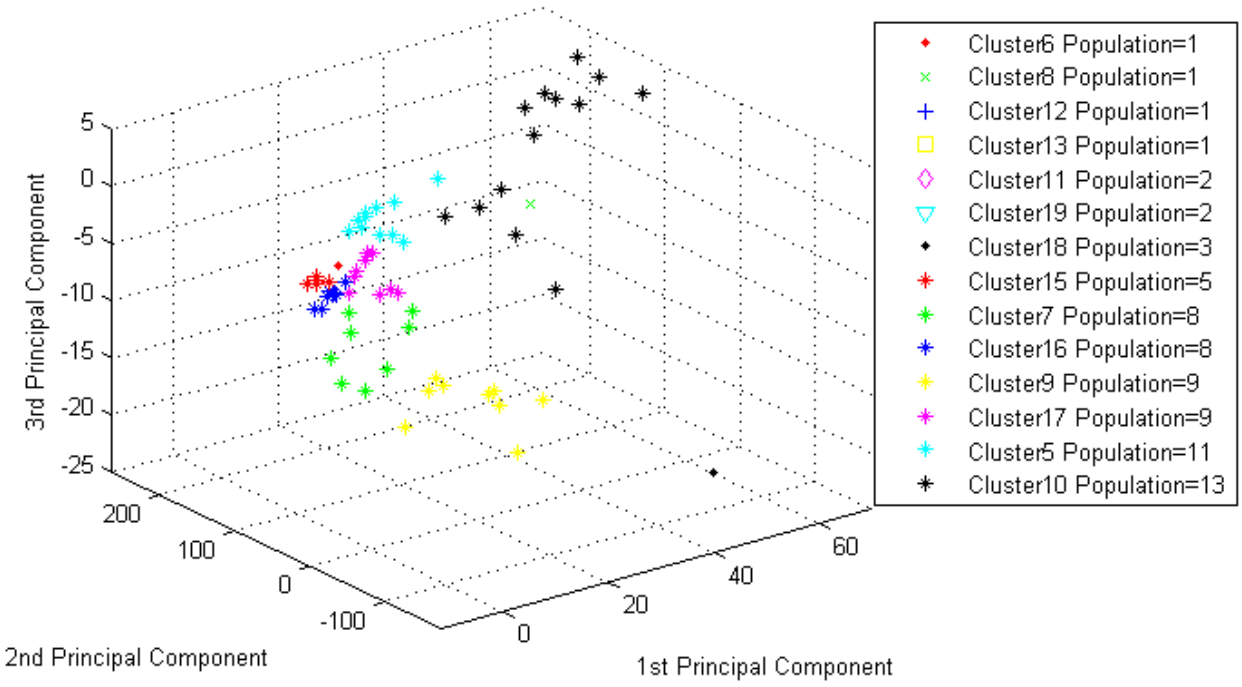

Figure 3.3.3. 3D plot of PC1-PC2-PC3.

It is evident from the plot that only a segment of the buildings' population is taken into account. Several buildings exhibited only a negligible amount of session arrivals during the tracing period analyzed. These buildings were discarded after the clustering procedure was complete. It is important to note that the filtering was not performed a priori, but rather all buildings participated in the analysis and then we could identify which clusters consisted of low mean session arrival buildings and eliminate them. Figure 3.3.3 plots the PC1-PC3, using different colors and symbols to present the various clusters obtained by the hierarchical clustering. To characterize each cluster, we calculate the cluster centroid in the projected space. To capture the main behavioral and statistical characteristics of each cluster, we then back transform its centroid, getting the "signature" session arrival rate series that corresponds to the specific centroid. This signature then works as a

profile for that particular cluster. Figure 3.3.4 plots the signature profiles for the obtained clusters. For example, clusters 5 and 10, which are mostly populated by residential buildings, exhibit a high arrival rate during afternoons and a similar pattern during weekends; on the other hand, clusters 9 and 16 consist of academic buildings, which have a strong session arrival rate peak during the mornings on weekdays and a distinct drop during weekends.

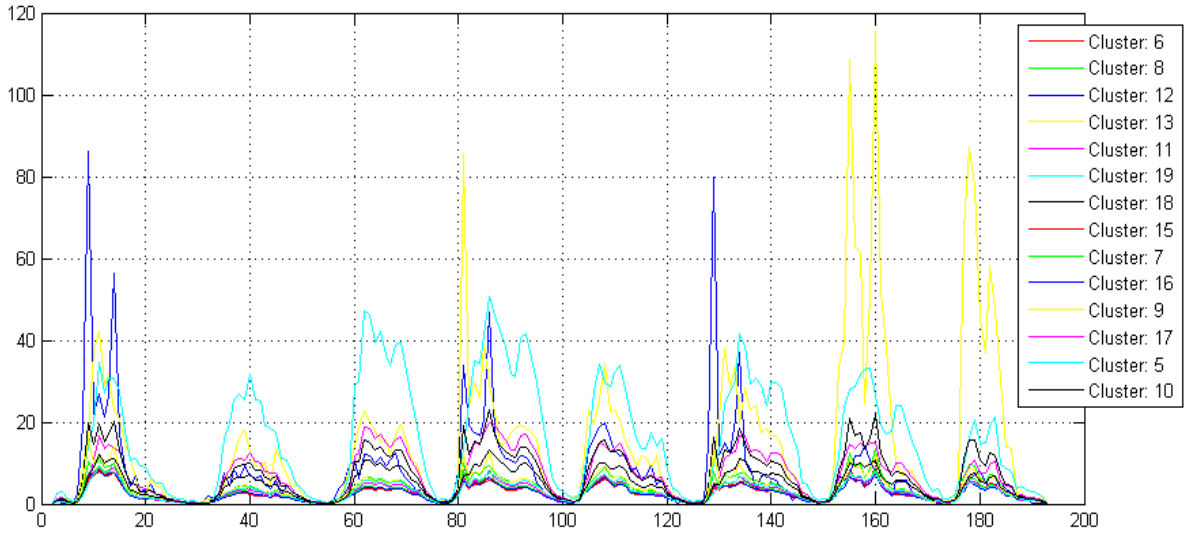

Figure 3.3.4. Cluster profiles.

C) Cluster validation

To validate the clustering result, we can use some internal criteria to measure how well a clustering fits the geometric structure of the data with no reference to information known a priori, such as the silhouette index.

The silhouette index is a confidence indicator on the membership of the i^{th} building in cluster X_j. A silhouette index value close to unity implies that the i^{th} building has been assigned to an appropriate cluster. An index value close to zero suggests that the sample could also be assigned to the nearest neighbouring cluster, i.e. such a sample lies equally far away from both clusters. A silhouette value close to –1, one may argue that such a sample has been misclassified. Thus, for a given cluster, X_j, it is possible to

calculate a cluster silhouette S_j, which characterises the heterogeneity and isolation properties of such a cluster. It is calculated as the sum of all samples' silhouette widths in X_j. Moreover, for any partition, a global silhouette value or silhouette index, GS_u, can be used as an effective validity index for a partition U. Based on this methodology we can not only examine the current clustering results but it is also possible to validate the 192 feature selection. One would argue that the usage of a more compact dataset, such as a 24hour feature set capturing the diurnal effects and preserving lower memory and computational demands is much more appropriate.

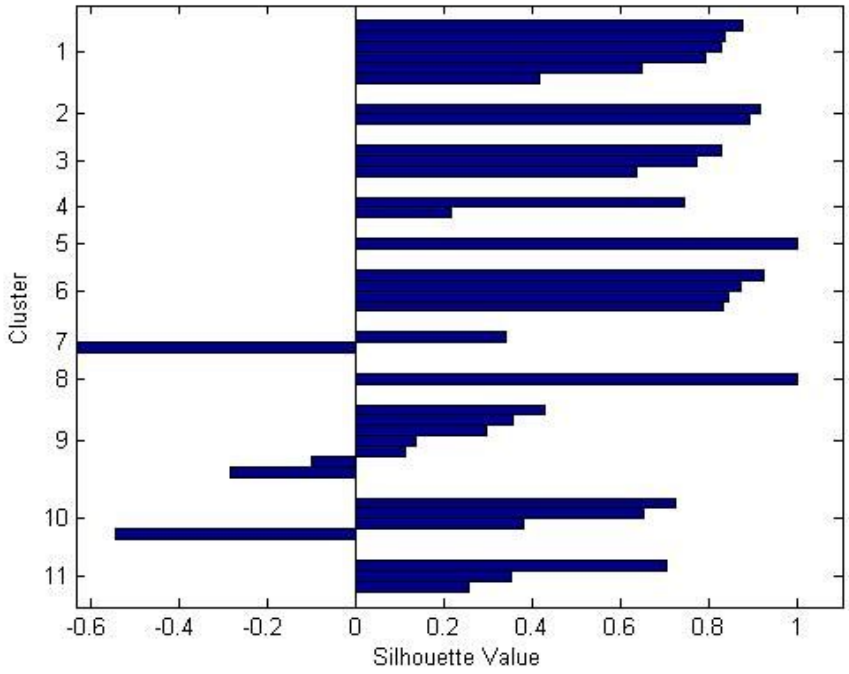

Figure 3.3.5. Silhouette cluster validation index (192-elements feature set).

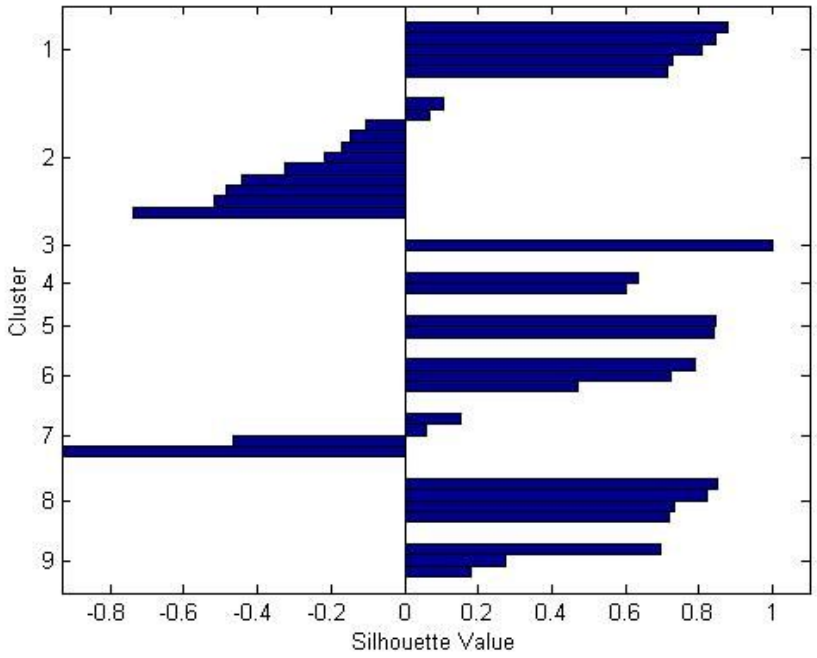

Figure 3.3.6. Silhouette cluster validation index (24h feature set).

Therefore, we repeated clustering with a reduced 24-element feature set. Each one of the 24 features corresponded to the mean hourly session arrival rate estimated over all days of the tracing period. This feature set is more compact, while it can still capture the diurnal effect. However, it results in a significant reduction of the GS_u value, 0.21 compared to 0.39 for the 192-element feature set. It comes out that the 192-element feature set reflects better the temporal variation across week days. Averaging daily session arrival rates, the 24-element feature set results in loss of detail in our data with negative impact on clustering.

With the first clustering alternative, combining SVD with the scale factor vector, we obtain 9 clusters. The largest cluster contains 216 buildings, all of which satisfy the filtering criteria for low-utilized buildings. The remaining 8 clusters all appear reasonable; the GS_u index is 0.76 when all clusters are considered. The second SVD-based clustering alternative produces 26 clusters; eighteen are filtered out when applying our filtering rules corresponding to 181 buildings. The estimated GS_u index is 0.40. Out of the three approaches, the SVD-scale factor approach results in the best

separation between high/low traffic buildings, whereas the one huge cluster produced consists of buildings with low session arrival rate.

3.4 Clustering with respect to session-level flow-related variables

After clustering the buildings according to session arrival rate, we need to model the session-level flow-related variables (see Table 3). There are different alternatives for this task in light of the clustering work for session arrivals. One option would be to perform a separate clustering of buildings for each flow-related attribute. However, this approach would give rise to a large total number of clusters and would complicate the modeling effort. A variant of this would be to carry out the additional clustering within each cluster obtained from the session arrival rate clustering. Another alternative is to consider modeling flow-related variables using the same building groups that come out of the clustering on session arrival rates. In our validation analysis, we will compare this last approach against more heuristic groupings (i.e., based on the building-type).

3.5 Conclusions

Traffic demand modeling has resulted in a hierarchical modeling framework evolving around wireless sessions and network flows. We have proposed parametric distributions for the four traffic variables we model: time varying Poisson process for hourly session arrivals, BiPareto distributions for in-session flow numbers and flow sizes, and Lognormal for in-session flow interarrivals. These distributions capture the modeled variables across different spatial scales, ranging from individual buildings to the whole network, and over two different monitoring periods spaced one year apart. Application of clustering techniques at building level is a promising way to compromise accuracy with scalability in large-scale wireless network modeling. We got good results combining clustering algorithms with PCA and SVD, but heuristical ways to group buildings also perform well with flow-related variables.

4. Validation

4.1 Introduction

In this section we evaluate the different modeling alternatives described in previous sections. We work with individual buildings and compare how the models' capability to capture the traffic demand at the building level changes as we zoom in/out of the trace data and consider different levels of detail in our modeling. We consider two alternatives for parameterizing the time-varying Poisson process for the session arrivals: the hourly session arrivals of the specific building we study and the signature of the cluster this building was assigned to. For the flow-related variables, the levels we consider in increasing order of spatial aggregation are the building, cluster, building-type and, for comparison reference purposes, the network level. We combine these alternatives into six scenarios and use them alternately in our simulations to assist the illustration of our main findings and support our discussion. Table 6 summarizes these scenarios and their requirements in terms of sampling distributions when the whole wireless network is to be modeled. In each we model four variables: the session arrival process and the three flow-related variables.

Modeling scenario	Description	Sampling distributions
bldg-bldg	Both session arrivals and flow-related variables are modeled after bldg-specific data for the whole trace duration	4*N N : number of bldgs
bldg-bldg (day)	The same with bldg-bldg, only now different distributions are derived for each day of the monitoring period	4*N*D D : number of days
bldg-bldgtype	Session arrivals are modeled after bldg-specific data and flow-related variables over data aggregated at bldg type level	N + 3*M, M : num. of bldg types
cluster-bldg	The cluster signature is used for session arrival modeling, whereas the distributions for flow-related variables are drawn from building-specific data	C + 3*N, C : number of clusters
cluster-cluster	Both session arrivals and flow-related variables are modeled after clustering	4*C
bldg-network	Bldg-specific data for session arrivals, network-wide distributions for the flow-related variables	N+3

Table 6. Modeling Alternatives-Scenarios for Simulation Validation.

Given the heavy-tailed session durations, our simulation times are in the order of days rather than hours. We compare synthetic traffic against traces with respect to building-level traffic variables not explicitly addressed by our models. Such variables are the aggregate flow arrival count process and the aggregate flow interarrival time-series for the building under study. We examine first-order and second-order statistics of the flow interarrival process and hourly flow arrival counts. We present results for two buildings, one academic (McColl) and one residential (Hinton James). They are two of the busiest campus buildings and represent the two main building types. In the same time they exemplify the use of heuristical (McColl) and more formal (Hinton James) clustering for achieving a good trade-off between model precision and scalability.

4.2 McColl academic building

McColl is the busiest campus building in terms of session arrivals. Irrespective of the clustering approach followed in Section 3.3, the McColl building does not group with other buildings but rather forms a separate cluster on its own. Therefore, the cluster signature coincides with the building hourly session arrival rate and the modeling alternatives are only relevant to the set of distributions for the flow-related variables. A first view of the "noise" that averaging introduces is given in Figure 4.2.1. The plot compares the cumulative empirical distribution function of the in-session number of flows for the McColl building with those estimated for all academic buildings and network-wide. The deviation between the curves increases with the degree of spatial aggregation of data. The way these discrepancies affect the aggregate flow-related metrics we described in Section 3.2 is summarized in Figures 4.2.1-4.2.3.

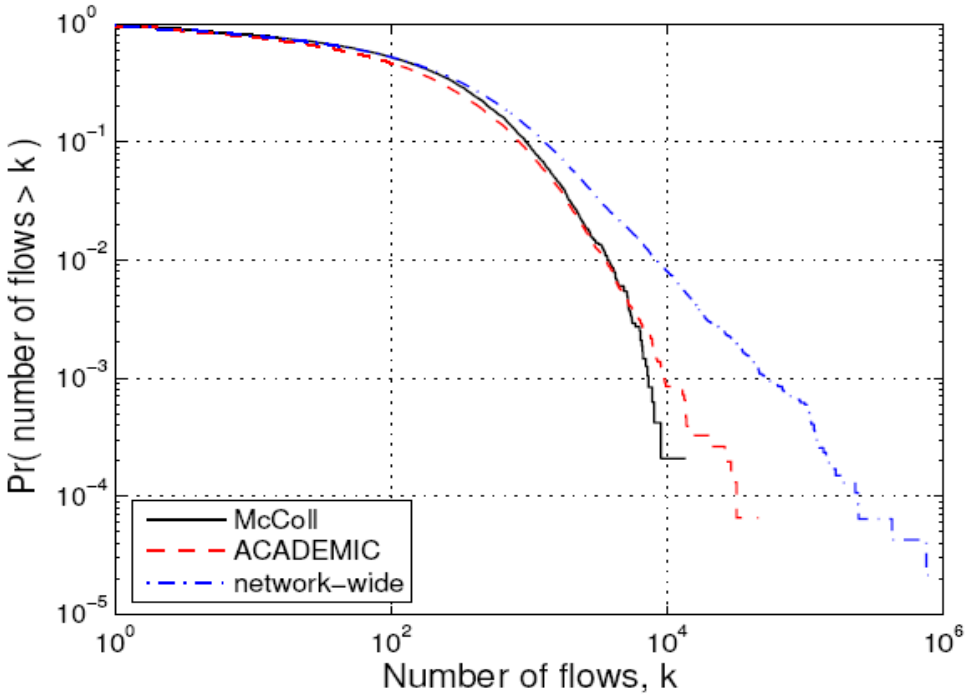

Figure 4.2.1. Number of flows per session: complementary cumulative distribution function under different building grouping alternatives.

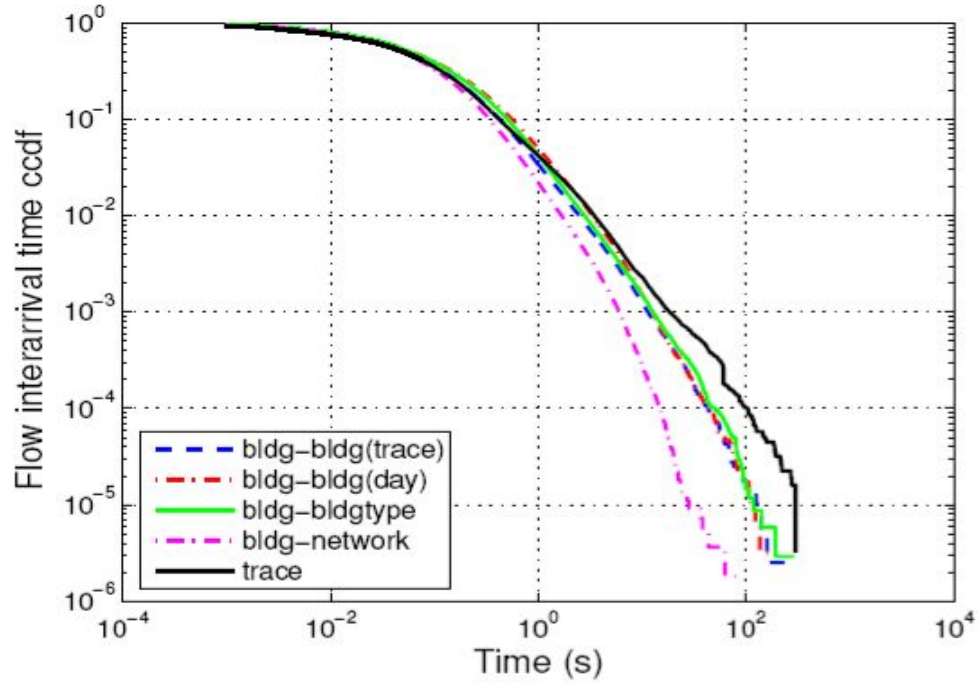

Figure 4.2.2. McColl building: cumulative empirical distribution function of aggregate flow interarrivals.

With respect to aggregate flow interarrivals, the synthetic traffic generator tracks most closely the trace when we model the in-session flow number and flow interarrivals separately for each one of the three days of simulation time. Considering a single set of distributions over the whole trace, does not give better results than when using the aggregate distributions for academic buildings. This implies that the averaging in the time-dimension may cancel out the benefit of getting higher spatial resolution out of the trace data. Moreover, staying at the building-type level gives us comparable precision with that obtained when zooming into the building-specific data. Figure 4.2.2 clearly suggests that reuse of the network-wide distributions for modeling traffic demand at finer spatial scales is not an attractive alternative. Looking at the autocorrelation process in Figure 4.2.3, one notes that the bldg-bldgtype curve is not much worse than the one corresponding to the bldg-bldg scenario. In fact, the former seems to underestimate less the short-term autocorrelation than the latter.

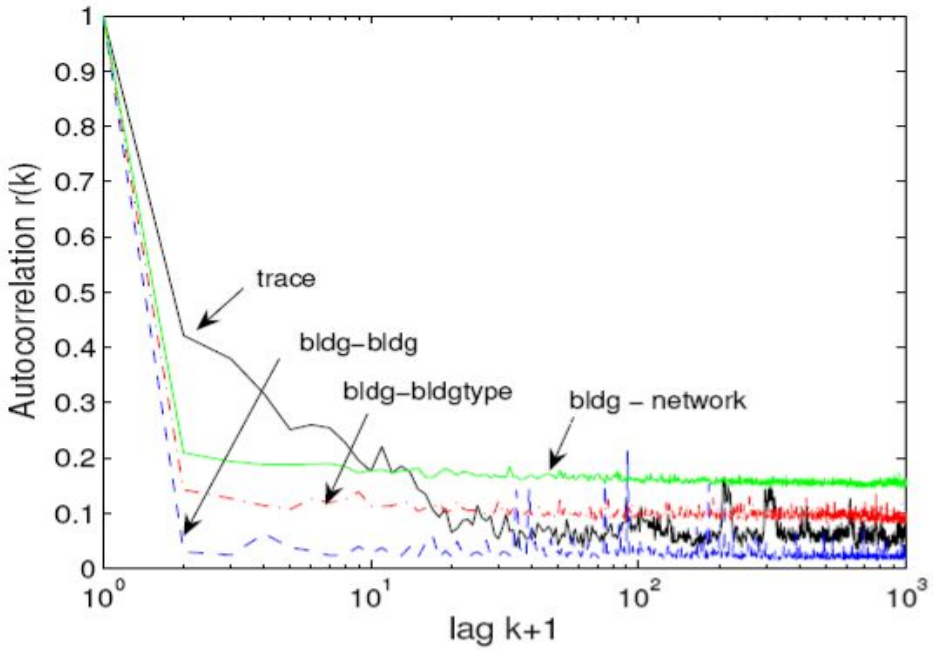

Figure 4.2.3. McColl building: autocorrelation of aggregate flow interarrivals.

Finally, inferior in absolute terms is the match for the hourly flow counts, which is the most demanding metric. Figure 4.2.4 plots the averages over 20 simulation runs along with their 95% confidence intervals. In this case, further improvement would be obtained by modeling the flow-related variables over shorter time periods than over the full monitoring period or a day. In fact, the standard practice is to focus the modeling attention on short time windows where the building activity experiences its peak (busy hour). In any case, the aggregation along the building type performs only marginally worse than the bldg-bldg scenario. The required number of sampling distributions for modeling the each campus building under the bldg-bldg scenario would be 4*N*D = 3000, for N = 250 and D = 3. When all buildings of the same type are modeled after a common set of distributions for flow-related variables, their number is reduced down to N + 3*M = 274, for M = 8.

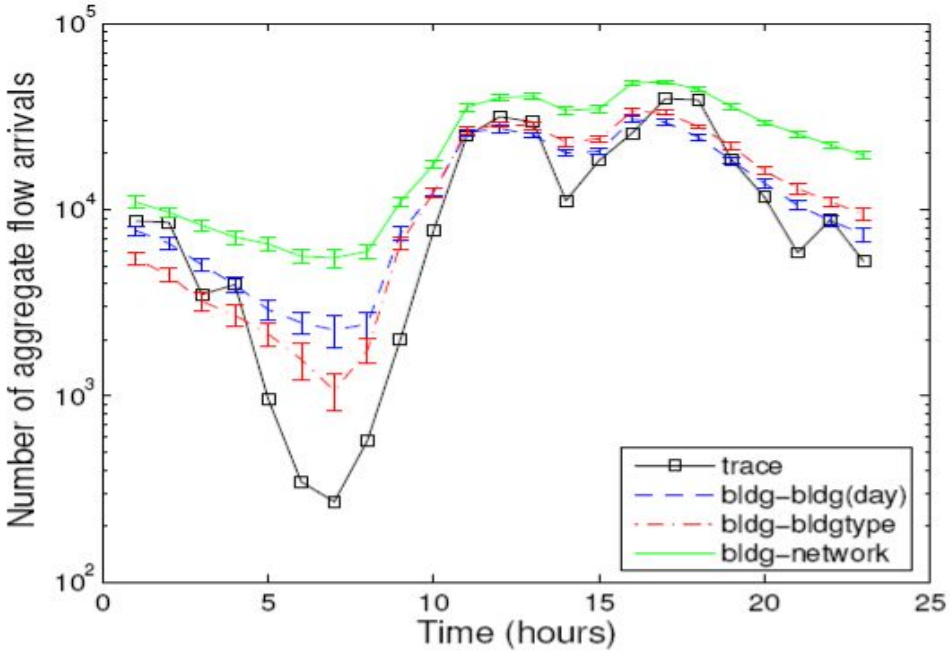

Figure 4.2.4. McColl building: aggregate hourly flow arrivals.

4.3 Hinton James residential building

Contrary to McColl, the Hinton James building was clustered together with other buildings under all clustering alternatives described in the clustering study. We consider for further analysis the cluster that came out of the

SVD-scale factor technique, which is the one that gave the highest global silhouette index amongst the three alternatives. This cluster comprises three other buildings, two of the social and one of the library type. Therefore, one additional modeling alternative now is to consider the cluster signature instead of its hourly session arrival rate for modeling the session arrivals in the building. We consider four alternative scenarios in our simulator. In two of them we model session arrivals after the building hourly session arrivals and in the other two we use the cluster signature. For the flow-related variables, we have three alternatives for getting the respective sampling distributions: consider only the building-specific data, aggregate over data from all four buildings in the cluster and, aggregate over data from all residential buildings in campus. The precision compared to aggregation level trade-off is even more clear than with the Mc-Coll building. Interestingly, the cluster-bldg combination presents a good compromise with matching score too close to the one obtained when we consider the specific building session arrival rates.

Figures 4.3.1-4.3.5 summarize the relative performance of the four alternatives. The bldg-bldg curve corresponds to modeling both the hourly session arrivals and the three flow-related variables after the building data, whereas for the cluster-bldg curve the hourly session arrival rate time series is assumed to be common for all buildings in the same cluster and coincides with the cluster signature. Likewise, the bldg-bldgtype arises when the distributions for the flow-related variables are derived by looking at data from all campus academic buildings. Under this scenario, the distributions for the three flow-related variables are the same for all academic buildings. With the bldg-network approach, averaging takes places over data from the whole network with the same three distributions being used in all buildings.

The "noise" that averaging introduces becomes clear in Figure 4.3.1, where the hourly sessions arrival rate in Hinton James is plotted against the cluster signature. How this deviation affects the modeling accuracy is summarized in Figures 4.3.2-4.3.4

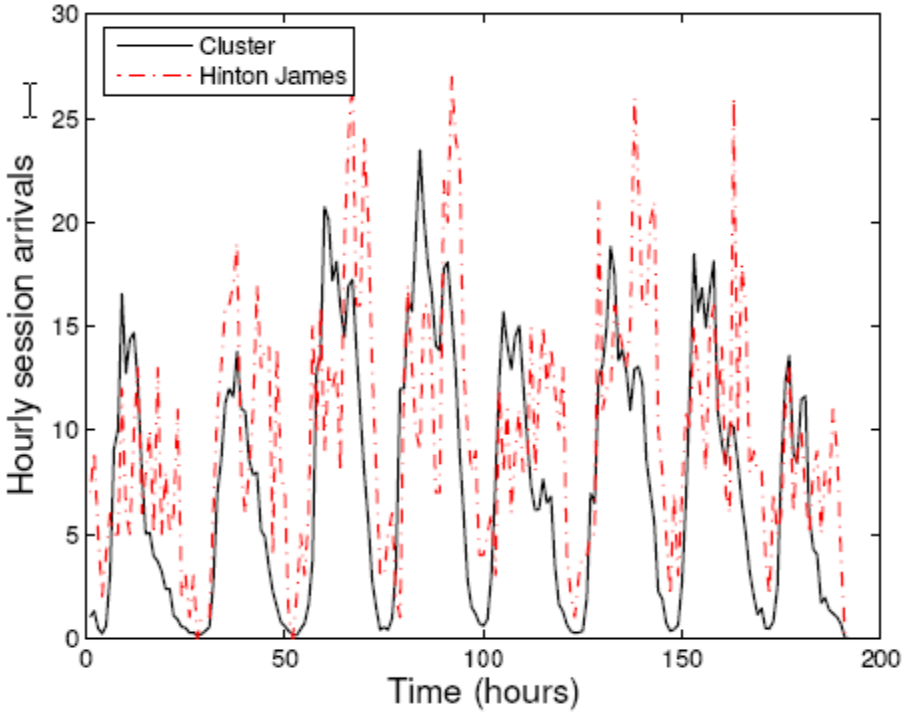

Figure 4.3.1. Hourly session arrivals: Hinton James compared to the cluster signature.

The bldg-bldgtype alternative is inferior to the cluster-bldg one, implying that modeling in-session flow number and flow interarrivals by simply taking into account data from all residential buildings has some cost. However, it is still better than applying the clustering results obtained for the hourly session arrival rates to the modeling of the flow-related variables as well. This becomes clear in Figures 4.3.2 and 4.3.3, which show that when this happens the mismatch is the worst of all scenarios. It comes out from these plots that the building type can be most useful in grouping buildings with respect to the session-level flow-related variables we model.

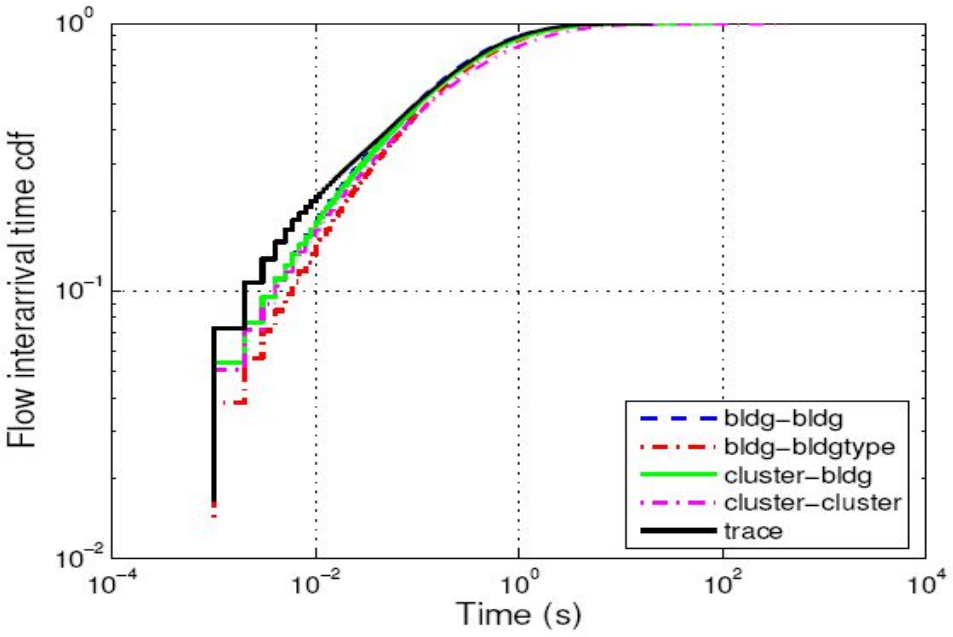

Figure 4.3.2. Hinton James building: empirical distribution function of aggregate flow interarrivals.

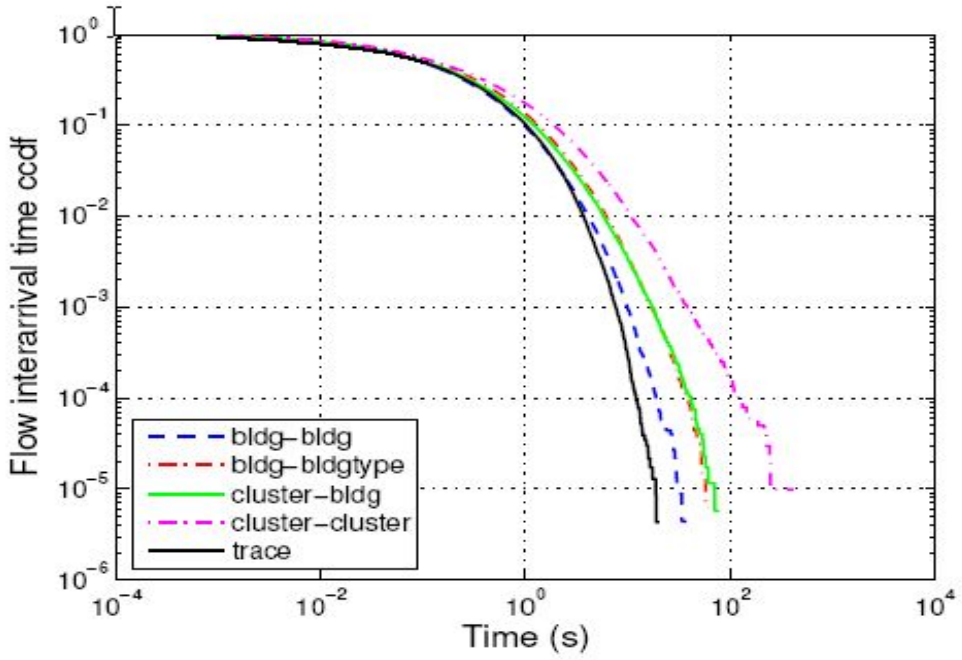

Figure 4.3.3. Complementary empirical distribution function of aggregate flow interarrivals.

The relative performance of the four scenarios is preserved with respect to the second-order statistics (Figure 4.3.5) and the flow-counts (Figure 4.3.4). In the second case, the match for the hourly flow counts, which is the most demanding metric, is inferior. In this case, further improvement would be obtained by modeling the flow-related variables over shorter time periods than over the full monitoring period or a day. In fact, the standard practice is to focus the modeling attention on short time windows where the building activity experiences its peak (busy hour). As with the McColl building, the implication is that additional detail in the time domain may be necessary to get higher precision. Nevertheless, considering the cluster signature instead of the per-building session arrival rates gives almost identical performance. Again, this happens at the benefit of scalability, since the required number of sampling distributions (parameter sets) to be entered in the simulator is C+3*N = 770, for C = 20 versus 4*N = 1000, respectively. Even better scalability in this case would be achieved under a combination cluster-bldgtype, i.e., if we used the clustering results to model the session arrival process and aggregate data at the building-type level for the flow-related variables. The required number of sampling distributions would be C+3*M = 44, resulting in an impressive reduction of complexity in the simulator.

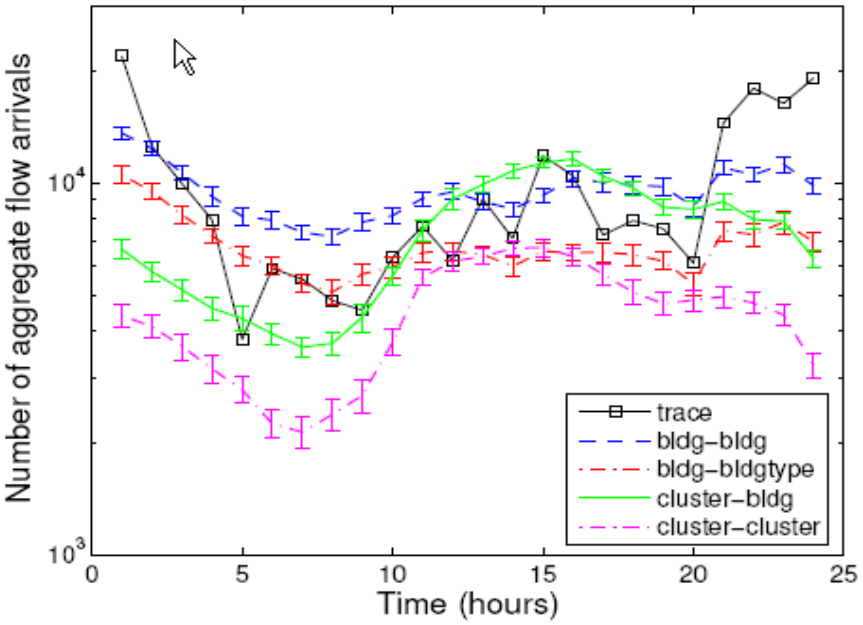

Figure 4.3.4. Aggregate hourly flow arrivals.

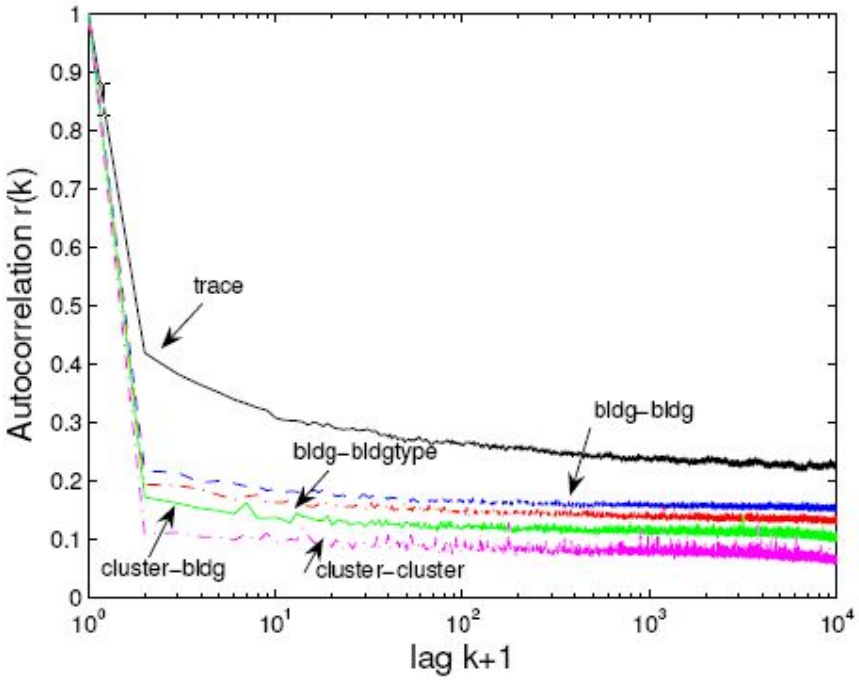

Figure 4.3.5. Autocorrelation of aggregate flow interarrivals.

With respect to aggregate flow interarrivals, as expected, the synthetic traffic generator tracks most closely the trace when we model the in-session flow number and flow interarrivals after the building data. It is more notable though that the bldg-bldgtype alternative is inferior to the cluster-bldg one, implying that modeling in-session flow number and flow interarrivals by simply aggregating data from all residential buildings has some cost. However, it is still better than reusing the clustering results obtained for the hourly session arrival rates to the modeling of the flow-related variables. This becomes clear in Figure 4.3.3, which shows that when this happens the mismatch is the worst of all scenarios. It comes out from these plots that the building type can be most useful in grouping buildings with respect to the flow-related variables in our model.

5. Short-term Traffic Forecasting

5.1 Introduction

One of the main goals of this study is to enable APs to perform short-term forecasting in order to provide better load balancing, admission control, and quality of service provisioning. Specifically, they can use the expected traffic estimations to decide whether or not to accept a new association request or advise a client to associate with a neighboring AP. In addition, the traffic models can assist in detecting abnormal traffic patterns (e.g., due to malicious attacks, AP or client misconfigurations and failures).

The main contributions of this section are the following: we discovered that both the total traffic load and number of associations at each AP are lognormal distributions. Our results also suggest that the average total traffic from an association in more "popular" APs (i.e., those with more associations) is larger than the average total traffic in less popular APs. As the popularity of an AP grows, the average traffic per association increases. More specifically, the logarithms of the total number associations and traffic load at each AP are strongly correlated. We also found a spatial locality in the most heavily utilized APs and observed diurnal periodicities at the total traffic load of the wireless infrastructure and also at several APs. Based on its periodicity and recent traffic history, we propose several models for the traffic load at an AP. Finally, we build some simple traffic forecasting algorithms that employ these models and evaluate their performance. To the best of our knowledge, this is the first study on traffic forecasting using actual traces from a 802.11 infrastructure.

Two types of data are used, namely, data collected using SNMP, and packet header collected from the link between the university and the rest of the Internet.

5.2 Traffic load notation

Based on the SNMP trace for each AP, we produce a time series of its traffic load at hourly intervals. This traffic is the total amount of bytes received and sent from all clients that were associated with the AP at that time interval. In the rest of this section, depending on the mathematical

expression, we will use two notations for these time series. Specifically, the traffic of the AP *i* during the i^{th} hour of day *d,* that corresponds to time *t,* is *Ti(h,d) = Xi(t).*

5.3 Basic statistics on traffic load

Figure 5.3.1 shows the total traffic (in GB) that each AP sent and received during the monitoring period. Total traffic was highly variable (notice the use of a logarithmic *y-axis*), so typical summary statistics, such as the mean, are of little use. The most heavily utilized AP handled 268 GB, while the most lightly utilized one handled only 9,621 bytes. In addition, seven APs did not handle any wireless traffic. The plot reveals a very consistent linear trend in the total traffic for almost 300 APs. It is therefore possible to model these data using an exponential model of the form $y = 10^{ax+b}$. The plot shows the result of fitting this model using linear regression (with a coefficient of determination R^2 of 0.9974 between ranks 75 and 300). The top 50 APs in terms of total traffic had significantly higher utilizations than this model predicts, while the bottom 125 had significantly lower utilizations. It is still striking to find such clear log-linear structure in the majority of data.

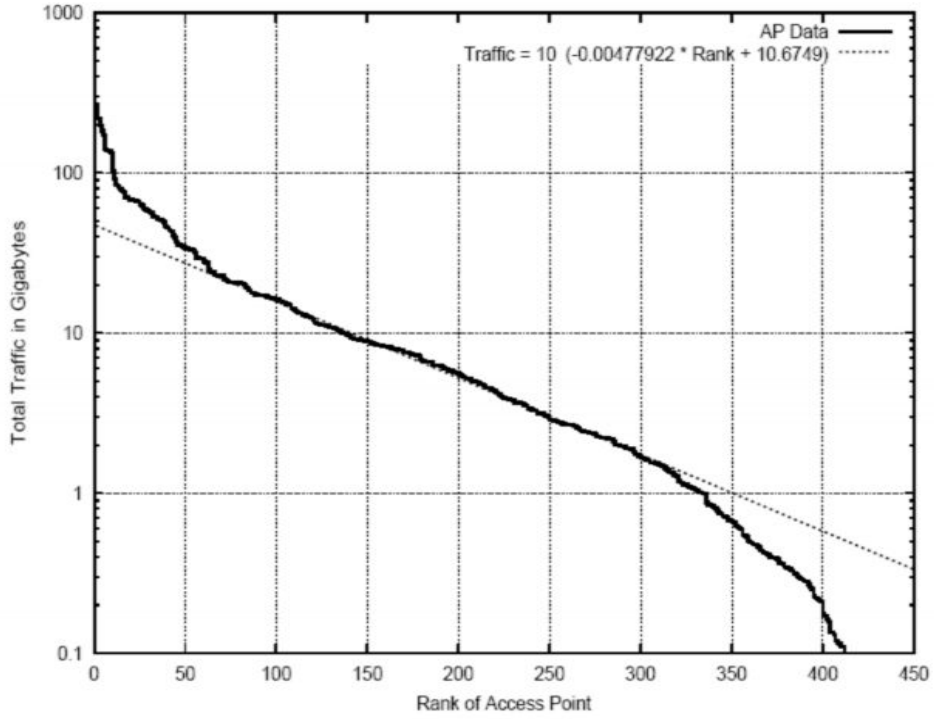

Figure 5.3.1. Distribution of total traffic load (GB) at each AP.

We further studied this data by plotting its cumulative distribution function, and this suggested a log-normal model, which turns out to work very well. Figure 5.3.2 shows the quantile-quantile plot of the total traffic data against the theoretical quantiles of a log-normal distribution with parameters $\mu = 22.35$ and $\sigma = 1.59$. The quantiles of the data are well within confidence intervals for all but the smallest values (lightest utilizations). Therefore, total traffic per AP appears to follow a log-normal distribution in our campus. Confirming that total traffic is log-normally distributed in other campus wireless networks would be a useful result applicable to the design of wireless testbeds and realistic experiments.

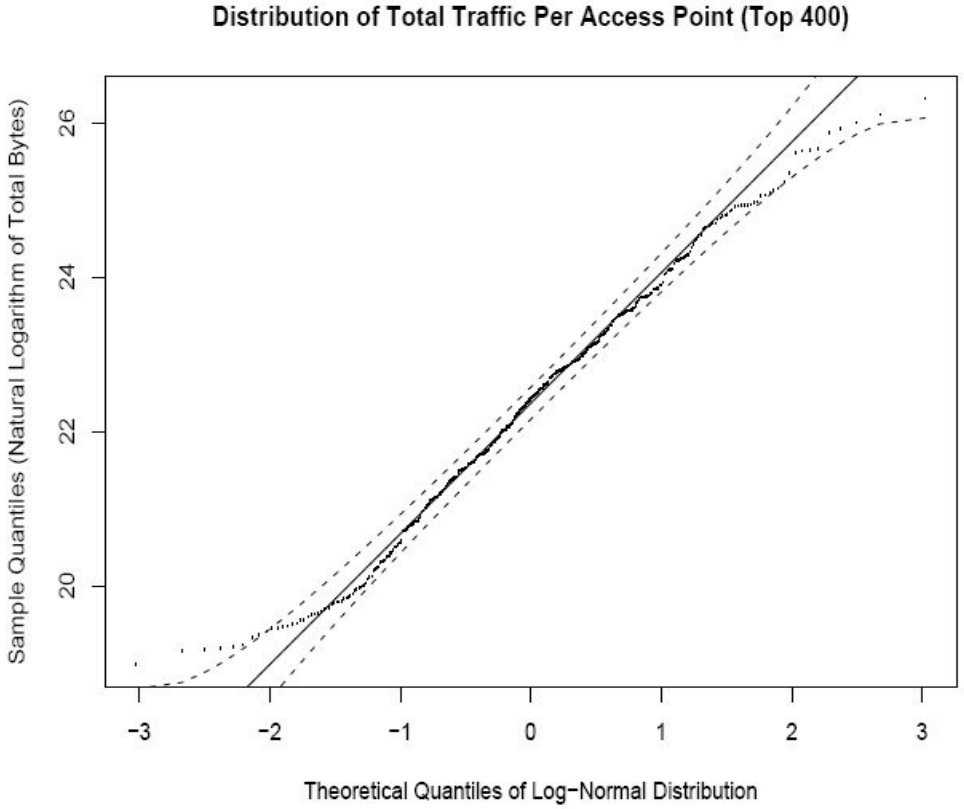

Figure 5.3.2. Distribution of total traffic load (GB) at APs.

Figure 5.3.3 shows the total number of associations per AP. As in the case of total traffic, there is a wide variability in the number of associations per AP. The most popular APs saw 33,581 associations, which constitute an average of 600 associations per day. Interestingly, we also found a

consistent linear region in this data (notice again the logarithmic y-axis). APs with the largest number of associations lie above the log-linear model, while those with the smallest number lie below it. We have also considered whether this data follows a log-normal distribution.

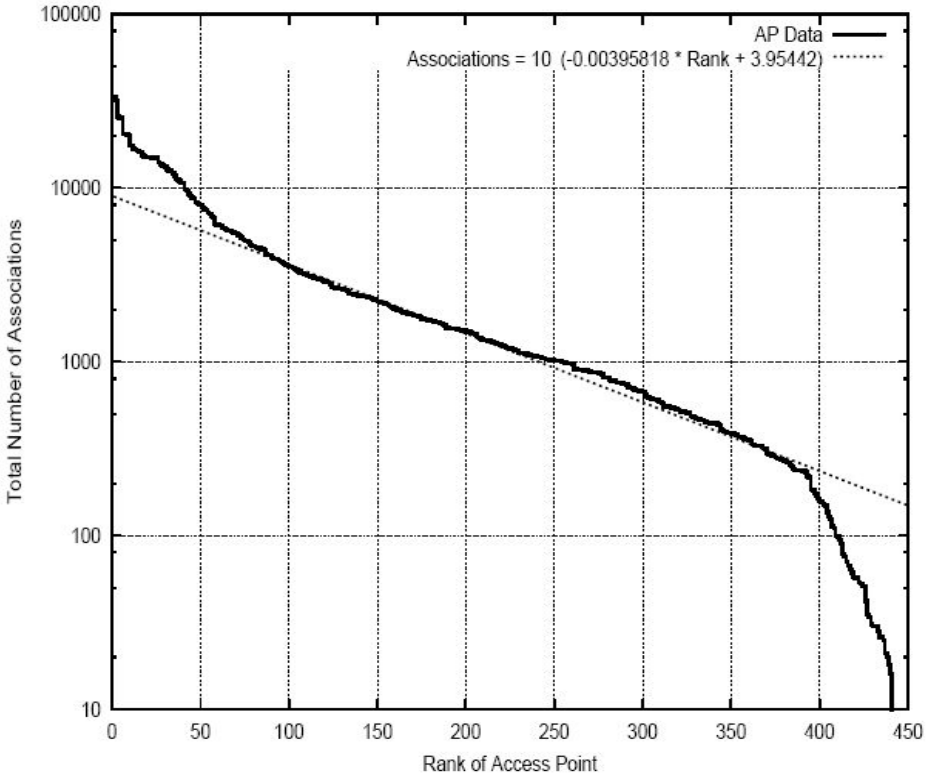

Figure 5.3.3. Distribution of associations at APs.

The quantile-quantile plot in Figure 5.3.4 shows a reasonable fit *(R^2* of 0.9919). However, the distribution of the data is heavier for the smallest values, and we found a significant deviation (ten quantiles outside confidence intervals) in the tail of the distribution. This finding implies that the average total traffic from an association in a more "popular" APs, those with more associations, is larger than the average in less popular APs. Associations in popular APs tend to send and receive more data, and as the popularity of an AP increases, the average traffic per association grows.

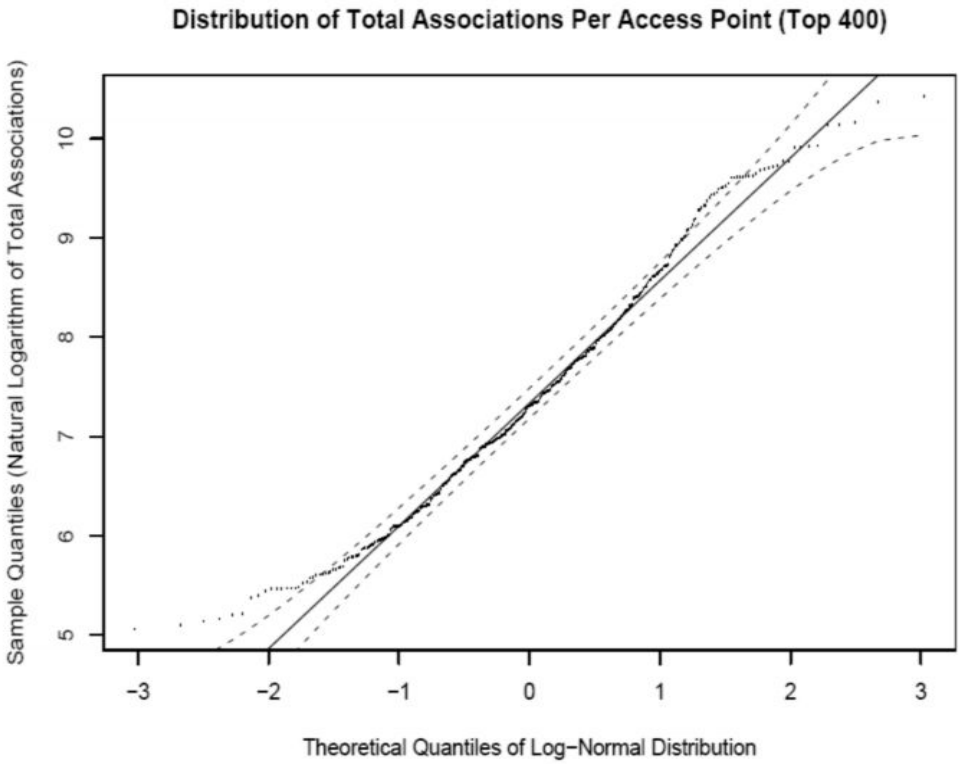

Figure 5.3.4. Distribution of total number of associations for each AP.

A. Correlation of the number of associations and traffic load

Previous studies [8] observed no correlation between the number of associations and the total traffic per AP. We have also studied this question and found only a weak correlation in our data (a Pearson's correlation coefficient of 0.46), which is consistent with previous work. However, we made an interesting discovery. Figure 5.3.5 shows a scatterplot of the logarithm of the total number of associations against the logarithm of the total traffic for each AP. There is a clear upward trend, with a Pearson's correlation coefficient of 0.8. This means that $log_{10}t \approx a$ x $log_{10}n + b$ where t is the total traffic and n is the number of associations. Therefore $t \approx 10^b n^a$, and the total amount of traffic grows very quickly with the total number of associations. Using linear regression, we found a = 0.619 and b = 8.096 (R^2 of 0.64). The interpretation of this result is that the number of associations has a multiplicative effect on the total traffic of an AP, rather than the more intuitive additive effect one would come to expect. We do not have a precise explanation for this phenomenon yet. We hypothesize that client reassociation dynamics could be a cause of it.

An IEEE802.11 client gets disconnected when its signal strength drops below a threshold. Frequently, the client reassociates immediately with the same AP. An increase in the number of associations at an AP can be due to reassociations of the same client or the arrival of new clients. Higher reassociation rates could be related to higher AP traffic loads.

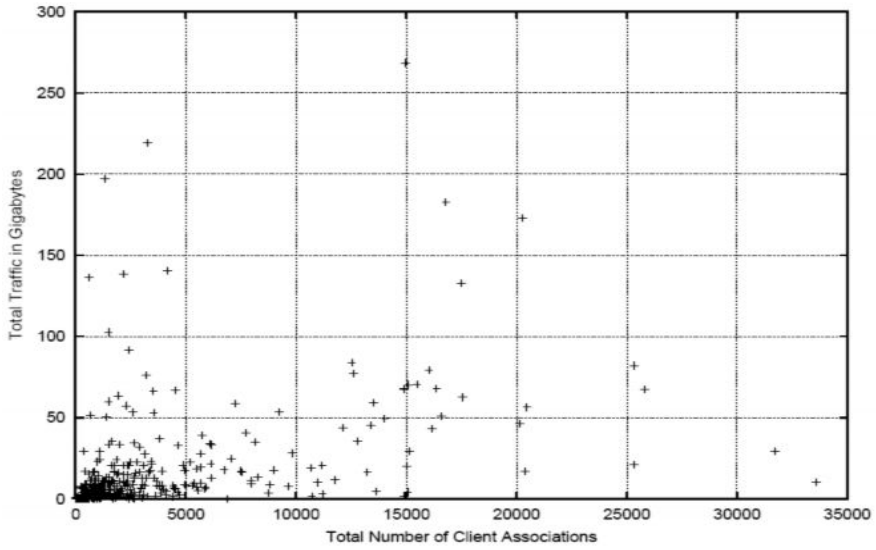

Figure 5.3.5. Total traffic load compared to the number of associations at each AP.

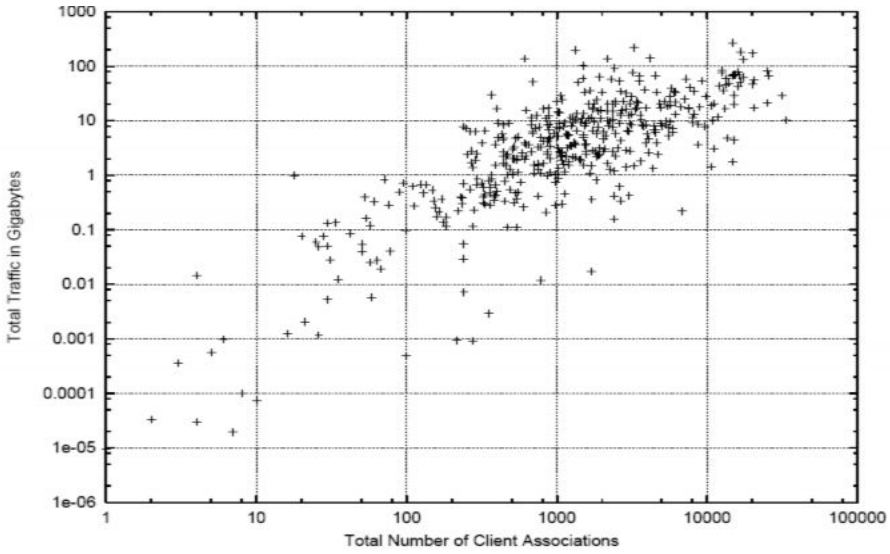

Figure 5.3.6. Total traffic load compared to the number of associations at each AP (log scale).

This finding implies that the average total traffic from an association in a more "popular" APs, those with more associations, is larger than the average in less popular APs. Associations in popular APs tend to send and receive more data, and as the popularity of an AP increases, the average traffic per association grows.

B. Hotspots APs and their spatial locality

We would like to distinguish the most heavily utilized APs. For that, we define the *hotspots* of the wireless infrastructure based on three metrics, namely, the maximum hourly traffic,the total traffic and the maximum daily traffic.

Hotspots based on maximum hourly traffic (set 1) These are the top *a%* APs ordered by their maximum traffic during an hour in the entire tracing period.

Hotspots based on total traffic (set 2) These are the top *a%* APs ordered by their total traffic during the tracing period.

Hotspots based on maximum daily traffic (set 3) These are the top *a%* APs ordered by their maximum traffic during a day in the entire tracing period.

Hotspot (main definition)

We define as a *hotspot* an AP that belongs in the top *a%* of APs with the highest maximum hourly traffic and in the top *a%* of APs with either the highest total traffic load or the highest maximum daily traffic load (i.e., the *set (set1 ∩ (set2 U set3)))*. We will use this definition in the following sections.

We first investigate the spatial locality of the hotspots and name two APs co-located, if they are placed in the same building. How likely is to find co-located hotspots in the campus? We found that for *a=20,* the percentage of co-located hotspots is above 76% and 79% for the hourly and total-traffic based definitions, respectively. 62% of the co-located APs belong in the *(set1 ∩ (set2* U *set3)))*. For $a = 10$, the corresponding percentages are about 11% smaller than their respective values for $a = 20$. Note that, if using the uniform distribution, we had randomly selected the same number of APs, the mean percentage of co-located APs in those selections is 48%. We are currently investigating other spatial locality properties of the hotspots (such as visit duration, applications, number of distinct clients, and usage

patterns) and plan to report these results in a followup study. For $a = 10$, there are 19 such APs.

5.4 Traffic forecasting methodology

We distinguish three main categories of forecasting algorithms based on their traffic models, namely, the *periodic-based*, the *AR-based*, and the *hybrid* algorithms. The first category exploits the periodicities (e.g., diurnal and weekly patterns) in the traffic by incorporating historical means of traffic. The second type considers only a *window of recent traffic history* (e.g., the traffic during the last three hours). The window size could be fixed throughout the forecasting process or dynamically adjusted. In the latter case, the algorithm monitors the traffic dynamically, detects prominent changes in the traffic, decides about the window size (amount of traffic history to be used) and applies the prediction algorithm using this window of traffic. Examples of such algorithms are the moving average-based algorithms described in the following paragraphs. When the traffic load exhibits both strong periodicities and temporal dependencies, hybrid models can be better choices. This classification can be extended depending on whether the algorithm uses the *predicted* or *actual* values of the recent history. More specifically, a forecasting algorithm is *one-step ahead,* when it uses the *actual* values of traffic for the traffic during the recent history window, and *multi-step ahead,* when it employs the predicted ones. To enhance the performance of the forecasting algorithm, additional information about the type of traffic, application, and client profile can be used. Finally, depending on whether one or several sources of data have been used, a forecasting algorithm is single-source or multi-source. For example, a forecasting algorithm using SNMP-based data and TCP-packet headers is a multi-source algorithm.

Our general methodology consists of the following steps: (A) Time-series extraction, data cleaning, and treatment of missing values; (B) Power spectrum and partial autocorrelation analysis; (C) Data normalization and traffic load modeling; and (D) Forecasting using the traffic load models.

A. Time-series extraction and treatment of missing values

While our monitoring system requested traffic load information from each AP precisely every five minutes, missing values are relatively frequent in our dataset. They are due to several reasons: (1) an AP may be down for

maintenance, or in the middle of an accidental reboot; (2) an AP may be too busy to reply to an SNMP query; (3) the network path between our monitor and the AP may be temporarily broken; and (4) query packets and response packets may be lost (they are transported using UDP). While these pathologies are expected to be infrequent, our dataset is large enough to contain numerous instances of each of them. Thanks to the cumulative nature of SNMP counters, we were able to reconstruct missing values quite accurately.

The basic technique for extracting an equally-spaced time-series $X = \{x_1, x_2, \ldots, x_n\}$ from SNMP data is to subtract the cumulative counters from two consecutive *polling* operations. In order to detect missing values and reboots, our polling samples include not only the cumulative counters but also the time of each polling operation, and the cumulative time that the AP has been running since the last reboot *(up time)*. This means that the *i-th* polling sample for an AP has the form (t_i, u_i, c_i), where t_i is time of the polling operation, u_i is the cumulative up time, and c_i is some cumulative counter *(i.e.,* total load in bytes).

Given two consecutive polling samples, the load x_i observed between t_{i-1} and t_i, generally equal to $c_i - c_{i-1}$. There are two exceptions. First, SNMP counters are represented using 32 bits, so counters often wrap-around. We consider that a counter has wrapped around whenever $c_i < 2^{30}$ and $c_{i-1} > 3*2^{30}$. In this case, x_i is equal to $c_i + (2^{32} - 1 - c_{i-1})$. Second, after a reboot, all the counters in an AP are reset. Therefore, if a reboot occurs at some point between t_{i-1} and t_i, x_i is equal to c_i and the value of c_{i-1} should not be subtracted from c_i. Reboots can be detected by checking the value u_i in each polling sample. If u_i is significantly less than $t_i - t_{i-1}$, the AP has been reset, and x_i is equal to c_i. Otherwise, x_i is equal to the subtraction of the two cumulative counters. Note that resets may create situations that look like a wrap-around, so the detection of the reboots should be performed before the detection of the wrap-arounds.

When all of the polling operations are successful, $t_i - t_{i-1}$ is equal to the polling interval *(i.e.,* 5 minutes). However, when a polling operation fails, $t_i - t_{i-1}$ is a multiple of the polling interval. If this is due to an AP reboot, the counter c_i only reports on the activity since the reboot operation.

Therefore, c_i becomes the last value of the time-series. The values between t_{i-1} and t_i, for which no polling samples were available, are set to zero (APs have no load while offline). If no reboot took place, the $c_i - c_{i-1}$ does not correspond to a single x_i but to the m values of the time-series between t_{i-1} and t_i. In this case, we perform linear interpolation and set each intermediate value of the time-series to $(c_i - c_{i-1})/m$. Finally, note that $t_i - t_{i-1}$ is not always exactly equal to the polling interval (or a multiple of it). The most significant cause is the retransmission mechanism in our SNMP monitor, which retransmits unanswered requests up to three times. Each new request is spaced by 5 seconds. Therefore, the maximum deviation of $t_i - t_{i-1}$ with respect to the polling interval is 20 seconds, and our time-series extraction program takes into account this deviation.

B. Spectrum analysis

We find that the aggregate hourly traffic for all APs in the infrastructure exhibits diurnal and weekly periodicities. Similar trends are observed in the hourly traffic for several APs by autocorrelation plot and spectrum analysis. 10 out of the 19 hotspots have a clear spike at 24 hours/cycle and do not have a high frequency variation. Also, some APs have weekly patterns at around 168 hours/cycle.

Figures 5.4.1(a) and (b) show the time series and spectrum plots of the hotspot AP 472. This AP exhibits strong diurnal periodicity. There are other APs with no clear periodic pattern, for which there is little prediction power among the historical data. Further smoothing does not appear to be helpful, at least with our current relatively short traces.

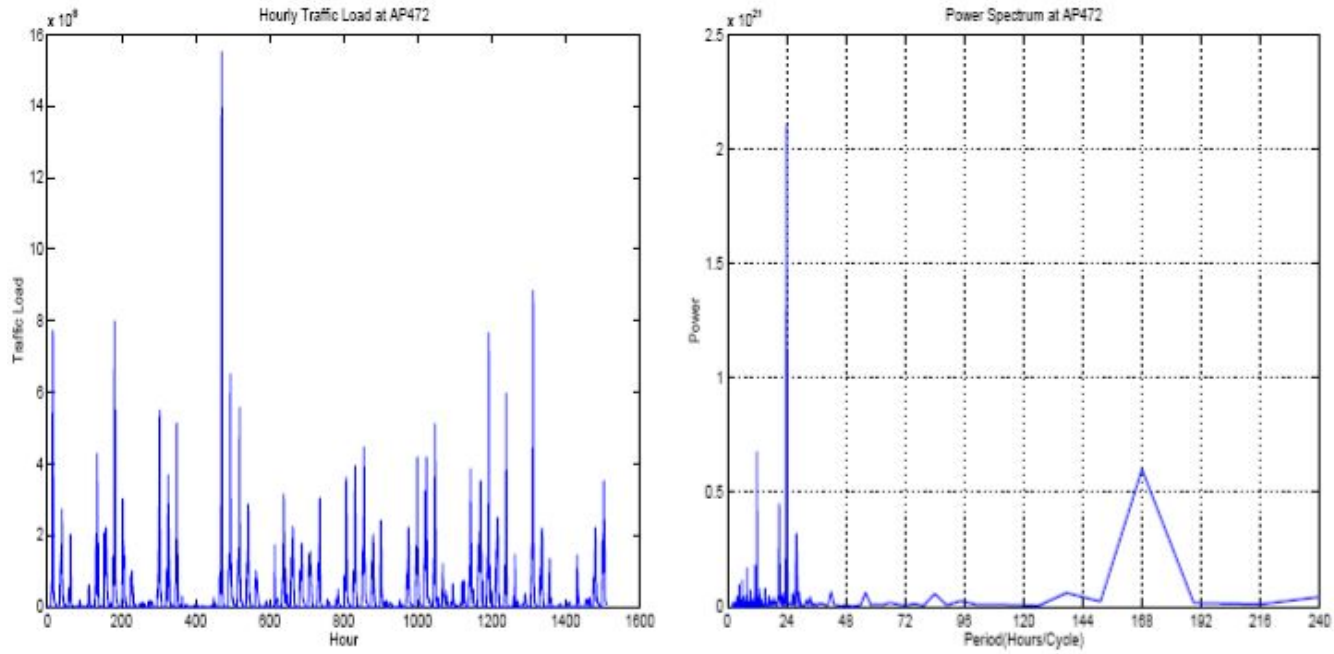

Figure 5.4.1. Traffic load at AP 472: (a) time series (b) power spectrum.

C. Modeling for Traffic Forecasting

1) Periodic-based forecasting: We propose several models that facilitate the diurnal and weekly periodicity of the traffic load. Let us first consider hourly traffic time series. We define the **historical mean hour** (P1) traffic of an AP as the mean of the traffic during that hour for each day in the history of that AP *(N_{days}* days). Similarly, the **historical mean hour-of-day** (P2) traffic is the mean of the traffic at such hour of day in the history of that AP. We tailor two simple models based on the historical mean hour and mean hour-of-day. Such periodic models can be extended for finer time scales (e.g., five-minute time series).

First, we model the traffic load at an AP during an hour. The model facilitates the diurnal and weekly periodicity of the traffic load. We define the *historical mean hour* traffic of an AP as the mean of the traffic during that hour for each day in the history of that AP *(N_{days}* days). We only consider weekdays.

For example, the historical mean-hour traffic for AP *i* is defined as

$$\mu_i(h) = (1/N_{weekdays}) \times \sum_{d=1}^{N_{days}} T_i(h,d) * IsAWeekday(d),$$

where $h = 1,..., 24$ and *IsAWeekday?(d)* is a binary indicator function that specifies whether or not the *d-th* day is a weekday, and

$$N_{weekdays} = \sum_{d=1}^{N_{days}} IsAWeekday?(d) \ .$$

Similarly, the *historical mean hour-of-day* traffic is the mean of the traffic at such hour of day in the history of that AP. For example, the mean hour-of-day for AP *i* is defined as

$$\mu_i(h,l) = (1/n(l)) \times \sum_{k=1}^{N_{days}} IsWeekday(k,l) \times T_i(h,k),$$

$$where\ h = 1,...,24,\ l\ "runs"\ from\ "Mon"\ through$$

$$"Sun",\ and\ n(l) = \sum_{k=1}^{N_{days}} IsWeekday(k,l)$$

The *IsW eekdayl (k,l)* is a binary indicator function that specifies whether or not the x is a weekday l. The *nw(l)* counts the total number of weekdays / (e.g., the number of Mondays). For example, for the $\mu_i(2)$ we take the historical mean of the traffic at AP *i* for all days in the history at 2am. Similarly, for the $\mu_i(2,$ *"Mon"),* we compute the mean of the traffic of all Mondays at 2am.

We tailor two simple models based on the historical mean hour and mean hour-of-day. Specifically, for each AP (e.g., AP *i),* we define the models Z_i^1 and Z_i^2, as follows:

$$(P1)\ Z_i^1(h,d) = \mu_i(h)$$

$$(P2)\ Z_i^2(h,d) = \sum_{i \in \{Mon,...,Sun\}} IsWeekday(h,d) \times \mu_i(h,l)$$

We propose two simple prediction algorithms based on the aforementioned models. P1 and P2 use the historical means to compute the $Z_i^k(h,d)$, $k = 1,2$ for P1 and P2, respectively, and predict the traffic load of AP *i* during the *t-th* time interval (that corresponds to the *h*-hour of day *d).*

2) Recent history based algorithms: We apply some simple linear predictors based on localized regression. They are less demanding than more complex linear predictors, such as ARIMA where selecting the order and coefficients requires a large amount of previous historical data. The linear models used are moving average and exponentially moving average.

A **simple moving average** (MA) is the unweighted mean of the previous *w* data points in the time series.

$$\hat{X}(t+1) = \frac{1}{w} \sum_{x=t-w+1}^{t} X(k)$$

A **weighted moving average** is a weighted mean of the previous *w* data points in the time series. A weighted moving average is more responsive to recent movements than a simple moving average.

An **exponentially weighted moving average** (EMA) is an exponentially weighted mean of previous data points,

$$\hat{X}(t+1) = aX(t) + (1-a)\hat{X}(t)$$

The parameter *a* of an EMA can be expressed as a proportional percentage. For example, in a 10% EMA, each time period is assigned a weight that is 90% of the weight assigned to the next (more recent) time period. A higher *a* cannot smooth out the measurement noise whereas a lower value is slow in adapting to changes in the time series. Note that for *a* equal to 1, we have a simple *AR(1)* model, very sensitive to the level changes in the traffic. For *a* equal to 0, the algorithm corresponds to a simple moving average less sensitive to the level changes. The prediction algorithm uses the aforementioned models to compute the predicted traffic for the next time interval. We will assume that these prediction algorithms consider a *fixed window* size for the recent traffic history.

3) Adaptive moving average based forecasting: Motivated by the need to better capture the burstiness of the traffic and adapt to its sudden changes during the forecasting process, we propose a novel forecasting approach. The **adaptive moving-average** algorithm dynamically detects the level shifts (i.e., prominent changes of traffic) in the traffic and establishes a new window size of the recent traffic. It then applies a moving-average based algorithm using this window of recent traffic. As the traffic is received and sent at an AP during each interval (e.g., *k-th* interval), the *X(k)* is being formed. The algorithm employs a sliding window that moves across the

$X(j)$, $j=k,k-1,\ldots$, and detects the level shifts. A recent history window has a minimum and maximum size (w_{min} and w_{max}, respectively), currently set at 3 and 12 time intervals.

Let us assume that the current window is L_m starting and ending at the beginning of the k_m-*th* and k_{m+1}-*th* interval, respectively ($w_{min} \leq k_{m+1}-k_m \leq w_{max}$). The traffic accessed during the window L_m is the set of all values in the $X(k_m),\ldots ,X(k_{m+1})$. Each traffic value $X(k_m)$ corresponds to the aggregate traffic accessed from the AP during the k_m-*th* time interval. Depending on the time-scale the k_m-*th* interval is one hour or 5-minute interval. The algorithm "scans backwards" the time series on the fly (while the AP operates), starting from the end of the current interval k_j ($k_j > k_{m+1}$) and detects level shifts in the traffic (accessed during those time intervals). A level shift flag is triggered when one of the following conditions becomes true.

$$(1) X(k_j) > X(k_l) \forall k_l \in L_m$$

$$(2) X(k_j) < X(k_l) \forall k_l \in L_m$$

If none of the above conditions are true, the current window L_m is expanded to include the current intervals up to k_j-*th* and the algorithm continues with the next interval k_{j+1}. When the current window size exceeds its maximum size, the algorithm keeps as its current window only the w_{max} most recent values. When one of the aforementioned conditions becomes true, a new interval Lj that corresponds to the subset $\{X(k_j),\ldots, X(k_{j+3})\}$ is formed. A level shift at k_j is *detected if* the confidence intervals of the traffic accessed during L_m and Lj are *non-overlapping (level shift criteria)*. In the case of a new level shift, the current window becomes the Lj and the algorithm continues to the next interval k_{j+1}.

4) Hybrid algorithms based on periodicities and recent history: To incorporate both the recent traffic and periodicities, we build a hybrid model that uses the moving average and historical means. Specifically, for each AP (e.g., AP *i),* we introduce the **hybrid model** (P3), a *weighted average* of the periodic-based models of the historical mean hour and hour-of-day and the simple moving average (MA) defined as

$$(P3)\ Z_i^3(h,d) = a \times (1/w) \sum_{k=t-w}^{t-1} X(k) + b \times \mu_i(h,d) + c \times \mu_i(h)$$

We experiment with different window sizes and weights to evaluate the impact of the recent history and periodicity on forecasting. Note that the P3 with weights (a,b,c) equal to (1,0,0) and history window w is a simple MA model of window w and has the form of an autoregressive process of order w, $AR(w)$. In that case, the prediction takes into account only the recent traffic history instead of the periodicity. The weights of the P3 can be established using multiple linear regression, revealing which of the predictors, namely, the historical mean hour, historical mean hour-of-day, recent-history have the greatest effect. The linear model takes the form $y = Xb + e$, where y is a vector of observations, X is a matrix of independent variables (regressors/predictors) and e is a vector of random disturbances. Multiple linear regression aims to obtain the best fitting curve by minimizing the least square errors. P3 with weights established via multiple linear regression is denoted as P-MA-RG. Note that P3 and P-MA-RG are *one-step ahead prediction algorithms,* since for recent traffic data points, they use the *actual* traffic values as opposed to the predicted ones (for the next-hour prediction).

5) Multi-sourced algorithms with flow-related information: The . Furthermore, we found a high correlation (above 90%) in the log-log scale between the traffic load and number of flows in both the hourly and five-minute time intervals. We designed two novel algorithms that use flow-based information to enhance the predictions. Unlike the previous algorithms that use only SNMP data, the new algorithms integrate SNMP and flow-based information. Specifically, we investigated the impact of number of flows and type of application on forecasting. For that, we correlated the SNMP-client based and TCP-packet headers information. Since the time granularity of the SNMP data is five minutes, we created five-minute time series of the number of active flows at each interval for each AP using the TCP-packet headers. We established the relation between number of active flows and traffic load per interval for each AP by applying linear regression on the two time-series generated using a *training data set.* Therefore, for each AP, the number of flow based model was a function of the form: $log(X(k)) = alog(N_{flows}(k - 1)) + b$, where $X(k)$ and $N_{flows}(k)$ are the total traffic and number of flows during the k^{th} interval of that AP, respectively. To forecast the traffic during the k^{th} interval the

number-of-flows-based prediction looks-up the number of flows at the previous interval and predicts as traffic demand the

$$\hat{X}(k) = e^{b}(N_{flows}(k-1))^{a} \qquad (1)$$

The **type-of-flow**-based forecasting algorithm predicts the traffic at the next time interval based on the estimation of the traffic "contribution" of its active flows at that interval. To compute the contribution of a flow, we use the median traffic size and duration of flows of the same port number (application type). There is a one-to-one mapping between the application type and port number of the destination of a TCP flow. For example, let us assume that $\bar{l}_p$ and $\bar{d}_p$ are the median traffic size and duration of flows of port number p and d the time-scale. To forecast the traffic load of an interval k, we find the set of active flows at the end of the $(k-1)^{th}$ interval. For each active flow f (of application type p) that started at time t_s we estimate it's expected traffic contribution at the k^{th} interval as the ratio of the median bandwidth requirement of this type of flows by the estimated duration of that flow,

$$\frac{\bar{l}_p}{\bar{d}_p}\min((t_s + \bar{d}_p - d*(k-1), d)$$

The type-of-flow-based forecasting algorithm will predict as the traffic load of the next time interval, the sum of the contributions of all active flows in that interval. Note that the type-of-flow-based algorithm does not consider the arrival of new flows at the next time interval.

6. Normalized ARIMA based time-series forecasting: There are hotspot APs whose traffic load shows strong diurnal periodicity. Figure 5.4.1(a) shows the time series plot of the hourly traffic load at AP 472, from which one can observe a clear diurnal pattern as well as a possible weekly pattern. To verify their existence, we plot the corresponding power spectrum in Figure 5.4.1(b). The plot indicates that the most dominate period is the 24-hour one, with smaller ones corresponding to 12 hours (day/night), and 168 hours (weekly period). Similar periodicities are observed in several other APs as well. The existence of such strong periodicities motivates us to consider forecasting traffic load using some time series models like ARIMA. Intuitively, one would expect such models will have a better forecasting performance than the aforementioned three algorithms.

Because such models take into account the strong periodic patterns as well as the auto-correlation among the hourly traffic load. Below we propose one such time series forecasting model using illustration with the traffic load observed at AP 472.

Suppose *Xi (t)* is the traffic load within hour *t* at a particular AP *i* (e.g., AP 472). Due to the nature of the wireless network traffic, $X_{472}(t)$ has local spikes that are very hard to predict as illustrated in Figure 5.4.1(a). In addition, it most likely has a skewed marginal distribution. Figure 5.4.2(a) plots the normal quantile plot of $X_{472}(t)$ for AP 472, which clearly suggests the marginal distribution of the traffic load is heavily skewed to the right. This calls for a suitable transformation to make the data closer to a normal distribution. Such a transformation can reduce the effect of those local spikes on the forecasting performance. In addition, standard time series modeling procedures are most suitable for situations with normal data [47]. After experimenting with different transformations, the 1/4 power transformation, $Y(t) = X_{472}(t)^{1/4}$, seems to give the best result. In particular, Figure 5.4.2(b) gives the normal quantile plot for the transformed load *Y(t)* at AP 472. As one can see, *Y(t)* is much closer to be normally distributed, and does not have extreme outliers as those in Figure 5.4.2(b). The following model will be performed on *Y(t)*.

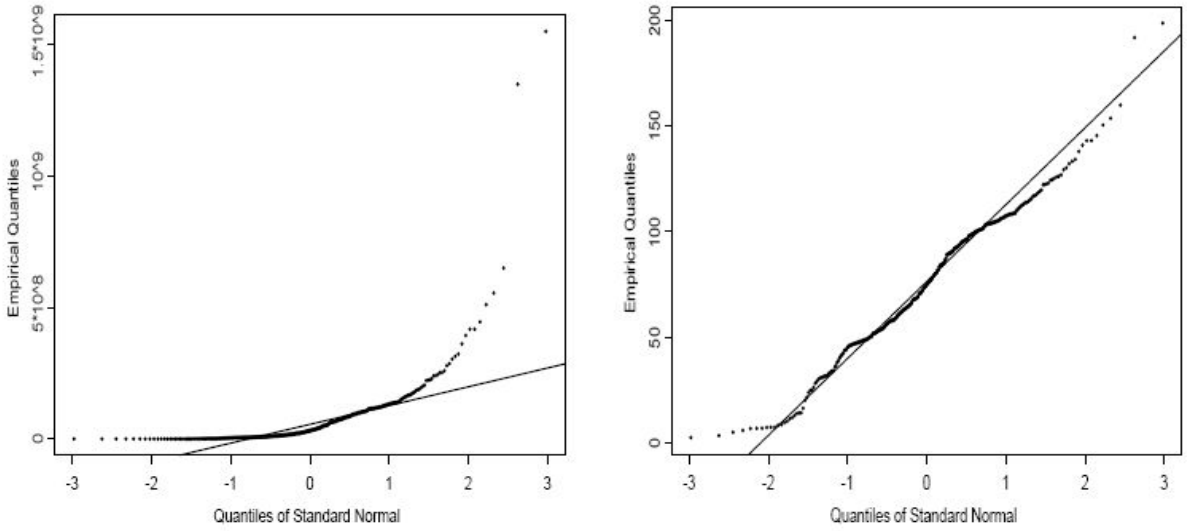

Figure 5.4.2. Traffic load at AP 472: (a) normal quantile plot for X472 (t) (b) normal quantile plot for Y(t).

We first point out that *Y(t)* exhibits strong non-stationarity in both the mean and the variance. Figure 5.4.3(a) plots the bimodal changing patterns of its mean, median, 25-th percentile and 75-th percentile as functions of hour-of-day *(h(t)),* which shows that both the mean and the percentiles change across the day. For example, the mean curve suggests that there is very little traffic between midnight and 7-8AM; then the load starts to increase until it reaches the first mode around 10AM and stays flat until noon; after lunch-break, the load increases again to the second mode around 3PM before it starts to decrease until midnight. Very sensible explanations can be given for such a diurnal pattern. Similarly, Figure 5.4.3 (b) indicates the diurnal patterns for the standard deviation and Inter Quartile Range (*IQR*) *(i.e., the difference between the 25-th and 75-th percentiles).* The plot suggests that there is increasing variability in the traffic load during 7AM-10AM and 1PM-3PM, exactly when the load increases. In addition, the variability stays small between 10AM and 1PM.

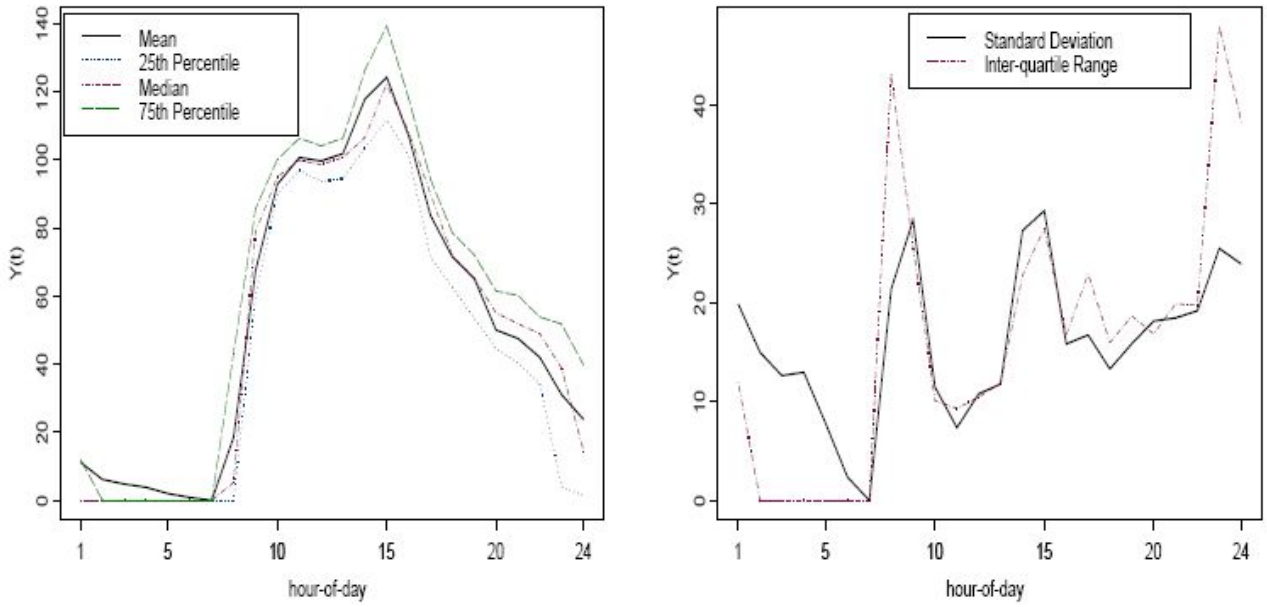

Figure 5.4.3. (a)Changing patterns of mean, median, and quartiles of Y(t)
(b) Changing patterns of standard deviation (SD) and inter-quartile range of Y(t).

The above exploratory data analysis motivates us to normalize the transformed load *Y(t)* in the following way,

$$e(t) = \frac{Y(t) - \mu_{h(t)}}{\sigma_{h(t)}}$$

where $h(t)$ is the corresponding hour-of-day for time t, $\mu_{h(t)}$ is the mean of $Y(t)$ during those time periods with the hour-of-day being $h(t)$ while $\sigma_{h(t)}$ is the standard deviation of $Y(t)$ during those time periods, and $e(t)$ can be treated as a normalized version of $Y(t)$. Note that $\mu_{h(t)}$ and $\sigma_{h(t)}$ have been plotted in Figure 5.4.3 (a) & (b) for AP 472.

After the normalization, we can assume $e(t)$ to be a stationary time series as shown in Figure 5.4.4(a). The corresponding partial autocorrelation function (Partial *ACF*) (Figure 5.4.4(b)) suggests that an *AR(1)* model is reasonable for the normalized time series, $e(t)$. Thus, we fit a family of *AR(p)* models to $e(t)$ using the Yule-Walker method and select the approximate order p by minimizing the Akaike Information Criterion (*AIC*). See Brockwell and Davis (1998) for details about the estimation method and the model selection criterion, AIC. Note that the order p specifies the number of lagged variables in the time series model and the *AR(p)* model is written as

$$e(t) = a_1 e(t-1) + \ldots + a_p e(t-p) + n(t),$$

where $n(t)$ is the model residual.

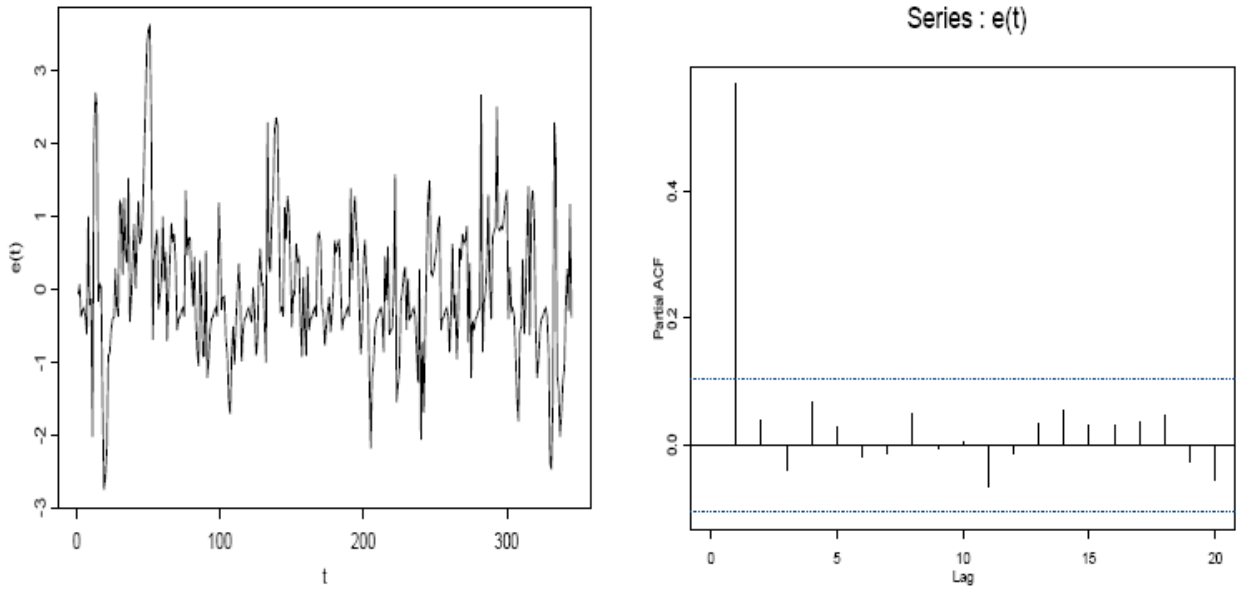

Figure 5.4.4. (a) Time series for e(t) (b) Partial ACF plot for e(t).

As for the load at AP 472, p is selected to be 1 and the fitted *AR(1)* model is

$$e(t) = 0.5689e(t-1) + n(t) \qquad (1)$$

with the residuals *n(t)* being normally distributed with mean 0 and variance 0.6349. One can then use (1) to predict the traffic load during the next hour, corresponding to time $(t + 1)$, $X_{472}(t)$. First, a point prediction for $e(t+1)$ can be obtained as

$$e(t+1) = 0.5689e(t)$$

then $Y(t + 1)$ can be predicted as

$$\hat{Y}(t+1) = \mu_{h(t+1)} + \sigma_{h(t+1)} \times \hat{e}(t+1).$$

Finally, a point forecast for $X_{472}(t + 1)$ is obtained by back-transforming $Y(t + 1)$,

$$\hat{X}(t+1) = \hat{Y}^4(t+1)$$

In general, our proposed time-series forecasting approach can be summarized as follows:

1) Transform the load in a reasonable way to make the data more normally distributed. Note that the transformation here is subjectively chosen, and it seems to be working well in the current application.
2) Investigate time-varying patterns of the mean and variability of the transformed load.
3) Normalize the transformed load if the mean and variability are indeed time-varying.
4) Develop standard time series models like *AR(p)* for the normalized series, and employ rigorous model selection procedures like *AIC* to select the optimal model.
5) Perform one-step-ahead or multi-step-ahead forecasting on the normalized series using the fitted model, and then back-transform the forecast to the original scale.

5.5 Evaluation of the performance of the forecasting algorithms

A. Metrics: absolute/ relative prediction error ratio and percentage of correct predictions

To evaluate the performance of the prediction algorithms, we compute the *absolute prediction error* defined as the absolute difference of the predicted from the actual traffic and the *relative prediction error* which is the ratio of this absolute error over the actual traffic. The relative prediction error *r(t)* at an AP *i* during the *t-th* time interval is defined as $r(t) = |Z_I^K(t) - X_I(t)| / X_I(t)$. A perfect prediction algorithm has absolute and relative prediction error equal to 0. The relative prediction error is a conservative metric, since it does not take into account the intervals with zero actual traffic, which are frequent in the trace. Furthermore, large relative prediction errors indicate large over or under-estimations of traffic. To better appreciate the performance of the forecasting algorithms, both the relative and absolute prediction errors need to be considered. A good prediction algorithm should have low absolute and relative prediction errors. The mean (median) relative prediction error of all hotspots is the average of the mean (median) relative prediction error considering all hotspots. Similarly, we compute the mean and median absolute prediction error. For the evaluation of the forecasting algorithms, we use the aforementioned metrics.

In the case of periodic based forecasting, the prediction algorithms apply a predicted interval based on the historical mean and a tolerance (or precision) error level. Specifically, we define the *e-tolerance prediction interval from a mean* μ to be the interval $[(1 - \varepsilon) * \mu, (1 + \varepsilon) * \mu]$. The prediction algorithm computes the percentage of times that the actual traffic is in the predicted interval. For example, in the case of the prediction P_k, $k = 1,2,3$, for the traffic of AP *i* during the *h-th* hour of day *d,* it computes the prediction interval $[(1-\varepsilon)*Z_i^k(h,d), (1+\varepsilon)*Z_i^k(h,d)]$ and checks if $X_i(t)$ is in that interval.

A good prediction algorithm should have a high correct prediction percentage and low prediction error ratio. A large prediction error ratio indicates large prediction estimates and may result in conservative prediction and resource underutilization.

B. Forecasting using historical means and recent traffic (P1,P2, P3)

For all the aforementioned prediction algorithms, we computed the means based on the history for each AP. The history corresponds to three weeks of the trace, excluding weekends and starting on Monday, October 18th, 2004. We predict the traffic for each AP, for all the hours during the weekdays of the following week (Monday, November 8th until Friday, November 12th). We call this period *forecasting period.*

Algorithm	Type	Trace	Tracing period	Forecasting period	Scale
P1 P2	Periodic Periodic	SNMP-aggregate	3 weeks excl. weekends start: Mon 18/10/04	5-day start: Mon 11/8/04	hour
P3 P-MA-RG	Hybrid	SNMP-aggregate	(same as above)	(same as above)	hour
NAMSA	Hybrid	SNMP-aggregate	(same as above)	(same as above)	5 min, 1 hour
MA fixed window size=2,3,4	Recent history	SNMP-aggregate	no training	(same as above)	hour
MA,EMA fixed window size	Recent-history	SNMP-aggregate	no training	1 day start Wed 9/29/05	5 min
Adaptive EMA (α=0.4)	Recent history	SNMP-aggregate	no training	1 day Wed 9/29/05	5 min
Type-of-flow	multi-traced	SNMP-client, TCP headers	no training	2-day start 4/14/05	5 min
Number-of-flows	multi-traced	SNMP-client, TCP headers	one day, Wed 4/13/05	2-day, start: Thu 4/14/05	5 min

Table 7. Summary of the forecasting algorithms with the type of traces used, their tracing and forecasting period and time scale.

For P3, we varied the recent history window size to be 2, 3, 4, and 5 hours. We evaluated P3 for various values of a,b, and c, including also values resulted from applying multiple linear regression for each AP.

Figures 5.5.1 and 5.5.2 show the histograms of the percentage of correct predictions for the P1, P2, and P3 considering all APs.

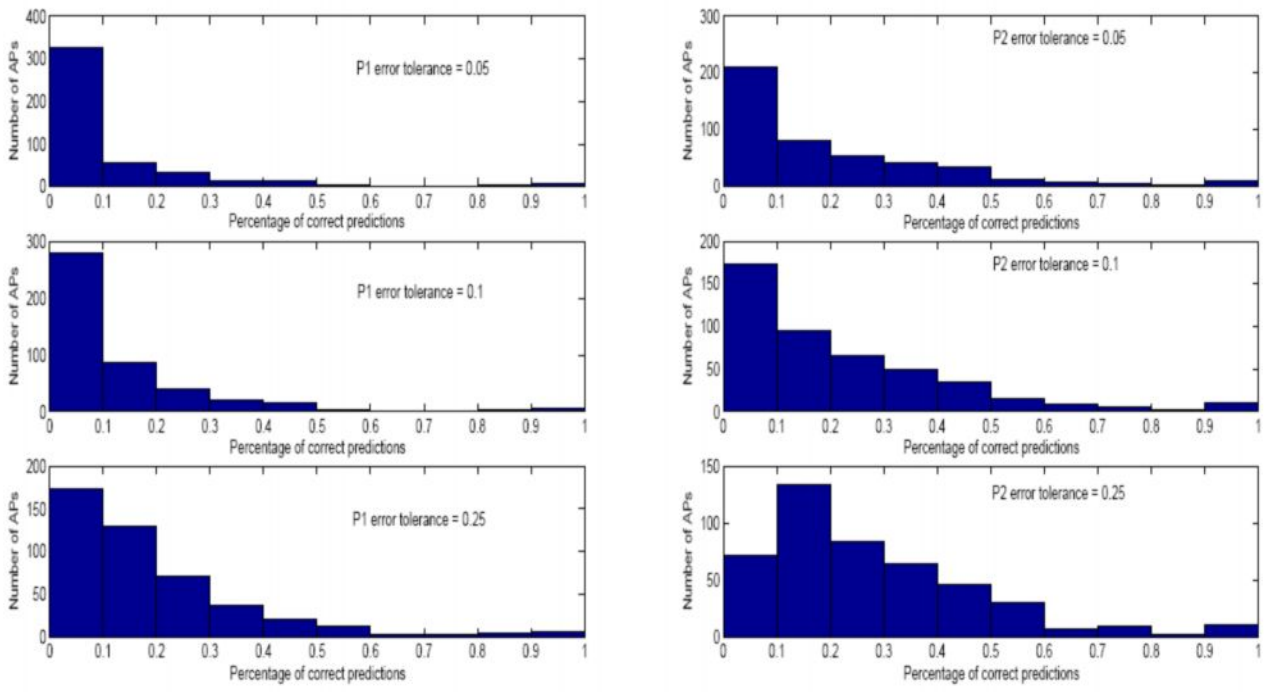

Figure 5.5.1. Performance of prediction algorithms P1, P2 considering all APs.

P3 outperforms P2 and P1 with respect to the correct predictions percentage. P3 also outperforms P2 and P1 with respect to the correct predictions percentage, when we only consider the hotspots. Specifically, for a window of two hours and (a,b,c) equal to (1,0,0), P3's percentage of correct predictions for a 25%-tolerance prediction interval has a (mean, median, std. deviation) equal to (34.17%, 24.17%, 22.86%).

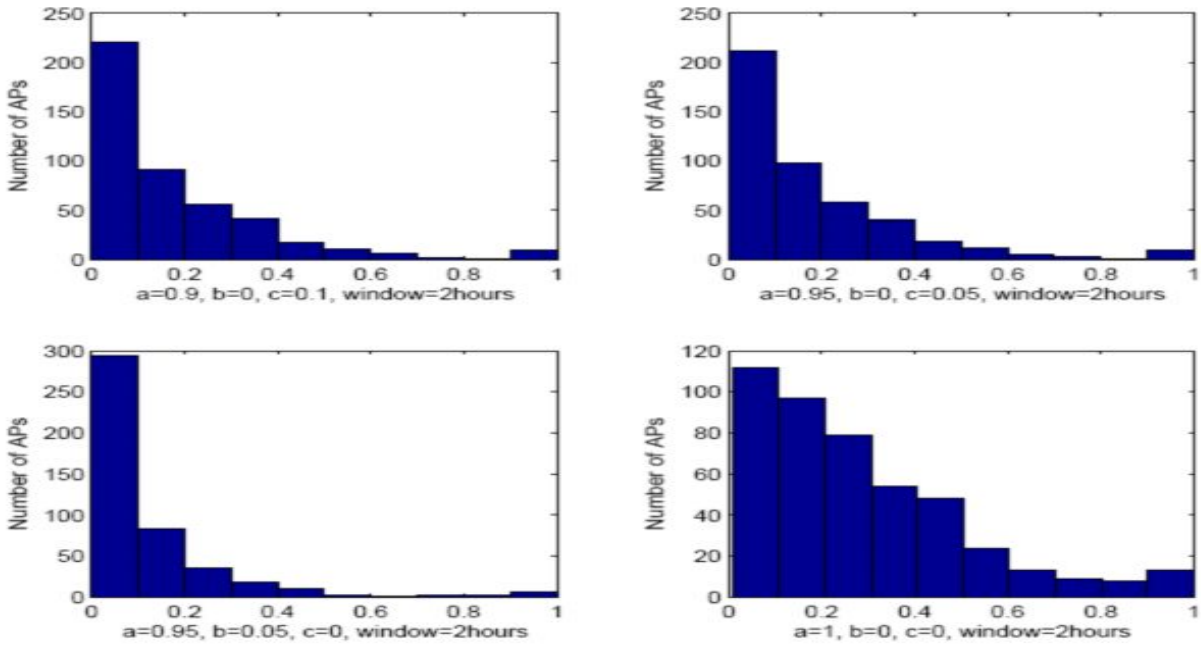

Figure 5.5.2. Performance of prediction algorithm P3 considering all APs, with 25% error tolerance.

The *mean percentage of correct predictions* of hotspots for an e-tolerance is the average of the percentages of correct predictions for that e-tolerance considering all hotspots. The *mean prediction error ratio* of hotspots is the average of the mean prediction error ratios considering all hotspots. In the same manner, we compute their median and std. deviation. For the same e-tolerance, P2 has a lower percentage of correct predictions than P3 but higher than P1 (for both median and mean prediction of correct percentages). Similarly, the median prediction error ratio for P3 is lower than for P1 and P2 (see Figure 5.5.4). On the other hand, P3's mean prediction error ratio is lower than P1's and higher than P2's one. The high mean prediction error ratio of P1, P2, and P3 are due to the high variability in the traffic.

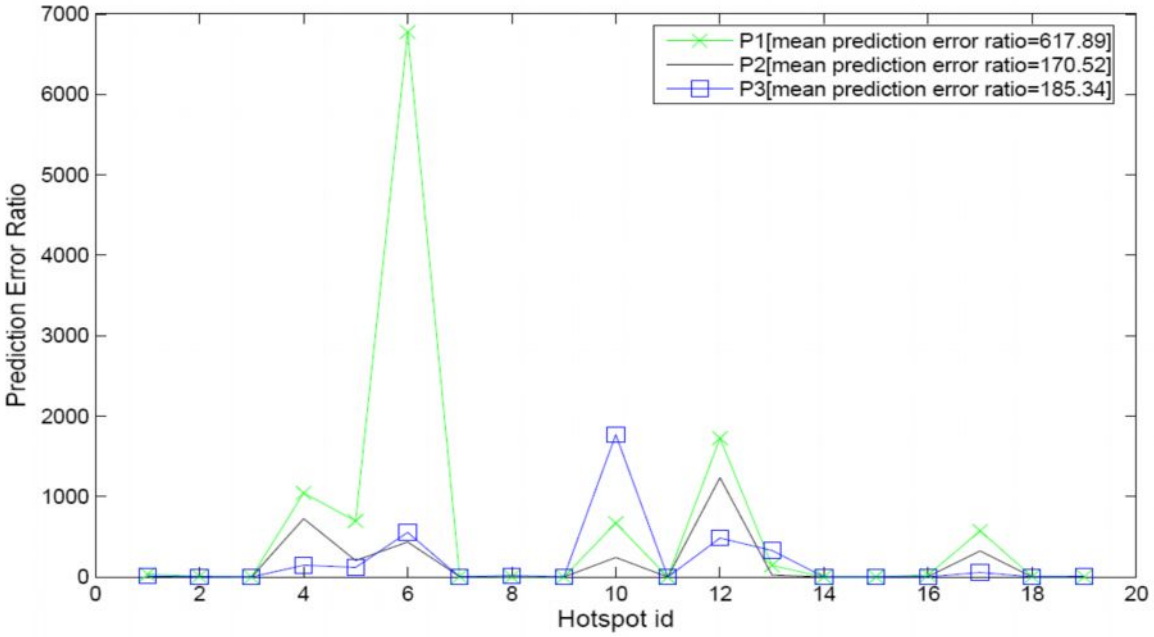

Figure 5.5.3. Mean prediction ratios for P1, P2, and P3 with weights (a,b,c)=(1,0,0) for each hotspot.

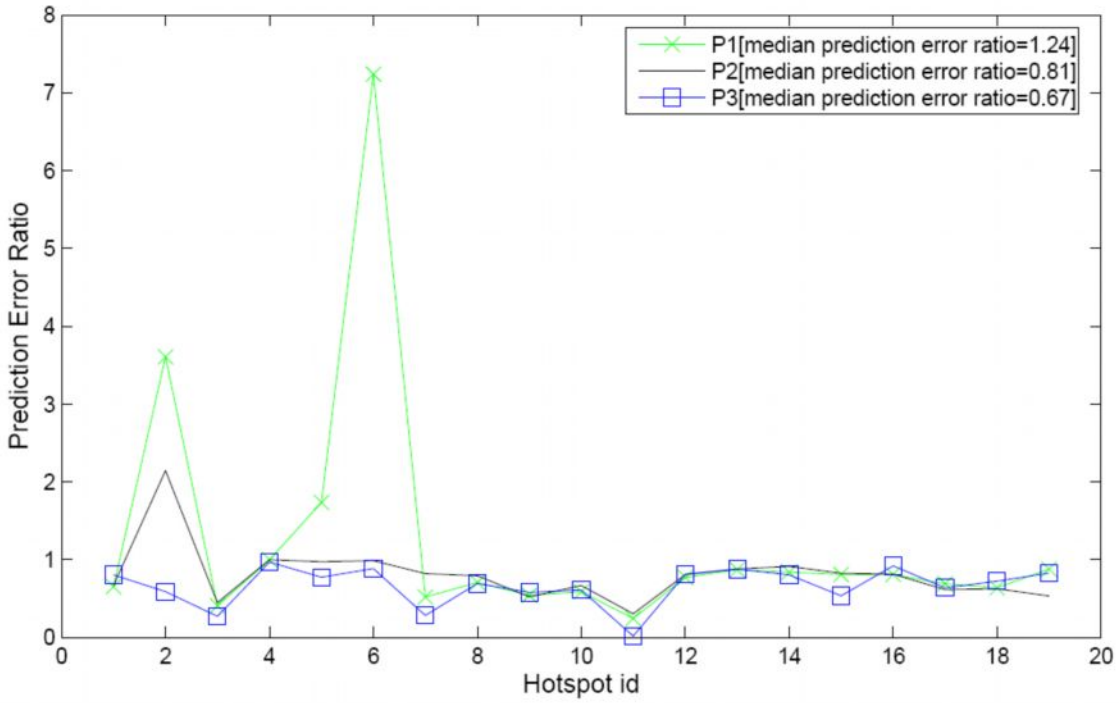

Figure 5.5.4. Median prediction ratio for P1, P2 and P3 with weights (a,b,c)=(1,0,0) for each hotspot.

C. Normalized ARIMA multi-step ahead time-series forecasting (NAMSA)

Using the same 3-week data (as in the other prediction algorithms), this normalized ARIMA multi-step ahead time series forecasting performs as follows. As Figures 5.5.5 and 5.5.6 illustrate, the prediction error ratio of the AP 472 (hotspot id 18) has a mean, median, and SD of 1.42, 0.72, and 3.77, respectively. Its correct percentages are 17.5%, 9.17%, and 6.67%, for a 25%, 10%, and 5%- tolerance prediction interval, respectively. The corresponding percentages for P1 are 20%, 10% and 6.67%, and forP2 20%, 18.33%, 16.67%, respectively. For a 25%-tolerance prediction interval, P3 with a two-hour window size and (a, b, c)=(1, 0, 0) has a 24.17% correct prediction percentage.

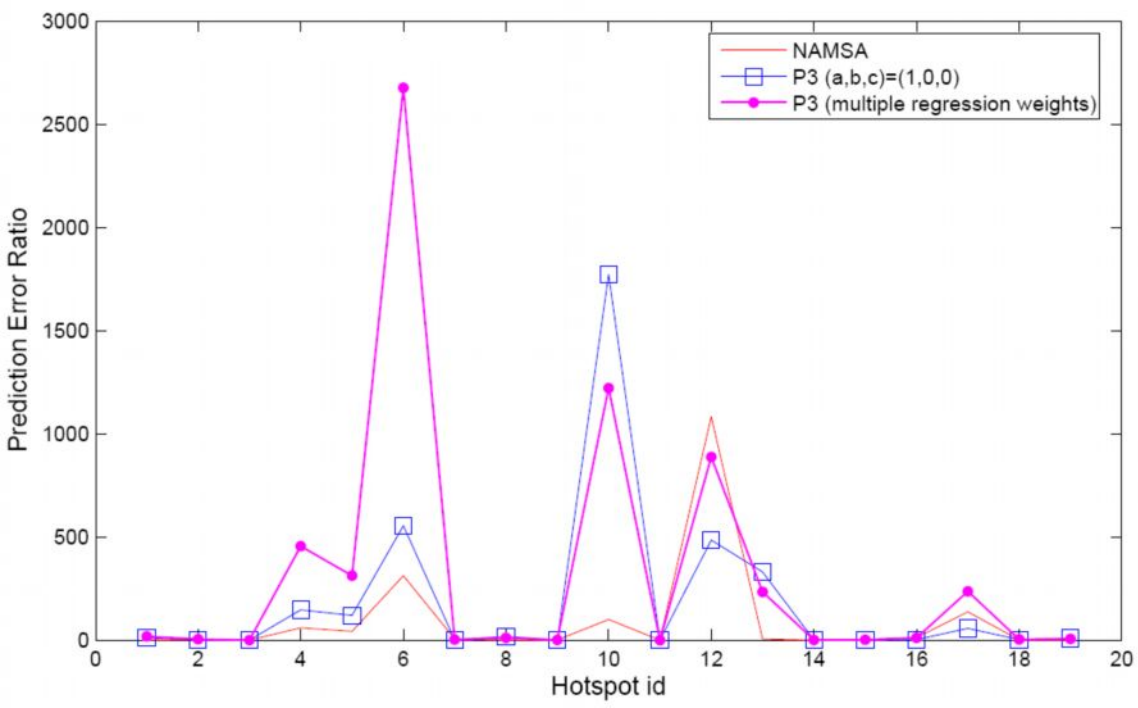

Figure 5.5.5. Mean prediction ratio for the P3 and NAMSA forecasting algorithms for each hotspot.

We apply the NAMSA algorithms to the 19 hotspots APs and the result is compared with the three aforementioned algorithm below. Note that the order of the *AR(p)* model is adaptively selected using *AIC* for each AP separately. Figures 5.5.5 and 5.5.6 illustrate the mean and median prediction error ratio of the P3 with weights (a,b,c)= (1,0,0), P3 with weights fitted using multiple linear regression, and NAMSA forecasting

algorithm for all hotspots. Compared to the simple prediction algorithms P1, P2, and P3, the NAMSA algorithm results in better values for the mean and the SD of the error ratio (Figure 5.5.5). On the other hand, the median of its error ratio is a bit worse than that of the P3 algorithm (Figure 5.5.6). This forecasting algorithm is a *multi-step-ahead* forecasting. That is, to predict a value, apart from the traffic model, the multi-step-ahead forecasting uses the recent *predicted* values instead of the actual ones. This makes the prediction even harder than the one-step ahead forecasting that uses the *actual* recent values like P3. We expect better performance when we use this algorithm for one-step ahead forecasting.

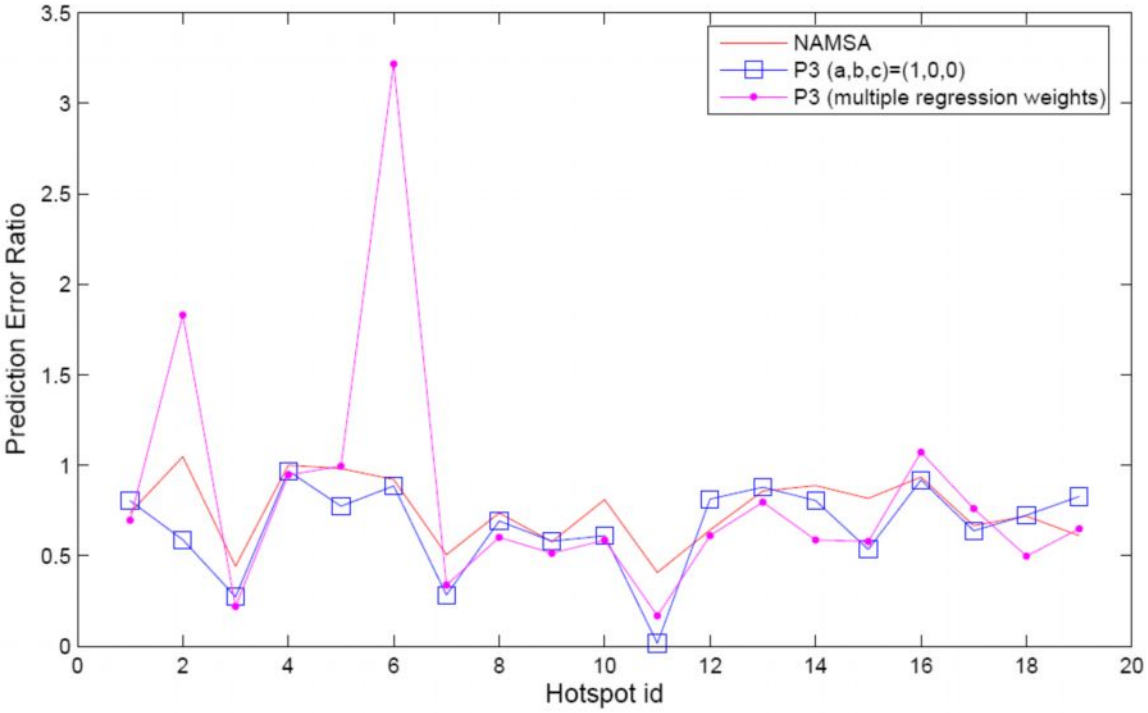

Figure 5.5.6. Median prediction ratio for the P3 and NAMSA forecasting algorithms for each hotspot.

Note that P3 with weights fitted using multiple linear regression performs worse than P3 and NAMSA (with respect to both mean and median error ratio). This is due to the difference in the metrics used: The prediction error ratio is the ratio of the absolute difference of the predicted from the actual traffic over the actual traffic, whereas the multiple regression minimizes the square difference. When we use as metric the difference of

the predicted from the actual traffic in square, we can observe that the mean of the overall improvement of P3 (with multiple regression) for hotspots reaches 26%. Furthermore, we found that the dominant regressor in the weighted sum of P3 is history (for all hotspots). Specifically, in average, the recent history predictor participates in P3 with a percentage of 43.8% while historical mean hour and historical mean hour-of-day percentages are 41.1% and 15.1%, respectively.

D. Improvement in forecasting performance using finer time scales

The performance of the MA and EMA when forecasting in finer time scales (in 5minute traces) has been improved dramatically. To evaluate the algorithms, we considered one-day from the original snmp-aggregate data in 5-minute intervals and run variations of the MA and EMA algorithms. We were particularly interested in testing the adaptivity of the adaptive-EMA algorithm, and for this, we select an $a = 0.4$ and 95%-confidence intervals, for the level-shift detection criteria. The adaptive EMA reports a relative prediction error of 2.95 and 0.46, for its mean and median considering all hotspots and 5-minute interval predictions, respectively. The mean and median absolute errors are 3.33MB and 0.75MB. This algorithm performs much worse on hourly traffic time-series.

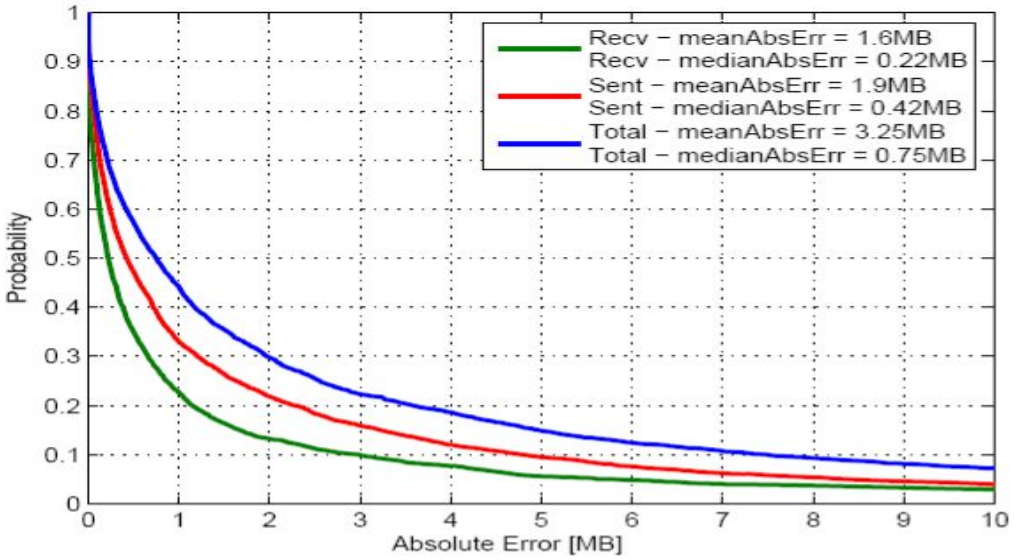

Figure 5.5.7. Absolute error for EMA with five-minute traces.

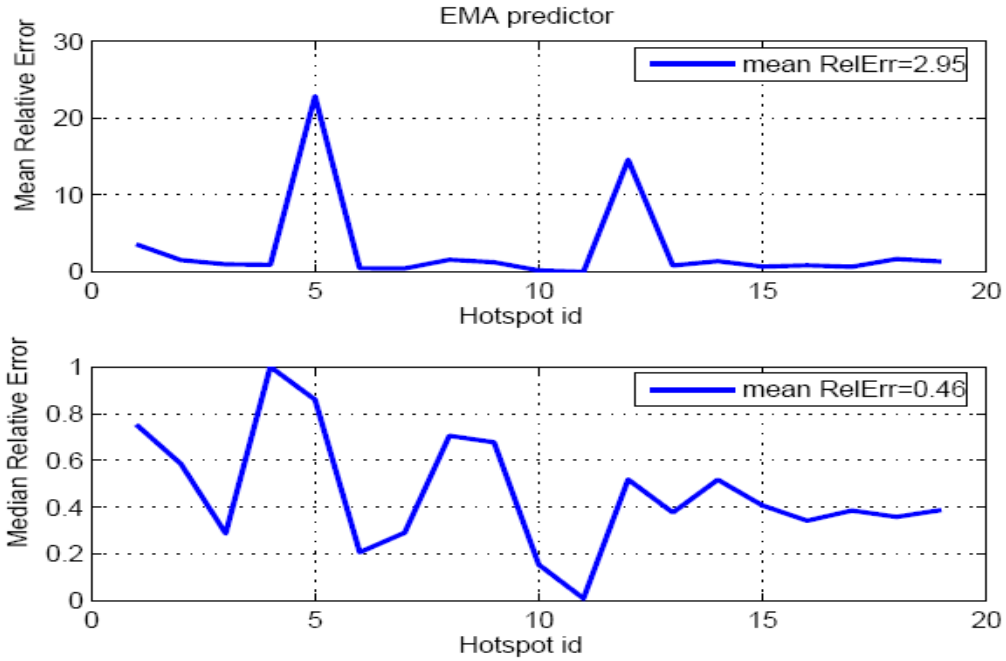

Figure 5.5.8. Relative error for EMA with five-minute traces.

The algorithm first forecasts the received and sent traffic at each five-minute interval. Based on these forecasts, it reports as the total traffic for that five-minute interval, the sum of the predicted receive and sent. Notice that its performance is one or two orders of magnitude better than P1's, P2's, and P3's performance on hourly based time-series. Although, several APs have strong diurnal patterns, the performance analysis reveals that the more "recent" the historical data and the finer their time scale, the stronger its impact on forecasting. Figures 5.5.7 and 5.5.8 illustrate the performance of adaptive EMA when it first predicts receive and sent traffic (from the corresponding recent history measurements) and then forecasts as total traffic the sum of the predicted receive and sent. We found that the improvement in forecasting the total traffic "indirectly" (i.e., through the forecasts of the received and sent traffic) over forecasting directly the five-minute time-series of total traffic is negligible. On the other hand, the adaptive mechanism in MA or EMA does not show any improvement when it is applied on hourly time-series. For example, the adaptive MA has a mean relative prediction error 3213.90 and a median relative prediction error 0.78 which is larger than the prediction errors of simple MA-based

models. As Figures 5.5.9 and 5.5.10 illustrate, the NAMSA on 5-minute traces has slightly better medians but worse means in both the absolute and relative prediction errors. However, the improvement in the performance of NAMSA on 5-minute traces over the one on hourly time intervals is distinct. In both cases, NAMSA was applied on traces that correspond to the same period.

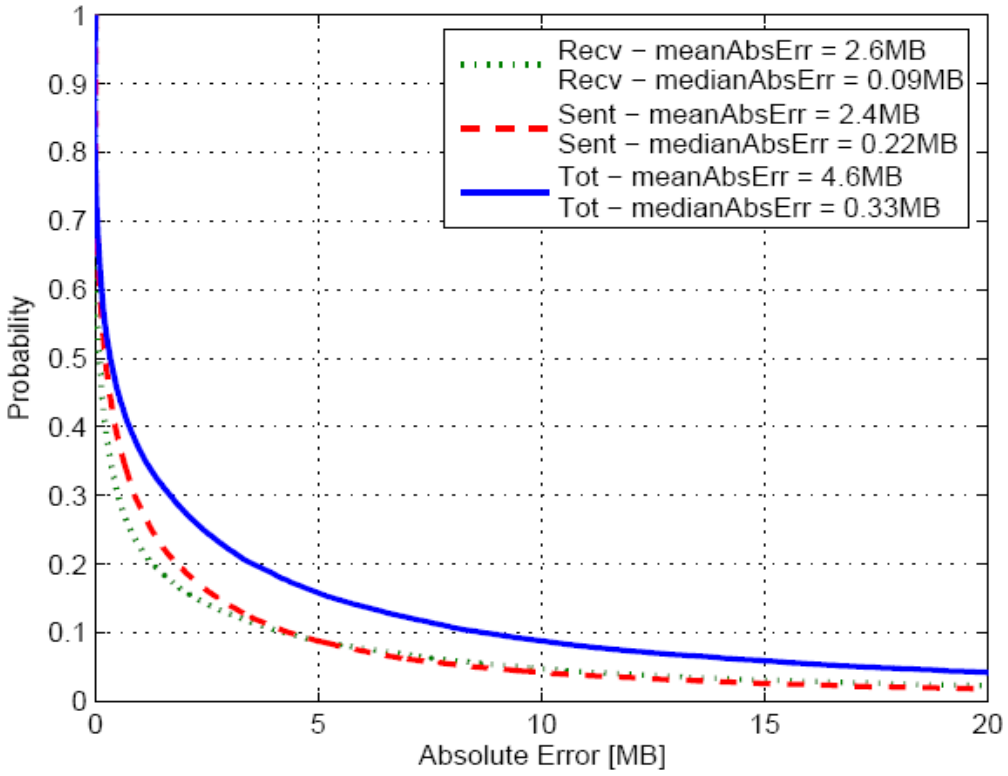

Figure 5.5.9. Absolute error for NAMSA with five-minute traces.

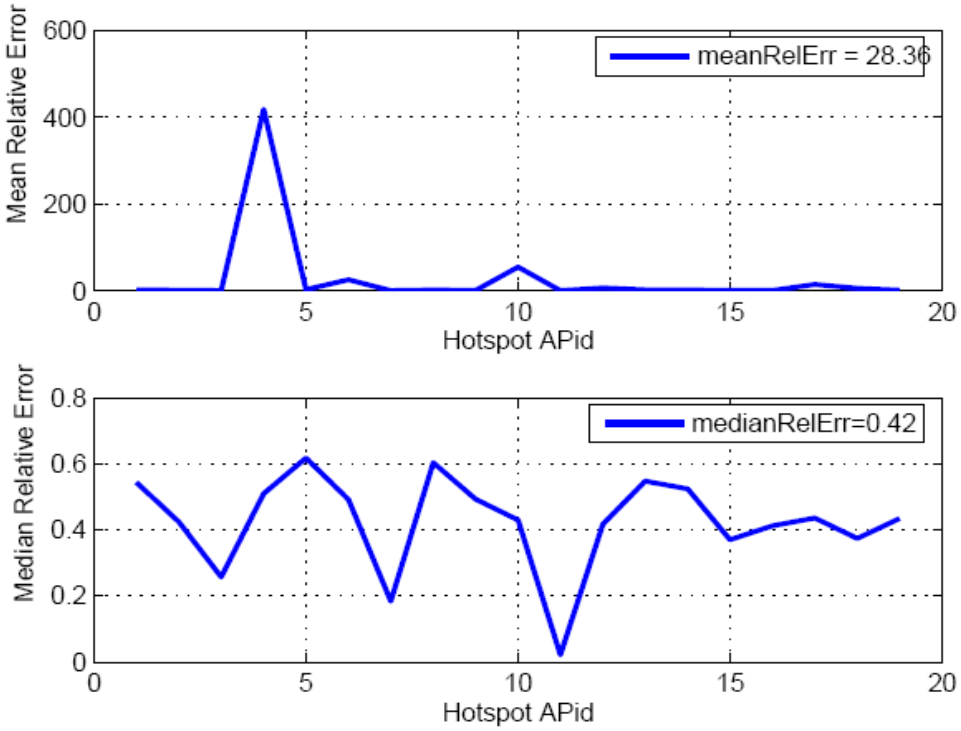

Figure 5.5.10. Relative error for NAMSA with five-minute traces.

Compared to the aforementioned algorithms that require some statistical analysis of the traffic, the flow-based algorithms presented in Section 5.4-C.V are simpler and require less processing overhead. We used the SNMP client-based and TCP-flow based data sets with one-day training period to create the 5-minute interval times series of traffic load and number of flows at each AP and then fit the parameters of

$$\hat{X}(k) = e^{b}(N_{flows}(k-1))^{a}$$

We evaluated the number-of-flows algorithm on the 5-minute-interval trace of the next two days. The (median, mean) pairs of the relative and absolute prediction errors are (0.89,19.68) and (0.23MB, 4.24MB), respectively.

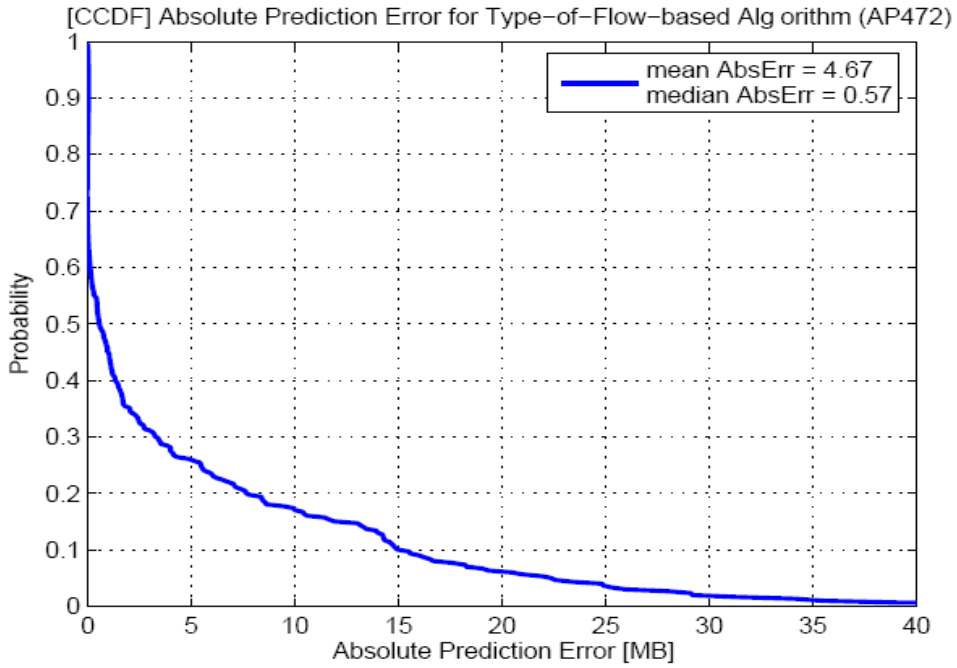

Figure 5.5.11. Absolute prediction error for the type-of-flow based algorithm on hotspot AP 472.

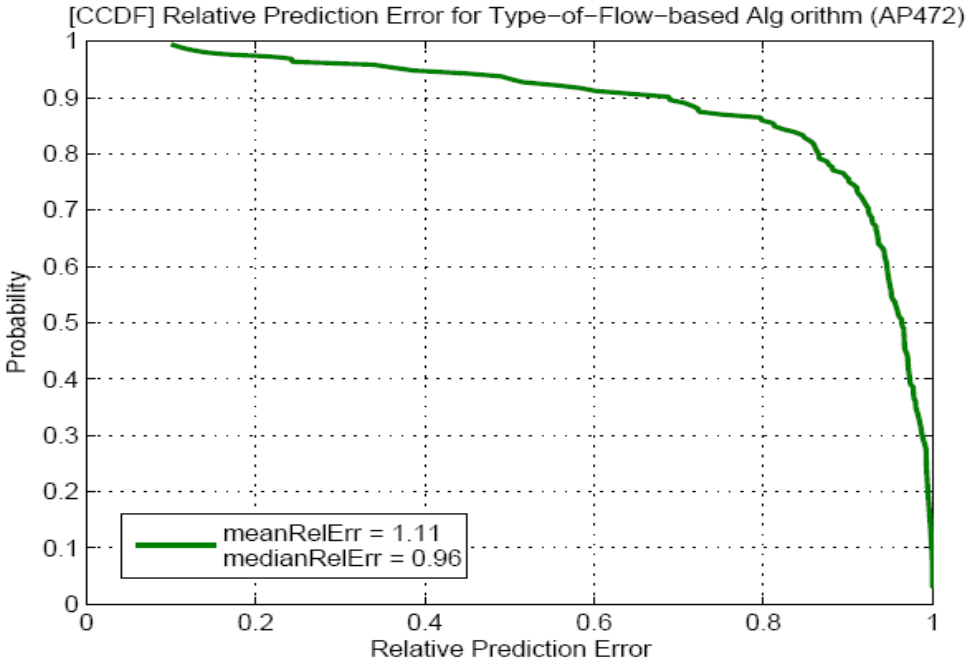

Figure 5.5.12. Relative prediction error for the type-of-flow based algorithm on hotspot AP 472.

The type-of-flow based algorithm applied on a 5-minute traces for the hotspot AP 472 has a median and mean absolute prediction error of 4.67MB (0.02MB) and 0.57MB (0.48MB), respectively. The median and mean relative prediction error of 0.96 (1) and 1.11 (30.33). In parenthesis, we indicate the performance of the number-of-flow based algorithm in 5-minute intervals for the same tracing period and AP. Figures 5.5.11 and 5.5.12 illustrate the CCDF of the absolute and relative errors when the type-of-flow algorithm was used on hotspot AP 472, respectively.

5.5 Conclusions

We designed a number of forecasting algorithms and evaluated their performance on the hotspots of a large production wireless network. Short-term forecasting on wireless networks is challenging but the use of finer time-scale can improve dramatically the mean prediction errors. We investigated the impact of recent history and periodicities of traffic, dependencies of number and type of flows. The algorithms achieve a median absolute and relative prediction error of 0.33MB and 0.42 respectively, considering all hotspot APs in 5-minute intervals.

We also noticed in the APs' traffic some hours with unexpectedly low (compared to the historical means) values. In this section, we proceeded with the prediction without pre-processing these values. A more rigorous approach is to impute those entries with some estimates, such as the mean traffic load during the same hour-of-day from the other days. We expect that it will improve the prediction performances of the algorithms and plan to investigate this further. We intend to study more systematically the spatial correlations of APs and classify APs based on various parameters (e.g., traffic characteristics, building type, number of associations, and distinct clients). Furthermore, we aim to explore the impact of the above parameters and spatial correlations on forecasting. Finally, our goal is to simulate load balancing mechanisms that facilitate these forecasting algorithms to evaluate their impact on the performance of the network.

6. Conclusions

Wireless networks grow in response to the increasing demand for wireless access. In the same time, more features are added to them to enable support of more demanding applications and efficient management of their resources. The availability of precise and scalable models for both the system-wide and more local-scale study of these networks is a fundamental requirement for their engineering. Moreover, traffic load forecasting estimates can be used by APs to not only better manage their traffic demand but also advice clients to associate with the appropriate APs to better utilize their local resources. Such predictions can be used to reduce the energy spendings at the client side, improve the capacity utilization of wireless LANs, and better load balance the traffic. We address this requirement in the context of traffic demand modeling and forecasting.

In this work, we introduced a novel methodology for modeling the wireless access and traffic demand by providing a multilevel perspective: it models the arrival and size of sessions and flows at systems-wide and AP levels. It investigates their statistical properties, dependencies and inter-relations. It shows the stationarity of the number of flows and flow inter-arrival in a session. Most of the modeling efforts have been on the AP-level. The shift to sessions and flows has gained two important advantages: Sessions at an AP can mask the network-related dependencies that are not important in a range of applications and simulation environments, such as brief transitions from one AP to another due to a transient behavior of the signal, and exhibit nice statistical properties (such as stationarity) that makes them amenable to modeling.

We have proposed a hierarchical modeling framework for traffic demand in large wireless LANs. We first confirmed that the same distributions apply when we look at traffic at finer spatial scales, such as the building level or all buildings having similar usage. We promote the building as the primary entity for traffic demand modeling at the spatial dimension. Working at building level circumvents several problems emerging when working at AP-level: non amenability to statistical processing, higher sensitivity of monitored traffic variables to the short-term propagation

conditions, lack of scalability. We emphasize this last aspect, considering ways to come up with models that scale with the size of the network whilst preserving modeling efficiency. To this end, we proposed both heuristic and more formal clustering techniques to group buildings for traffic demand modeling purposes. By way of example, we showed that both of them can benefit the traffic modeling task. Interestingly, the two approaches are complementary. Heuristical segregation of buildings scores better with flow-related variables, but is not as efficient with the modeling of session arrivals, where the requirement is to group buildings depending on size rather than shape. Clustering combined with PCA and SVD behaves better there, even if the interpretation is less intuitive than with the heuristical alternative.

Moreover, we designed a number of forecasting algorithms based on the knowledge acquired from the modeling task, and evaluated their performance on the hotspots of a larger production wireless network. We investigated the impact of recent history and periodicities of traffic, dependencies of number and type of flows, and tailored different types of forecasting algorithms using multiple sources of data. Our ultimate goal is to design and develop admission control, capacity planning and load balancing tools incorporating these forecasting mechanisms.

REFERENCES

1. A. Adya, P. Bahl, R. Chandra, and L. Qiu. Architecture and techniques for diagnosing faults in ieee802.11 infrastructure networks. In *ACM International Conference on Mobile Computing and Networking* (MobiCom), Philadelphia, September 2004.
2. Ishwar Ramani and Stefan Savage. SyncScan: Practical fast handoff for 802.11 infrastructure networks. In Proceedings of the IEEE Conference on Computer Communications (Infocom), Miami, FL, March 2005.
3. Arunesh Mishra, Minho Shin, and William A. Arbaugh. An empirical analysis of the IEEE 802.11 MAC layer handoff process. April 2003.
4. T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campuswide wireless network. In *Proceedings of ACM MobiCom*, Philadelphia, PA, USA, September 2004.
5. M. Balazinska and P. Castro. Characterizing mobility and network usage in a corporate wireless local-area network. In *Proceedings of MobiSys*, San Francisco, CA, USA, May 2003.
6. A. Balachandran, G. Voelker, P. Bahl, and V. Rangan. Characterizing user behavior and network performance in a public wireless LAN. In *Proceedings of ACM Sigmetrics*, CA, June 2002.
7. Diane Tang and Mary Baker. Analysis of a local-area wireless network. In ACM International Conference on Mobile Computing and Networking (MobiCom), pages 1-10, Boston, Massachusetts, USA, August 2000.
8. Magdalena Balazinska and Paul Castro. Characterizing mobility and network usage in a corporate wireless local-area network. In First International Conference on Mobile Systems, Applications, and Services (MobiSys), San Francisco, USA, May 2003.
9. Anand Balachandran, Geoffrey Voelker, Paramvir Bahl, and Venkat Rangan. Characterizing user behavior and network performance in a public wireless lan. In Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, California, USA, 2002.
10. David Kotz and Kobby Essien. Analysis of a campus-wide wireless network. Technical Report TR2002-432, Dept. of Computer Science, Dartmouth College, September 2002.
11. Tristan Henderson, David Kotz, and Ilya Abyzov. The changing usage of a mature campuswide wireless network. In ACM International Conference on Mobile Computing and Networking (MobiCom), Philadelphia, September 2004.

12. Xiaoqiao Meng, Starsky Wong, Yuan Yuan, and Songwu Lu. Characterizing flows in large wireless data networks. In ACM International Conference on Mobile Computing and Networking (MobiCom), pages 174-186, Philadelphia, 2004.
13. Maria Papadopouli, Elias Raftopoulos, and Haipeng Shen. Evaluation of short-term traffic forecasting algorithms in wireless networks. In 2nd Conference on Next Generation Internet Design and Engineering, Valencia, Spain, 2006.
14. Amiya Bhattacharya and Sajal K. Das. LeZi-update: an information-theoretic approach to track mobile users in PCS networks. In Proceedings of the Annual ACM/IEEE International Conference on Mobile Computing and Networking, pages 1–12, Seattle, Washington, USA, August 1999.
15. Maria Papadopouli, Haipeng Shen, and Manolis Spanakis. Characterizing the duration and association patterns of wireless access in a campus. In 11th European Wireless Conference, Nicosia, Cyprus, April 2005.
16. Cristian Tuduce and Thomas Gross. A mobility model based on wlan traces and its validation. In Proceedings of the IEEE Conference on Computer Communications (Infocom), Miami, FL, March 2005.
17. Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, and James Scott. Impact of human mobility on the design of opportunistic forwarding algorithms. In Proceedings of the IEEE Conference on Computer Communications (Infocom), Barcelona, Spain, April 2006.
18. Minkyong Kim and David Kotz. Modeling users' mobility among wifi access points. In WiTMeMo '05, Berkeley, CA, USA, June 2005. USENIX Association.
19. Ravi Jain, Dan Lelescu, and Mahadevan Balakrishnan. Model T: an empirical model for user registration patterns in a campus wireless lan. Cologne, Germany, August 2005.
20. A. Jardosh, E. M. Belding-Royer, K. C. Almeroth, and S. Suri. Towards realistic mobility models for mobile ad hoc networks. In ACM International Conference on Mobile Computing and Networking (MobiCom), San Diego, CA, September 2003.
21. D. Eckhardt and P. Steenkiste. Measurement and analysis of the error characteristics of an in building wireless network. ACM Computer Communication Review, 26(4):243–254, October 1996.
22. Felix Hernandez-Campos and Maria Papadopouli. Assessing the real impact of 802.11 wlans: A large-scale comparison of wired and wireless traffic. In 14th IEEE Workshop on Local and Metropolitan Area Networks, Chania, Greece, September 2005.
23. Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In ACM

International Conference on Mobile Computing and Networking (MobiCom), San Diego, CA, September 2003.
24. Sanjit Biswas and Robert Morris. Opportunistic routing in multi-hop wireless networks. In SIGCOMM Symposium on Communications Architectures and Protocols, Philadelphia, PA, August 2005.
25. Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. Link-level measurements from an 802.11b Mesh network. In SIGCOMM Symposium on Communications Architectures and Protocols, August 2004.
26. John Bicket, Daniel Aguayo, Sanjit Biswas, and Robert Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In ACM International Conference on Mobile Computing and Networking (MobiCom), Cologne, Germany, August 2005.
27. Krishna Ramachandran, Elizabeth Belding, Kevin Almeroth, and Milind Bud-dhikot. Interference-aware channel assignment in multi-radio wireless mesh networks. In Proceedings of the IEEE Conference on Computer Communications (Infocom), Barcelona, Spain, April 2006.
28. Henrik Lundgren, Krishna Ramachandran, Elizabeth Belding-Royer, Kevin Almeroth, Michael Benny, Andrew Hewatt, Alexander Touma, and Amit Jar-dosh. Experiences from the design, deployment, and usage of the UCSB Mesh-Net testbed. IEEE Wireless Communications, April 2006.
29. Self-organizing neighbourhood wireless mesh networks. http://research.microsoft.com/mesh/.
30. Roofnet is an experimental 802.11b/g mesh network in development at MIT. http://pdos.csail.mit.edu/roofnet/doku.php.
31. P. Bahl, R. Chandra, and J. Dunagan. Ssch: Slotted seeded channel hopping for capacity improvement in ieee 802.11 ad-hoc wireless networks. Philadelphia, PA, September 2004.
32. R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. Philadelphia, PA, September 2004.
33. R. Draves, J. Padhye, and B. Zill. Comparison of routing metrics for static multi-hop wireless networks. Portland, OR, August 2004.
34. w. e. Leland, m. s. Taqqu, w. Willinger, and d. v. Wilson. On the self-similar nature of ethernet traffic. ACM Computer Communication Review, 25(1):202–213, 1995.
35. W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through high-variability: Statistical analysis of ethernet lan traffic at the source level. ACM Computer Communication Review, 25(4):100–113, October 1995.
36. Mark Crovella and Azer Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. In Proceedings of SIGMETRICS '96, 1996.

37. "Cisco Aironet AP specifications," Product information sheet. [Online]. Available: http://www.cisco.com/en/US/products/hw/wireless
38. X. G. Meng, S. H. Y. Wong, Y. Yuan, and S. Lu. Characterizing flows in large wireless data networks. In *Proc. of ACM MobiCom*, New York, ny, United States, 2004, pp. 174–186.
39. F. Hernandez-Campos, M. Karaliopoulos, M. Papadopouli, and H. Shen. Spatio-temporal modeling of traffic workload in a campus wlan. In Second Annual International Wireless Internet Conference, Boston, ma, USA, 2006.
40. M. Karaliopoulos, E. Raftopoulos, M. Papadopouli, and H. Shen. On scalable measurement-driven modeling of traffic demand in large WLANs. ICS-FORTH, Heraklion, Greece, Tech. Rep. 383, July 2006.
41. L. D. Brown, N. Gans, A. Mandelbaum, A. Sakov, H. Shen, S. Zeltyn, and L. Zhao, "Statistical analysis of a telephone call center: a queueing-science perspective," Journal of the American Statistical Association, vol. 100, no. 469, pp. 36-50, March 2005.
42. C. Nuzman, I. Saniee, W. Sweldens, and A. Weiss, "A compound model for TCP connection arrivals for LAN and WAN applications," Computer Networks, vol. 40, no. 3, pp. 319–337, 2002.
43. F. Hernandez-Campos, M. Karaliopoulos, M. Papadopouli, and H. Shen, "Spatio-temporal modeling of traffic workload in a campus WLAN," in Second Annual International Wireless Internet Conference, Boston, MA, USA, 2006.
44. M. Karaliopoulos, E. Raftopoulos, M. Papadopouli, and H. Shen, "On scalable measurement-driven modeling of traffic demand in large WLANs," ICS-FORTH, Heraklion, Greece, Tech. Rep. 383, July 2006. [Online]. Available: http://www.ics.forth.gr/ftp/tech-reports/2006/2006.TR383_WLANScalable_Traffic_Modeling.pdf
45. h. c. Rosemburg, Cluster Analysis for Researchers. Belmont, ca, United States: Lifetime Learning Publications, 1984.
46. r. Mojena, "Hierarchical grouping methods and stopping rules: An evaluation." Computer Journal, vol. 20, p. 359363, 1977.
47. P.F. Brockwell and R.A. Davis. Time Series: Theory and Methods. New York: Springer-Verlag, New York, 1998.

APPENDIX

SeperateTrace2APFiles.pl

```
#!/usr/local/bin/perl

# ------------------------------------------------------------------------------
# USAGE
# ------------------------------------------------------------------------------

my ($progname) = $0 =~ /([^\/]+)$/;

my $usage=<<EOF;
Description:
    Helper function used to split original data to AP files
EOF


# ------------------------------------------------------------------------------
# PRAGMAS
# ------------------------------------------------------------------------------

    use strict;

# ------------------------------------------------------------------------------
# MAIN
# ------------------------------------------------------------------------------

#Number of Input Files - Trace File Days

    my $input_files_no = 63;
    my $file_num;
    my $filename;

    for $file_num( 1 ... $input_files_no ) {

        $filename = 'd'.$file_num.'.dat';
        print 'Trying to open :'."$filename"."\n";
        open ( DATA, "$filename" );

        while (<DATA>) {
            next if (/^\s*$/); # skip empty lines

            # Parse client polling line

            my (    $poll_time, $ap_num, $ap_up_time, $speed, $bytes_recv,
                $uni_pkts_recv, $multi_pkts_recv, $pkts_recv_not_delivered,
                $pkts_recv_discarded, $bytes_sent, $uni_pkts_sent, $multi_pkts_sent,
                $pkts_sent_not_deliv, $pkts_sent_discarded, $num_assoc, $num_authent,
                $num_roams_in, $num_roams_away, $num_deauthent, $num_disassoc ) = split;

            die "cannot parse line\n[$_]" if ($poll_time eq "");

            my $APfilename;
            $APfilename = 'ap'.$ap_num;

            if( -e $APfilename ){
                open( OUT, ">>$APfilename") || die "cannot create $APfilename: $!";
            }
            else{
                open( OUT, ">$APfilename") || die "cannot create $APfilename: $!";
```

```
            }
            print OUT $_;
            close( OUT ) || die "cannot close $APfilename: $!";
        }
        close( DATA );
    }
```

clt2anon-clt.pl

```
#!/usr/local/bin/perl

# ------------------------------------------------------------------------
# USAGE
# ------------------------------------------------------------------------

my ($progname) = $0 =~ /([^\/]+)$/;

my $usage=<<EOF;
Description:
    Anonymize SNMP client data.
Usage:
    $progname [options] list-of-APs list-of-clients < snmp-clt-data
e.g.,
    list-of-APs:     ~fhernand/src/wireless/misc/ap_list/unc.may25.ap_list
    list-of-clients: ~fhernand/src/wireless/misc/clt_MACS.jun2.txt.gz
    this is the initial location of files, now all AP-related and client-related files
    are at the UNC-infrastructure directories
Options:
    -h      Show this help message.
EOF

# ------------------------------------------------------------------------
# PRAGMAS
# ------------------------------------------------------------------------

use strict;

# ------------------------------------------------------------------------
# MAIN
# ------------------------------------------------------------------------

#
# OPTIONS
#

use Getopt::Std;

my (%opts);
if (not getopts("h", \%opts) or $opts{"h"}) {
    print "$usage";
    exit;
}

my $AP_fname = $ARGV[0];
die "cannot find AP list '$AP_fname'" if (not -e $AP_fname);

my $clt_fname = $ARGV[1];
die "cannot find clt table '$clt_fname'" if (not -e $clt_fname);

#
# Read list of APs and load the AP_IP to APnum mapping on the
```

```
# ap_ip2num hash
#
my %ap_ip2num;

open AP_LIST, "$AP_fname" || die "cannot open $AP_fname";
while (<AP_LIST>) {
    next if (/^\#/);
    chomp;
    my ($AP_Number, $Name, $IP, $OS, $Version,
    $LAN_MAC, $Radio_MAC, $Channel) = split;

    if (exists $ap_ip2num{$IP}) {
        die "New Num $AP_Number != " . $ap_ip2num{$IP}
        if ($AP_Number != $ap_ip2num{$IP});
    } else {
        $ap_ip2num{$IP} = $AP_Number;
    }
}

#
# Read list of clients from the single-column file with client MAC addresses
# Load the mapping on the clts hash
#

my %clts; # key: IP; value: AP_type

open CLT_LIST, "$clt_fname" || die "cannot open $clt_fname";
my $num_clts = 1;
while (<CLT_LIST>) {
    next if (/^\#/);
    chomp;
    $clts{"$_"} = $num_clts++;
}
close CLT_LIST || die "cannot close $clt_fname";

my @new_clts;

#
# Anon line - here do the anonymization for both fields - AP IP and client MAC address
#

while (<STDIN>) {
    chomp;
    my @fields = split;
    # $fields[1] ap
    # $fields[3] clt
    die "cannot find AP $fields[1]" if (not exists $ap_ip2num{$fields[1]});
    $fields[1] = $ap_ip2num{$fields[1]};
    if (not exists $clts{$fields[3]}) {
        $clts{$fields[3]} = $num_clts++;
        push @new_clts, $fields[3];
    }
    $fields[3] = $clts{$fields[3]};
    print "@fields[0..($#fields - 1)]\n";
}

#
# Update list of clients (append new MAC-to-ClientNum assignments)
#

if (@new_clts > 0) {
    open CLT_LIST, ">>$clt_fname" || die "cannot append to $clt_fname";
    my $mac;
```

```
    # ClientNum is given by the row number, starting at 1
    foreach $mac (@new_clts) {
        print CLT_LIST "$mac\n";
    }
    close CLT_LIST || die "cannot close $clt_fname after appending";
}
```

ap_ts2tot_traf.pl

```
#!/usr/local/bin/perl


# ------------------------------------------------------------------------------
# USAGE
# ------------------------------------------------------------------------------

my ($progname) = $0 =~ /([^\/]+)$/;

my $usage=<<EOF;
Transform original time series to final traffic format
EOF

# ------------------------------------------------------------------------------
# PRAGMAS
# ------------------------------------------------------------------------------

use strict;

# ------------------------------------------------------------------------------
# OPTIONS
# ------------------------------------------------------------------------------

use Getopt::Std;

my (%opts);
if (not getopts("h", \%opts) or $opts{"h"}) {
    print "$usage";
    exit;
}

# ------------------------------------------------------------------------------
# MAIN
# ------------------------------------------------------------------------------

    my $AP2Bldg_fname = 'apmapbldg2.txt';
    open AP2Bldg_LIST, "$AP2Bldg_fname" || die "cannot open $AP2Bldg_fname";

    my $B_CNT = 1; my %Bldgs; my %APs;
    while(<AP2Bldg_LIST>){
        next if (/^\#/);
        chomp;
        # APIP Building Building_type
        my( $AP_Number, $IP, $B_ID, $B_TYPE ) = split;
        if(exists $Bldgs{$B_ID}) {
            #already mapped
        }
        else{
            $Bldgs{$B_ID} = $B_CNT++;
        }
        $APs{$AP_Number} = $Bldgs{$B_ID};
    }
    close AP2Bldg_LIST;
```

```
    my $ap_mapping = 'ap2bnum.txt';
    open OUT_AP_LIST, ">$ap_mapping" || die "cannot append to $ap_mapping";
    for my $key ( keys %APs ) {
        my $value = $APs{$key};
        print OUT_AP_LIST "$key $value\n";
    }
    close OUT_AP_LIST || die "cannot close $ap_mapping after appending";

    my $bldg_mapping = 'bldgid2bnum.txt';
    open OUT_B_LIST, ">$bldg_mapping" || die "cannot append to $bldg_mapping";
    for my $key ( keys %Bldgs ) {
        my $value = $Bldgs{$key};
        print OUT_B_LIST "$key $value\n";
    }
    close OUT_B_LIST || die "cannot close $bldg_mapping after appending";
```

ts2ts.pl

```
#!/usr/local/bin/perl

#
# Felix Hernandez Campos
#
# February-March 2005
#

# ------------------------------------------------------------------------------
# USAGE
# ------------------------------------------------------------------------------

my ($progname) = $0 =~ /([^\/]+)$/;

my $usage=<<EOF;
Description:
   Change bin length in time-series.
Usage:
   $progname [options] bin-size < tr.ts
Options:
   -h           Show this help message.
EOF

# ------------------------------------------------------------------------------
# PRAGMAS
# ------------------------------------------------------------------------------

use strict;

# ------------------------------------------------------------------------------
# MAIN
# ------------------------------------------------------------------------------

#
# OPTIONS
#

use Getopt::Std;

my (%opts);
if (not getopts("h", \%opts) or $opts{"h"} or @ARGV != 1) {
  print "$usage";
  exit;
```

```
}

my $binlen = $ARGV[0];

print STDERR "Bin length: $binlen\n";

my $next_bin = $binlen;
my $tot = 0;

while (<STDIN>) {

    if (/^\#/) {
        print $_;
    }
    else {
        my ($ts, $value) = split;
        while ($next_bin < $ts) {
            print "$next_bin $tot\n";
            $tot = 0;
            $next_bin += $binlen;
        }
    $tot += $value;
    }
}
```

APTotalTrafficPerDayAndHour.pl

```
#!/usr/local/bin/perl

# Code Based on
# Felix Hernandez Campos
#
# November 2004
#
# Versions:
#   1.0 11/2/2004 10 PM: First version
#   1.1 11/3/2004 11 AM: Removed absolute value comparison
#    2.0 17/5/2005 7PM : Changed function's define_day_and_hour implementation

#    Modifications by Elias Raftopoulos

# ------------------------------------------------------------------------------
# USAGE
# ------------------------------------------------------------------------------

#my ($progname) = $0 =~ /([^\/]+)$/;

my $usage=<<EOF;
Get hourly traffic data for all APs
EOF

# ------------------------------------------------------------------------------
# PRAGMAS
# ------------------------------------------------------------------------------

use strict;

# ------------------------------------------------------------------------------
# MAIN
# ------------------------------------------------------------------------------
```

```
#
# OPTIONS
#

use Getopt::Std;

my (%opts);
if (not getopts("dh", \%opts) or $opts{"h"}) {
  print "$usage";
  exit;
}

#Thresholds used
my $clt_up_time_threshold = 20;
my $deltas_threshold = 20;

#Number of Input Files
my $input_files_no = 14710;
#Duration of traces in days
my $no_of_days = 58;

# Previous line data
my $prev_line; # just for better error messages
my $prev_clt_num;
my $prev_poll_time;
my $prev_ap_num;
my $prev_clt_up_time;
my $prev_bytes_sent; # just for sanity check
my $prev_bytes_recv; # just for sanity check

#Initial timestamp based on CL SNMP data
my $cl_SNMP_init_time = 1096463101;
#my $syslog_init_time = 1096430465;

############
my $row;
my $col;
my  @hour_day;
my @matrix;
#Initialize output matrix
my $maxAPID = define_maxID( $input_files_no );

print 'maxAPID found'."$maxAPID"."\n";
for $row ( 1 .. $no_of_days*24 ){   #days * hours_per_day
    for $col ( 1 .. $maxAPID ){
        $matrix[ $row ][ $col ] = 0;
    }
}

my $file_num;
my $filename;

for $file_num( 1 ... $input_files_no ) {

    $filename = 'sortedClient'. $file_num.'.txt';
    print 'Trying to open :'."$filename"."\n";
    open ( DATA, "$filename" );

############
    while (<DATA>) {
        next if (/^\s*$/); # skip empty lines

        print "INPUT: $_" if ($opts{"d"});
```

```
        # Parse client polling line

        my ($poll_time, $ap_num, $ap_up_time, $clt_num, $clt_up_time,
        $clt_state, $dbm, $power_save,
        $pkts_recv, $bytes_recv, $pkts_sent, $bytes_sent,
        $dups, $retrans_msdu, $undelivered_msdu) = split;

        die "cannot parse line\n[$_]" if ($poll_time eq "");

        if ($prev_clt_num eq "" or $prev_clt_num ne $clt_num) {

            # New client => first association
########
            @hour_day = define_day_and_hour( $poll_time );
            $row = $hour_day[0];
            $matrix[$row][$ap_num] += ($bytes_sent + $bytes_recv);
########
            print "NEW CLIENT\n" if ($opts{"d"});
            print "$clt_num $ap_num $poll_time\n"if ($opts{"d"});
        }
        elsif ($prev_ap_num eq "" or $prev_ap_num ne $ap_num) {

            # New AP => new association
########
            @hour_day = define_day_and_hour( $poll_time );
            $row = $hour_day[0];
            $matrix[$row][$ap_num] += ($bytes_sent + $bytes_recv);
########
            print "NEW AP\n" if ($opts{"d"});
            print "$clt_num $ap_num $poll_time\n" if ($opts{"d"});
        }
        else {

            # Same client
            my $polling_delta = $poll_time - $prev_poll_time;
            die "polltime going backward. Was the input trace properly sorted?\n$prev_line$_" if
($polling_delta < 0);

            # Detect associations
            if ($clt_up_time - $polling_delta <= -$clt_up_time_threshold) {

            # Association
########
                @hour_day = define_day_and_hour( $poll_time );
                $row = $hour_day[0];
                $matrix[$row][$ap_num] += ($bytes_sent + $bytes_recv);
########
                print "ASSOCIATION: CLT UP TIME BELOW THRESHOLD\n" if ($opts{"d"});
                print "$clt_num $ap_num $poll_time\n" if ($opts{"d"});
            }
            elsif ($clt_up_time - $polling_delta >= $clt_up_time_threshold) {
            # No Association
            print "NO ASSOCIATION: CLT UP TIME ABOVE TRESHOLD\n" if ($opts{"d"});

            if ($bytes_sent < $prev_bytes_sent ) {
                    if ($prev_bytes_sent >= 3*2**30 and # prev close to 2^32
                                                            $bytes_sent <= 2**30) { # current
close to 0
                        # wrap around is ok => do not output an ERROR
########
                        @hour_day = define_day_and_hour( $poll_time );
                        $row = $hour_day[0];
```

```
                        $matrix[$row][$ap_num]  +=  2^32  -  1  -  $prev_bytes_sent  +  $bytes_sent  -
$prev_bytes_recv + $bytes_recv;
########
                    }
                    else {
                        warn "ERROR: wrong sorting? cisco bug?\n$prev_line$_"
                    }
            }
########
            else{
                # No Association - normal flow
                @hour_day = define_day_and_hour( $poll_time );
                    $row = $hour_day[0];

                    if( ($bytes_recv - $prev_bytes_recv ) < 0 ){    #cisco bug in recv counter
                    $matrix[$row][$ap_num] += ($bytes_sent - $prev_bytes_sent);
                }
                else{
                    $matrix[$row][$ap_num]  +=  ($bytes_sent  -  $prev_bytes_sent)  +  ($bytes_recv  -
$prev_bytes_recv);

                    #Debug
                    my $test = ($bytes_sent - $prev_bytes_sent) + ($bytes_recv - $prev_bytes_recv);
                    if( $test < 0 ){
                        print "Bug Found FILE:$filename\n";
                        print "INPUT: $_";
                        print "PREV INPUT: $prev_line";
                    }
                    #endDebug
            }
            }
########
            }
            else
            {
            my $clt_up_time_delta = $clt_up_time - $prev_clt_up_time;

            if ($polling_delta - $clt_up_time_delta >= $deltas_threshold) {
                # Association
########
                    @hour_day = define_day_and_hour( $poll_time );
                    $row = $hour_day[0];
                    $matrix[$row][$ap_num] += ($bytes_sent + $bytes_recv);
########
                    print "ASSOCIATION: HIGH CLT UP TIME DELTA\n" if ($opts{"d"});
                    print "$clt_num $ap_num $poll_time\n" if ($opts{"d"});
            }
            elsif ($clt_up_time_delta - $polling_delta >= $deltas_threshold) {
                    die "Unexpected negative difference between deltas";
            }
            else {
                    print "UNCLEAR => ASSUME NO REASSOCIATION\n" if ($opts{"d"});
            }
            }
        }
        $prev_line = $_;
        $prev_clt_num = $clt_num;
        $prev_poll_time = $poll_time;
        $prev_ap_num = $ap_num;
        $prev_clt_up_time = $clt_up_time;
        $prev_bytes_sent = $bytes_sent;
        $prev_bytes_recv = $bytes_recv;
    }#while (<DATA>) closing here
```

```
    close( DATA );
}#for $file_num( 1 ... $input_files_no ) closing here

############
#write matrix data to output file

open( OUT, ">APTrafficPerDayAndHourInit.txt");

for $row ( 1 .. $no_of_days*24 ){
    for $col ( 1 .. $maxAPID ){
        print OUT $matrix[ $row ][ $col ]." ";
    }
    print OUT "\n";
}
############


sub define_maxID{

    my $maxAPID = 0;
    my $input_files = $_[ 0 ];
    my $file_num;

    for $file_num( 1 ... $input_files ) {

        $filename = 'sortedClient'. $file_num.'.txt';
        open ( DATA, "$filename" );

        while (<DATA>) {
            next if (/^\s*$/); # skip empty lines

            print "INPUT: $_" if ($opts{"d"});

            # Parse client polling line

            my ($poll_time, $ap_num, $ap_up_time, $clt_num, $clt_up_time,
            $clt_state, $dbm, $power_save,
            $pkts_recv, $bytes_recv, $pkts_sent, $bytes_sent,
            $dups, $retrans_msdu, $undelivered_msdu) = split;

            die "cannot parse line\n[$_]" if ($poll_time eq "");

            if( $maxAPID < $ap_num ){
                $maxAPID = $ap_num;
            }
        }
    }
    return $maxAPID;
}

sub define_day_and_hour{

    my $curr_poll = $_[ 0 ];
    my $poll_hour;
    my $poll_day;
    my $error_log = 'ERROR_LOG.txt';
    my $poll_hour_final;

    $poll_hour = (int (($curr_poll - $cl_SNMP_init_time )/3600));
    $poll_day = (int $poll_hour/24) + 1;
    $poll_hour_final = $poll_hour - ($poll_day-1)*24 + 1;

    #some sanity checking...
```

```
    if( $poll_hour > $no_of_days*24 or $poll_hour < 0 ){
        if( -e $error_log ){
            open( ERRORLOG, ">>$error_log") || die "cannot create $error_log: $!";
        }
        else{
            open( ERRORLOG, ">$error_log") || die "cannot create $error_log: $!";
        }
        print                   ERRORLOG                  'ERROR::Invalid                   timestamp
'.$curr_poll."[hour=".$poll_hour.",day=".$poll_day."]\n";
        print     'ERROR::Invalid     timestamp'."[hour=".$poll_hour.",day=".$poll_day."]\n"     if
($opts{"d"});
        close( ERRORLOG ) || die "cannot close $error_log: $!";
        $poll_hour = -1;
        $poll_day = -1;
    }
    return ($poll_hour,$poll_day);
}
```

corelate_cvec.pl

```
#!/usr/local/bin/perl

# ------------------------------------------------------------------------
# USAGE
# ------------------------------------------------------------------------

my ($progname) = $0 =~ /([^\/]+)$/;

my $usage=<<EOF;
Description:
    Correlating SNMP client data and cvec data.

Usage:
    $progname [options] snmp-session-file start-trace < cvec-file > flow-info

Input format:
    - snmp-session-file is the sorted output of clt_data2session.v4.pl, filtered
    for single client IP sessions with
    grep "^s" | grep -v "," | grep " 152\.[12]" | sort +9 -10 +3n -4 -s
    - cvec-file is the output of ascii-file processing with tcp2cvec

Options:
    -c  Conservative connection duration estimation.
    -d  Debugging mode.
    -h      Show this help message.
    -i       3rd rule (associations inside connections). Requires -c.
EOF

# ------------------------------------------------------------------------
# PRAGMAS
# ------------------------------------------------------------------------

use strict;

# ------------------------------------------------------------------------
# IMPORTS
# ------------------------------------------------------------------------

#die "Environment variable \$SRC undefined or incorrect"
#  if ($ENV{SRC} eq "");
#require "$ENV{SRC}/stats/cdf.pl";
require "./stats/cdf.pl";
```

```
# --------------------------------------------------------------------------
# MAIN
# --------------------------------------------------------------------------
#
# OPTIONS
#

use Getopt::Std;

my (%opts);
if (not getopts("cdhi", \%opts) or $opts{"h"} or @ARGV != 2) {
    print "$usage";
    exit;
}

my $snmp_fname = $ARGV[0];
die "cannot find $snmp_fname" if (not -e $ARGV[0]);
if ($ARGV[0] =~ /\.gz$/) {
    open SNMP_FILE, "zcat $ARGV[0] |";
} else {
    open SNMP_FILE, "$ARGV[0]";
}

open DFILE, ">out.debug-session-cvec";

my $start_trace = $ARGV[1];

my $poll_int = 300; # 5 minutes
my $delta_poll = 25; # conservative adjustment (due to req/rsp losses)

my $snmp_line;

my $snmp_clt_ip;
my $snmp_clt_mac;
my $ap_ip;

my @session_set; # list of hashes storing basic information about session (start/end point, id)
           # for a single client. The list is emptied (undefined) upon change of client
my $session_num = 1;

#(MK) : the extra allow a flow in the cvec file to finish at some point beyond the last
# SNMP polling interval that a session of the same client was seen, as long as this interval
# does not exceed the one POLLING INTERVAL
my @correlated_sessions_stats_certain;
my @correlated_sessions_stats_extra; # end of SNMP assoc + poll_int
my @correlated_sessions_stats_all; # end of SNMP assoc + poll_int

# (MK) variables logging stats
my ($flow_not_correlated, $flow_correlated, $flow_session_not_found,
    $snmp_clt_not_correlated, $snmp_clt_correlated);

# (MK) part of the output file format
my ($lan_ip, $lan_port, $wan_ip, $wan_port,
    $start_ts, $end_ts, $start_ts_lan, $end_ts_lan,
    $tot_pkts, $tot_bytes,
    $tot_pkt_lost, $rtt_mad, $rtt_sd, $term);

my %num_sessions_per_clt;

while (not eof STDIN) {
    &read_snmp_clt; # read all assoc info for one client (MK : store all his sessions in the list
of hashes session_set)
```

```
    &process_clt_flows; # process client if the next one in flow info
                # is same as above, skip or advance otherwise
}

print STDERR "STATS:\n\n";
print STDERR "Uncorrelated flows (missing clt): $flow_not_correlated\n";
print STDERR "Uncorrelated flows (missing session): $flow_session_not_found\n";
print STDERR "Correlated flows: $flow_correlated\n";
print STDERR "Uncorrelated SNMP clients: $snmp_clt_not_correlated\n";
print STDERR "Correlated SNMP clients: $snmp_clt_correlated\n\n";

my $i;
print STDERR "Correlated Sessions stats (certain):\n";
for ($i = 0; $i <= $#correlated_sessions_stats_certain; $i++) {
    print STDERR "  $i = $correlated_sessions_stats_certain[$i]\n";
}
print STDERR "Correlated Sessions stats (heuristic):\n";
for ($i = 0; $i <= $#correlated_sessions_stats_extra; $i++) {
    print STDERR "  $i = $correlated_sessions_stats_extra[$i]\n";
}
print STDERR "Correlated Sessions stats (all):\n";
for ($i = 0; $i <= $#correlated_sessions_stats_all; $i++) {
    print STDERR "  $i = $correlated_sessions_stats_all[$i]\n";
}

use IO::File;

my ($m, %cdf);
foreach $m (keys %num_sessions_per_clt) {
    &add_data_point($num_sessions_per_clt{$m}, \%cdf);
}

my $fname = "num_sessions_per_clt.cdf";
my $fh = new IO::File "> $fname";
die "cannot open $fname" if (not defined $fh);
build_freqbased_cdf($fh, \%cdf);
$fh->close;

# ---------------------------------------------------------------------

sub read_snmp_clt {
    # empty assoc set
    undef @session_set;
    if (eof SNMP_FILE) {
        $snmp_clt_ip = "";
        return;
    }

    # reuse old line if possible
    # 1113510964 172.29.148.71 91407143 0.5.78.72.175.4 5631 3 -80 1 6323 1088981 5683 4205332 74
546 0 152.23.64.10
    $snmp_line = <SNMP_FILE> if ($snmp_line eq "");

    # (MK) : out of the previous processing step we now only have session entries with single
client IP address
    # s 0.5.60.3.102.101 1113754247 1113754078 1113775547 1113775547 15 2 1 152.2.124.10
    my ($type, $mac, $first_ts, $start_ts, $last_update, $last_ts,
    $num_visits, $num_APs, $num_IPs, $IP, $apip) = split /\s/, $snmp_line;
    die "session lines expected\n" if ($type ne "s");
    # Here I change the original script so that it allows for the last field to be an IP address
(that of the
    # attachment AP). The check was made only to secure that there is only one IP address there but
this
```

```
    # should not be a problem given that in the previous processing step we filtered against
entries
    # with 2 or more client IPs

    # die "only one IP expected\n$snmp_line" if ($IP !~ /^\d+\.\d+\.\d+\.\d+$/);
    $snmp_clt_ip = $IP;
    $snmp_clt_mac = $mac;
    $ap_ip = $apip;
    print "SL: $snmp_line" if ($opts{"d"});

    # create or update assoc from polls of these clients

    while ($IP eq $snmp_clt_ip) {
        # (MK) : session is a hash storing the starting/end points of sessions and its global id
        my %session = ( "start" => "$start_ts", "end" => "$last_ts",
        "num" => $session_num++ );
        push @session_set, \%session;
        $num_sessions_per_clt{$snmp_clt_mac}++;

        &print_sessions if ($opts{"d"});

        return if (eof SNMP_FILE);
        $snmp_line = <SNMP_FILE>;
        ($type, $mac, $first_ts, $start_ts, $last_update, $last_ts,
        $num_visits, $num_APs, $num_IPs, $IP,$apip) = split /\s/, $snmp_line;

        print "SL: $snmp_line" if ($opts{"d"});
    }
}

sub print_session {
    my ($index) = @_;
    printf "SESSION: %d. %s %s\n", $session_set[$index]->{"num"},
    $session_set[$index]->{"start"}, $session_set[$index]->{"end"};
}

sub print_sessions {
    my $i;
    for ($i = 0; $i <= $#session_set; $i++) {
        print_session($i);
    }
}

sub process_clt_flows {

    # we come here with a snmp_clt_IP at hand together with the list of all his sessions
    # in the 7-8 days
    print ">> proc_flows\n" if ($opts{"d"});

    # 1113422694.133 30.085 152.23.64.10 1041 65.57.174.62 80 19 8862 Complete

    # (MK) : read from cvec file. Read all entries up to lines starting with CAP
    # apparently corresponding to a single connection
    &read_cvec if ($start_ts eq "");

    # find next relevant connection
    # (MK) : keep on reading from cvec file till the lan_ip address coincides with the snmp_clt_ip
    # read from the SNMP file before entering the process_clt_flows
    while (not eof STDIN and $snmp_clt_ip gt $lan_ip) {
        print "> skipped $snmp_clt_ip gt $lan_ip" if ($opts{"d"});
        &read_cvec;
        $flow_not_correlated++;
    }
```

```
    # (MK): either the snmp_clt_ip address will be equal to the IP address of the flow read in
read_cvec or
    # the latter will have exceeded it without finding a flow to correlate with this SNMP client
    if ($snmp_clt_ip eq $lan_ip) {

        $snmp_clt_correlated++;
        # client found - (MK) : nothing yet concluded for a certain session
        # enter the loop below to scan all flows with the same lan_ip, to see how many
        # of them can be correlated with one of the client sessions
        while ($snmp_clt_ip eq $lan_ip) {
            print "> correlated clt IP $snmp_clt_ip\n" if ($opts{"d"});

            $flow_correlated++;

            # find assocs that overlap with this flow
            my $i = 0;
            my %correlated_sessions;
            my $session_count = 0;
            my $session_count_extra = 0;

            # (MK) : if we have chosen conservative estimation of the connection, consider
            # as start and end time of the connection the time you saw the first (last packet).

            my $start = (exists $opts{"c"} ? $start_ts_lan : $start_ts);
            my $end = (exists $opts{"c"} ? $end_ts_lan : $end_ts);
            # (MK) : scan the sessions to see where fo I have flows within sessions
            # the rationale is anything  other than apparent : A flow should be correlated with a
session
            # only if start_flow > start_session and end_flow < end_session
            while ($i <= $#session_set) {
                if (# overlap on the start
                ($session_set[$i]->{"start"} <= $start and
                $session_set[$i]->{"end"} >= $start) or
                # overlap on the end
                ($session_set[$i]->{"start"} <= $end and
                $session_set[$i]->{"end"} >= $end) or
                # problem with duration for lan disconnected
                # or
                # association within flow lifetime
                ($opts{"i"} and
                $session_set[$i]->{"start"} >= $start and
                $session_set[$i]->{"end"} <= $end)) {
                    # (MK) : if the flow overlaps with one of the sessions, when session end are
defined as a
                    # POLLING INTERVAL-long larger, delete an entry that might exist in the second
definition
                    # and add one for the normal case
                    if (not exists $correlated_sessions{$session_set[$i]->{"num"}}) {
                        if (exists $correlated_sessions{"(" . $session_set[$i]->{"num"} . ")"}) {
                            delete $correlated_sessions{"(" . $session_set[$i]->{"num"} . ")"};
                            # (MK) : Shouldn't I reduce the sessions_count_extra counter here?
                        }
                        $correlated_sessions{$session_set[$i]->{"num"}}++;
                    }

                    $session_count++;

                } elsif (($session_set[$i]->{"start"} <= $start and
                $session_set[$i]->{"end"} + $poll_int >= $start) or
                ($session_set[$i]->{"start"} <= $end and
                $session_set[$i]->{"end"} + $poll_int >= $end)) {

                    $correlated_sessions{"(" . $session_set[$i]->{"num"} . ")"}++
```

```
                    if (not exists $correlated_sessions{$session_set[$i]->{"num"}});

                    $session_count_extra++;
                }
                $i++;
            }
            my @session_nums = sort { $correlated_sessions{$a} <=> $correlated_sessions{$b} }
            keys %correlated_sessions;

            if (@session_nums == 0) {
                if ($opts{"d"}) {
                    print "NOT FOUND: ";
                    &print_cvec;
                    print "\n";
                }
                $flow_session_not_found++;

            } else {
                &print_cvec;
                print "$session_count $session_count_extra ";

                my ($i, $certain, $extra);
                for ($i = 0; $i <= $#session_nums; $i++) {
                    print "$session_nums[$i]" . ($i == $#session_nums ? "" : " ");
                    if ($session_nums[$i] =~ /^\(/) {
                        $extra++;
                    } else {
                        $certain++;
                    }
                }
                print "\n";
                $correlated_sessions_stats_certain[$certain]++;
                $correlated_sessions_stats_extra[$extra]++;
                $correlated_sessions_stats_all[$certain + $extra]++;
            }

            last if (eof STDIN);
            &read_cvec;
        }

    } else {
        # snmp will read another client
        print "> cannot find $snmp_clt_ip\n" if ($opts{"d"});
        $snmp_clt_not_correlated++;
    }
}

sub read_cvec {
    $_ = <STDIN>;
    while (not /^CAP/) {

        if (/^CONC/ or /^SEQ/) {
            # SEQ 152.23.64.10.1034 > 128.109.34.38.80 t 456715003146 456805030131 3425 >SYN <SYN
>FIN/RST
                my @fields = split;

            ($lan_ip, $lan_port) = $fields[1] =~ /(\d+\.\d+\.\d+\.\d+)\.(\d+)/;
            ($wan_ip, $wan_port) = $fields[3] =~ /(\d+\.\d+\.\d+\.\d+)\.(\d+)/;
            $start_ts = $fields[5] / 1e6 + $start_trace;
            $end_ts = $fields[6] / 1e6 + $start_trace;
            die "negative duration\n$_" if ($end_ts - $start_ts < 0);
            if ($fields[$#fields] eq "<FIN/RST") {
                if ($fields[$#fields-1] eq ">FIN/RST") {
```

```
                    $term = "Complete";
                } else {
                    $term = "WAN_Closed";
                }
            } elsif ($fields[$#fields] eq ">FIN/RST") {
                $term = "LAN_Closed";
            } else {
                $term = "Not_Closed";
            }

        } elsif (/^LOSS/) {
            my @fields = split;
            $tot_pkt_lost = $fields[3] + $fields[4] + $fields[7] + $fields[8];

        } elsif (/^More-RTT/) {
            my @fields = split;
            # lan side
            $rtt_mad = $fields[11];
            $rtt_sd = $fields[9];

        } elsif (/^More-TS/) {
            my @fields = split;

            $start_ts_lan = $fields[2] / 1e6 + $start_trace;

            if ($fields[3] != -1) {
                $end_ts_lan = $fields[3] / 1e6 + $start_trace;
            } elsif ($fields[4] != -1) {
                $end_ts_lan = $fields[4] / 1e6 + $start_trace;
            } elsif ($fields[5] != -1) {
                $end_ts_lan = $fields[5] / 1e6 + $start_trace;
            } else {
            # (MK) : this adaptation is for avoiding some abnormal entries of More-TS to
            # cause termination of the program with die
                print DFILE "not lan end timestamp\n";
                print DFILE $_;
                $start_ts_lan = $start_ts;
                $end_ts_lan = $end_ts;
                #die "not lan end timestamp\n$_";
            }

            die "negative LAN duration\n$_" if ($end_ts_lan - $start_ts_lan < 0);

        } elsif (/^w/) {
        # w 17520 7504 > 9 0 767 399 < 9 0 8310 7942

            my @fields = split;

            $tot_pkts = $fields[4] + $fields[9];
            $tot_bytes = $fields[6] + $fields[11];
        }

        $_ = <STDIN>;
    }
}
sub print_cvec {

    printf "%.3f %.3f %.3f %.3f %s %s %s %s %s %s %s %s %s %s $term ",
    $start_ts, $end_ts, $start_ts_lan, $end_ts_lan,
    $snmp_clt_mac, $lan_ip, $lan_port, $wan_ip, $wan_port,
    $tot_pkts, $tot_bytes, $tot_pkt_lost, $rtt_mad, $rtt_sd;
}
```

create_Bid2BtypeMap.pl

```
#!/usr/local/bin/perl

my ($progname) = $0 =~ /([^\/]+)$/;

# ------------------------------------------------------------------------------
# USAGE
# ------------------------------------------------------------------------------
my $usage=<<EOF;
Map specific buildings to building types
EOF

# ------------------------------------------------------------------------------
# PRAGMAS
# ------------------------------------------------------------------------------

    use strict;

#-------------------------------------------------------------------------------
# OPTIONS
#-------------------------------------------------------------------------------

    use Getopt::Std;
    my (%opts);
    if (not getopts("dhrowfc", \%opts) or $opts{"h"}) {
        print "$usage";
    exit;
    }

# ------------------------------------------------------------------------------
# MAIN
# ------------------------------------------------------------------------------

    my %bldgid2bnum_hash;
    my %buildings_10June06_hash;
    my %btype2btype_num;
    my %results = ();
    get_Bnum();
    write_output_data();

# ------------------------------------------------------------------------------
# FUNCTIONS
# ------------------------------------------------------------------------------
    sub get_Bnum{
        my $Bldg_fname = 'bldgid2bnum.txt';
        open Bldg_LIST, "$Bldg_fname" || die "cannot open $Bldg_fname";
        while(<Bldg_LIST>){
            next if (/^\#/);
            chomp;
            my( $B_ID, $B_NUM ) = split;
            if(exists $bldgid2bnum_hash{$B_NUM}) {
                #already mapped
            }
            else{
                $bldgid2bnum_hash{$B_NUM} = $B_ID;
            }
        }
        close Bldg_LIST;
        my $Bldg_fname = 'buildings_10June06.txt';
        open Bldg_LIST, "$Bldg_fname" || die "cannot open $Bldg_fname";
        my $Btype_num = 1;
```

```
        while(<Bldg_LIST>){
            next if (/^\#/);
            chomp;
            my( $B_ID, $B_TYPE ) = split;
            if(exists $btype2btype_num{$B_TYPE}) {
                #already mapped
            }
            else{
                $btype2btype_num{$B_TYPE} = $Btype_num++;
            }

            if(exists $buildings_10June06_hash{$B_ID}) {
                #already mapped
            }
            else{
                $buildings_10June06_hash{$B_ID} = $B_TYPE;
            }
        }
        close Bldg_LIST;
    }
    sub write_output_data{
        #write output data to corresponding files
        open( OUT, ">Bnum2BtypeNum.txt");

        my ($cur_bid, $cur_num, $cur_btype, $cur_btype_num);
        foreach $cur_num( keys %bldgid2bnum_hash ){
            $cur_bid = $bldgid2bnum_hash{$cur_num};
            $cur_btype = $buildings_10June06_hash{$cur_bid};
            $cur_btype_num = $btype2btype_num{$cur_btype};
            #print OUT "$cur_num $cur_bid $cur_btype $cur_btype_num\n";
            print OUT "$cur_num $cur_btype_num\n";
        }
        close( OUT );
    }

    sub write_output_data2{
        #write output data to corresponding files
        open( OUT, ">Btype2Bnumtype.txt");

        my ($cur_bid, $cur_num, $cur_btype, $cur_btype_num);
        foreach $cur_btype( keys %btype2btype_num ){
            $cur_btype_num = $btype2btype_num{$cur_btype};
            print OUT "$cur_btype $cur_btype_num\n";
        }
        close( OUT );
    }
```

session-cvec-analysis-v2.pl

```
#!/usr/local/bin/perl

# ------------------------------------------------------------------------------
# USAGE
# ------------------------------------------------------------------------------

my ($progname) = $0 =~ /([^\/]+)$/;

my $usage=<<EOF;
Description:
    Correlating SNMP client data and cvec data.

Usage:
```

```
    $progname [options] snmp-session-file start-trace < cvec-file > flow-info

Input format:
    - snmp-session-file is the sorted output of clt_data2session.v4.pl, filtered
    for single client IP sessions with
    grep "^s" | grep -v "," | grep " 152\.[12]" | sort +9 -10 +3n -4 -s
    - cvec-file is the output of ascii-file processing with tcp2cvec

Options:
    -c  Conservative connection duration estimation.
    -d  Debugging mode.
    -h      Show this help message.
    -i       3rd rule (associations inside connections). Requires -c.
EOF

# ------------------------------------------------------------------------------
# PRAGMAS
# ------------------------------------------------------------------------------

use strict;

# ------------------------------------------------------------------------------
# IMPORTS
# ------------------------------------------------------------------------------

#die "Environment variable \$SRC undefined or incorrect"
#  if ($ENV{SRC} eq "");
#require "$ENV{SRC}/stats/cdf.pl";
require "./stats/cdf.pl";

# ------------------------------------------------------------------------------
# MAIN
# ------------------------------------------------------------------------------
#
# OPTIONS
#

use Getopt::Std;

my (%opts);
if (not getopts("cdhi", \%opts) or $opts{"h"} or @ARGV != 2) {
    print "$usage";
    exit;
}

my $snmp_fname = $ARGV[0];
die "cannot find $snmp_fname" if (not -e $ARGV[0]);
if ($ARGV[0] =~ /\.gz$/) {
    open SNMP_FILE, "zcat $ARGV[0] |";
} else {
    open SNMP_FILE, "$ARGV[0]";
}

open DFILE, ">out.debug-session-cvec";

my $start_trace = $ARGV[1];

my $poll_int = 300; # 5 minutes
my $delta_poll = 25; # conservative adjustment (due to req/rsp losses)

my $snmp_line;

my $snmp_clt_ip;
```

```
my $snmp_clt_mac;
my $ap_ip;

my @session_set; # list of hashes storing basic information about session (start/end point, id)
            # for a single client. The list is emptied (undefined) upon change of client
my $session_num = 1;

#(MK) : the extra allow a flow in the cvec file to finish at some point beyond the last
# SNMP polling interval that a session of the same client was seen, as long as this interval
# does not exceed the one POLLING INTERVAL
my @correlated_sessions_stats_certain;
my @correlated_sessions_stats_extra; # end of SNMP assoc + poll_int
my @correlated_sessions_stats_all; # end of SNMP assoc + poll_int

# (MK) variables logging stats
my ($flow_not_correlated, $flow_correlated, $flow_session_not_found,
    $snmp_clt_not_correlated, $snmp_clt_correlated);

# (MK) part of the output file format
my ($lan_ip, $lan_port, $wan_ip, $wan_port,
    $start_ts, $end_ts, $start_ts_lan, $end_ts_lan,
    $tot_pkts, $tot_bytes,
    $tot_pkt_lost, $rtt_mad, $rtt_sd, $term);

my %num_sessions_per_clt;

while (not eof STDIN) {
    &read_snmp_clt; # read all assoc info for one client (MK : store all his sessions in the list
of hashes session_set)
    &process_clt_flows; # process client if the next one in flow info
                # is same as above, skip or advance otherwise
}

print STDERR "STATS:\n\n";
print STDERR "Uncorrelated flows (missing clt): $flow_not_correlated\n";
print STDERR "Uncorrelated flows (missing session): $flow_session_not_found\n";
print STDERR "Correlated flows: $flow_correlated\n";
print STDERR "Uncorrelated SNMP clients: $snmp_clt_not_correlated\n";
print STDERR "Correlated SNMP clients: $snmp_clt_correlated\n\n";

my $i;
print STDERR "Correlated Sessions stats (certain):\n";
for ($i = 0; $i <= $#correlated_sessions_stats_certain; $i++) {
    print STDERR "  $i = $correlated_sessions_stats_certain[$i]\n";
}
print STDERR "Correlated Sessions stats (heuristic):\n";
for ($i = 0; $i <= $#correlated_sessions_stats_extra; $i++) {
    print STDERR "  $i = $correlated_sessions_stats_extra[$i]\n";
}
print STDERR "Correlated Sessions stats (all):\n";
for ($i = 0; $i <= $#correlated_sessions_stats_all; $i++) {
    print STDERR "  $i = $correlated_sessions_stats_all[$i]\n";
}

use IO::File;

my ($m, %cdf);
foreach $m (keys %num_sessions_per_clt) {
    &add_data_point($num_sessions_per_clt{$m}, \%cdf);
}

my $fname = "num_sessions_per_clt.cdf";
my $fh = new IO::File "> $fname";
```

```
die "cannot open $fname" if (not defined $fh);
build_freqbased_cdf($fh, \%cdf);
$fh->close;

# ------------------------------------------------------------------

sub read_snmp_clt {
    # empty assoc set
    undef @session_set;
    if (eof SNMP_FILE) {
        $snmp_clt_ip = "";
        return;
    }

    # reuse old line if possible
    # 1113510964 172.29.148.71 91407143 0.5.78.72.175.4 5631 3 -80 1 6323 1088981 5683 4205332 74
546 0 152.23.64.10
    $snmp_line = <SNMP_FILE> if ($snmp_line eq "");

    # (MK) : out of the previous processing step we now only have session entries with single
client IP address
    # s 0.5.60.3.102.101 1113754247 1113754078 1113775547 1113775547 15 2 1 152.2.124.10
    my ($type, $mac, $first_ts, $start_ts, $last_update, $last_ts,
    $num_visits, $num_APs, $num_IPs, $IP, $apip) = split /\s/, $snmp_line;
    die "session lines expected\n" if ($type ne "s");
    # Here I change the original script so that it allows for the last field to be an IP address
(that of the
    # attachment AP). The check was made only to secure that there is only one IP address there but
this
    # should not be a problem given that in the previous processing step we filtered against
entries
    # with 2 or more client IPs

    # die "only one IP expected\n$snmp_line" if ($IP !~ /^\d+\.\d+\.\d+\.\d+$/);
    $snmp_clt_ip = $IP;
    $snmp_clt_mac = $mac;
    $ap_ip = $apip;
    print "SL: $snmp_line" if ($opts{"d"});

    # create or update assoc from polls of these clients

    while ($IP eq $snmp_clt_ip) {
        # (MK) : session is a hash storing the starting/end points of sessions and its global id
        my %session = ( "start" => "$start_ts", "end" => "$last_ts",
        "num" => $session_num++ );
        push @session_set, \%session;
        $num_sessions_per_clt{$snmp_clt_mac}++;

        &print_sessions if ($opts{"d"});

        return if (eof SNMP_FILE);
        $snmp_line = <SNMP_FILE>;
        ($type, $mac, $first_ts, $start_ts, $last_update, $last_ts,
        $num_visits, $num_APs, $num_IPs, $IP,$apip) = split /\s/, $snmp_line;

        print "SL: $snmp_line" if ($opts{"d"});
    }
}

sub print_session {
    my ($index) = @_;
    printf "SESSION: %d. %s %s\n", $session_set[$index]->{"num"},
    $session_set[$index]->{"start"}, $session_set[$index]->{"end"};
```

```
}

sub print_sessions {
    my $i;
    for ($i = 0; $i <= $#session_set; $i++) {
        print_session($i);
    }
}

sub process_clt_flows {

    # we come here with a snmp_clt_IP at hand together with the list of all his sessions
    # in the 7-8 days
    print ">> proc_flows\n" if ($opts{"d"});

    # 1113422694.133 30.085 152.23.64.10 1041 65.57.174.62 80 19 8862 Complete

    # (MK) : read from cvec file. Read all entries up to lines starting with CAP
    # apparently corresponding to a single connection
    &read_cvec if ($start_ts eq "");

    # find next relevant connection
    # (MK) : keep on reading from cvec file till the lan_ip address coincides with the snmp_clt_ip
    # read from the SNMP file before entering the process_clt_flows
    while (not eof STDIN and $snmp_clt_ip gt $lan_ip) {
        print "> skipped $snmp_clt_ip gt $lan_ip" if ($opts{"d"});
        &read_cvec;
        $flow_not_correlated++;
    }
    # (MK): either the snmp_clt_ip address will be equal to the IP address of the flow read in
read_cvec or
    # the latter will have exceeded it without finding a flow to correlate with this SNMP client
    if ($snmp_clt_ip eq $lan_ip) {

        $snmp_clt_correlated++;
        # client found - (MK) : nothing yet concluded for a certain session
        # enter the loop below to scan all flows with the same lan_ip, to see how many
        # of them can be correlated with one of the client sessions
        while ($snmp_clt_ip eq $lan_ip) {
            print "> correlated clt IP $snmp_clt_ip\n" if ($opts{"d"});

            $flow_correlated++;

            # find assocs that overlap with this flow
            my $i = 0;
            my %correlated_sessions;
            my $session_count = 0;
            my $session_count_extra = 0;

            # (MK) : if we have chosen conservative estimation of the connection, consider
            # as start and end time of the connection the time you saw the first (last packet).

            my $start = (exists $opts{"c"} ? $start_ts_lan : $start_ts);
            my $end = (exists $opts{"c"} ? $end_ts_lan : $end_ts);
            # (MK) : scan the sessions to see where fo I have flows within sessions
            # the rationale is anything  other than apparent : A flow should be correlated with a
session
            # only if start_flow > start_session and end_flow < end_session
            while ($i <= $#session_set) {
                if (# overlap on the start
                ($session_set[$i]->{"start"} <= $start and
                $session_set[$i]->{"end"} >= $start) or
                # overlap on the end
```

```
                ($session_set[$i]->{"start"} <= $end and
                $session_set[$i]->{"end"} >= $end) or
                # problem with duration for lan disconnected
                # or
                # association within flow lifetime
                ($opts{"i"} and
                $session_set[$i]->{"start"} >= $start and
                $session_set[$i]->{"end"} <= $end)) {
                    # (MK) : if the flow overlaps with one of the sessions, when session end are
defined as a
                    # POLLING INTERVAL-long larger, delete an entry that might exist in the second
definition
                    # and add one for the normal case
                    if (not exists $correlated_sessions{$session_set[$i]->{"num"}}) {
                        if (exists $correlated_sessions{"(" . $session_set[$i]->{"num"} . ")"}) {
                            delete $correlated_sessions{"(" . $session_set[$i]->{"num"} . ")"};
                            # (MK) : Shouldn't I reduce the sessions_count_extra counter here?
                        }
                        $correlated_sessions{$session_set[$i]->{"num"}}++;
                    }

                    $session_count++;

                } elsif (($session_set[$i]->{"start"} <= $start and
                $session_set[$i]->{"end"} + $poll_int >= $start) or
                ($session_set[$i]->{"start"} <= $end and
                $session_set[$i]->{"end"} + $poll_int >= $end)) {

                    $correlated_sessions{"(" . $session_set[$i]->{"num"} . ")"}++
                    if (not exists $correlated_sessions{$session_set[$i]->{"num"}});

                    $session_count_extra++;
                }
                $i++;
            }
            my @session_nums = sort { $correlated_sessions{$a} <=> $correlated_sessions{$b} }
            keys %correlated_sessions;

            if (@session_nums == 0) {
                if ($opts{"d"}) {
                    print "NOT FOUND: ";
                    &print_cvec;
                    print "\n";
                }
                $flow_session_not_found++;

            } else {
                &print_cvec;
                print "$session_count $session_count_extra ";

                my ($i, $certain, $extra);
                for ($i = 0; $i <= $#session_nums; $i++) {
                    print "$session_nums[$i]" . ($i == $#session_nums ? "" : " ");
                    if ($session_nums[$i] =~ /^\(/) {
                        $extra++;
                    } else {
                        $certain++;
                    }
                }
                print "\n";
                $correlated_sessions_stats_certain[$certain]++;
                $correlated_sessions_stats_extra[$extra]++;
                $correlated_sessions_stats_all[$certain + $extra]++;
```

```
            }

            last if (eof STDIN);
            &read_cvec;
        }

    } else {
        # snmp will read another client
        print "> cannot find $snmp_clt_ip\n" if ($opts{"d"});
        $snmp_clt_not_correlated++;
    }
}

sub read_cvec {

    $_ = <STDIN>;
    while (not /^CAP/) {

        if (/^CONC/ or /^SEQ/) {
            # SEQ 152.23.64.10.1034 > 128.109.34.38.80 t 456715003146 456805030131 3425 >SYN <SYN
>FIN/RST
                my @fields = split;

            ($lan_ip, $lan_port) = $fields[1] =~ /(\d+\.\d+\.\d+\.\d+)\.(\d+)/;
            ($wan_ip, $wan_port) = $fields[3] =~ /(\d+\.\d+\.\d+\.\d+)\.(\d+)/;
            $start_ts = $fields[5] / 1e6 + $start_trace;
            $end_ts = $fields[6] / 1e6 + $start_trace;
            die "negative duration\n$_" if ($end_ts - $start_ts < 0);
            if ($fields[$#fields] eq "<FIN/RST") {
                if ($fields[$#fields-1] eq ">FIN/RST") {
                    $term = "Complete";
                } else {
                    $term = "WAN_Closed";
                }
            } elsif ($fields[$#fields] eq ">FIN/RST") {
                $term = "LAN_Closed";
            } else {
                $term = "Not_Closed";
            }

        } elsif (/^LOSS/) {
            my @fields = split;
            $tot_pkt_lost = $fields[3] + $fields[4] + $fields[7] + $fields[8];

        } elsif (/^More-RTT/) {
            my @fields = split;
            # lan side
            $rtt_mad = $fields[11];
            $rtt_sd = $fields[9];

        } elsif (/^More-TS/) {
            my @fields = split;

            $start_ts_lan = $fields[2] / 1e6 + $start_trace;

            if ($fields[3] != -1) {
                $end_ts_lan = $fields[3] / 1e6 + $start_trace;
            } elsif ($fields[4] != -1) {
                $end_ts_lan = $fields[4] / 1e6 + $start_trace;
            } elsif ($fields[5] != -1) {
                $end_ts_lan = $fields[5] / 1e6 + $start_trace;
            } else {
            # (MK) : this adaptation is for avoiding some abnormal entries of More-TS to
```

```
            # cause termination of the program with die
                print DFILE "not lan end timestamp\n";
                print DFILE $_;
                $start_ts_lan = $start_ts;
                $end_ts_lan = $end_ts;
                #die "not lan end timestamp\n$_";
            }

            die "negative LAN duration\n$_" if ($end_ts_lan - $start_ts_lan < 0);

        } elsif (/^w/) {
        # w 17520 7504 > 9 0 767 399 < 9 0 8310 7942

            my @fields = split;

            $tot_pkts = $fields[4] + $fields[9];
            $tot_bytes = $fields[6] + $fields[11];
        }

        $_ = <STDIN>;
    }
}

sub print_cvec {

    printf "%.3f %.3f %.3f %.3f %s %s %s %s %s %s %s %s %s %s $term ",
    $start_ts, $end_ts, $start_ts_lan, $end_ts_lan,
    $snmp_clt_mac, $lan_ip, $lan_port, $wan_ip, $wan_port,
    $tot_pkts, $tot_bytes, $tot_pkt_lost, $rtt_mad, $rtt_sd;
}
```

SeperateTrace2CLFiles.pl

```
#!/usr/local/bin/perl

# ------------------------------------------------------------------------------
# USAGE
# ------------------------------------------------------------------------------

my ($progname) = $0 =~ /([^\/]+)$/;

my $usage=<<EOF;
Description:
    Split original data to separate files for each client
EOF

# ------------------------------------------------------------------------------
# PRAGMAS
# ------------------------------------------------------------------------------

    use strict;

# ------------------------------------------------------------------------------
# MAIN
# ------------------------------------------------------------------------------

#Number of Input Files - Trace File Days

    my $input_files_no = 10;
    my $file_num;
    my $filename;
```

```
    for $file_num( 1 ... $input_files_no ) {

        $filename = 'd'.$file_num.'.dat';
        print 'Trying to open :'."$filename"."\n";
        open ( DATA, "$filename" );

        while (<DATA>) {
            next if (/^\s*$/); # skip empty lines

            # Parse client polling line

            my ($poll_time, $ap_num, $ap_up_time, $clt_num, $clt_up_time,
            $clt_state, $dbm, $power_save,
            $pkts_recv, $bytes_recv, $pkts_sent, $bytes_sent,
            $dups, $retrans_msdu, $undelivered_msdu, $br) = split;

            die "cannot parse line\n[$_]" if ($poll_time eq "");

            my $CLfilename;
            $CLfilename = 'cl'.$clt_num;

            if( -e $CLfilename ){
                open( OUT, ">>$CLfilename") || die "cannot create $CLfilename: $!";
            }
            else{
                open( OUT, ">$CLfilename") || die "cannot create $CLfilename: $!";
            }
            print OUT $_;
            close( OUT ) || die "cannot close $CLfilename: $!";
        }
        close( DATA );
    }
```

sortClientFiles.pl

```
#!/usr/local/bin/perl

# ------------------------------------------------------------------------------
# USAGE
# ------------------------------------------------------------------------------

my ($progname) = $0 =~ /([^\/]+)$/;

my $usage=<<EOF;
Description:
    Helper function used to sort client files - sort first by AP and then by timestamp
EOF


# ------------------------------------------------------------------------------
# PRAGMAS
# ------------------------------------------------------------------------------

    use strict;

# ------------------------------------------------------------------------------
# MAIN
# ------------------------------------------------------------------------------

    my %hash = ();
    my $client;
```

```
    my $fileName;
    my $totalClients = 15000;
    my $ap;
    my $timestamp;

    for $client( 1 ... $totalClients )
    {
        $fileName = 'cl' . $client;

        if( -e $fileName )
        {
            print 'Trying to open: '."$fileName"."\n";
            open(DATA, "$fileName")|| die "cannot open source file";

            %hash = ();

            while (<DATA>)
            {
                next if (/^\s*$/); # skip empty lines

                my ($poll_time, $ap_num, $ap_up_time, $clt_num, $clt_up_time,
                $clt_state, $dbm, $power_save,
                $pkts_recv, $bytes_recv, $pkts_sent, $bytes_sent,
                $dups, $retrans_msdu, $undelivered_msdu) = split;

                die "cannot parse line\n[$_]" if ($poll_time eq "");

                $hash{$ap_num}{$poll_time} = $_;
            }

            close(DATA);

            $fileName = 'sortedClient' . $client . '.txt';

            open(outCFILE, ">$fileName") || die "cannot open target file";

            foreach $ap (sort by_number keys(%hash))
            {
                foreach $timestamp (sort by_number keys(%{$hash{$ap}}))
                {
                    print outCFILE "$hash{$ap}{$timestamp}";
                }
            }

            close(outCFILE);
        }
    }

sub by_number
{
    $a <=> $b;
}
```

StatisticsPerClient.pl

```
#!/usr/bin/perl -w

#-------------------------------------------------------------------------------#
# DESCPRIPTION :
#   This script is used to calculate some statistics about each client in the tcp flow trace.
```

```
# INPUT
#   Flow data split to client files.
# OUTPUT
#   The columns will contain the following elements :
#   col0 : client id
#   col1 : mean flow size
#   col2 : median flow size
#   col3 : std
#   col4 : 95% confidence interval lower bound for flow size
#   col5 : 95% confidence interval upper bound for flow size
#   col6 : mean duration
#   col7 : median duration
#   col8 : std of the duration
#   col9 : 95% confidence interval lower bound for flow duration
#   col10 : 95% confidence interval upper bound for flow duration
#-------------------------------------------------------------------------------#

# ------------------------------------------------------------------------------
# USAGE
# ------------------------------------------------------------------------------

my ($progname) = $0 =~ /([^\/]+)$/;
my $usage=<<EOF;
StatisticsPerClient.pl mode
 -mode 0 : perform analysis for all clients
 -mode 1 : perform analysis for clients that have home AP
  HomeAPwith75Thres.txt has the corresponding mapping from clientID to homeAP
EOF

# ------------------------------------------------------------------------------
# PRAGMAS
# ------------------------------------------------------------------------------

use strict;

# ------------------------------------------------------------------------------
# OPTIONS
# ------------------------------------------------------------------------------

use Getopt::Std;

my (%opts);
if (not getopts("dh", \%opts) or $opts{"h"}) {
  print "$usage";
  exit;
}

# ------------------------------------------------------------------------------
# MAIN
# ------------------------------------------------------------------------------

my $functiontype = $ARGV[0]; # 0 all aps - 1 aps that have home AP
my $numArgs = $#ARGV + 1;
if ($numArgs < 1 or (($functiontype != 0)and($functiontype != 1))){
    print "$usage";
    exit 0;
}
my $max_client_id = 20000;
my ($i,$j);
my %homeAP;
my ($input_filename, $output_filename);
my (@array, @sample);
my $counter_in_array = 0; # number of entries in array
```

```
#-------------------------------------------------------------------------------#
# OUTPUT Format (array)
# col0 : client id
# col1 : mean flow size
# col2 : median flow size
# col3 : std flow size
# col4 : 95% confidence interval lower bound for flow size
# col5 : 95% confidence interval upper bound for flow size
# col6 : mean duration
# col7 : median duration
# col8 : std of the duration
# col9 : 95% confidence interval lower bound for flow duration
# col10 : 95% confidence interval upper bound for flow duration
#-------------------------------------------------------------------------------#
getHomeAP();
for($i=0; $i<$max_client_id; ++$i){
    $input_filename = "cl"."$i";
    my @temp_duration_vector; my @temp_traffic_vector; my @temp_100pkt_traffic_vector;
    my $temp_duration_counter = 0; my $temp_traffic_counter = 0; my $temp_100pkt_traffic_counter =
0;
    my @temp_100pkt_duration_vector, my $temp_100pkt_duration_counter = 0;
    my @temp_100pkt_fromHome_traffic_vector, my $temp_100pkt_fromHome_traffic_counter = 0;
    my @temp_100pkt_fromHome_duration_vector, my $temp_100pkt_fromHome_duration_counter = 0;
    if( -e $input_filename  ){
        open( INPUT, "$input_filename");
        print 'Opening Client File :'."$input_filename"."\n";
        while( <INPUT> ) {
            @sample = split;
            #$poll_time, $wireless_cl_num, $ap_id, $tot_pkts, $tot_bytes, $duration, $duration_lan,
$tcp_term, $port_num_lan, $port_num_wan
            $temp_duration_vector[$temp_duration_counter++] = $sample[5];
            $temp_traffic_vector[$temp_traffic_counter++] = $sample[4];
            if( $sample[3] >= 100 ){
                $temp_100pkt_traffic_vector[$temp_100pkt_traffic_counter++] = $sample[4];
                $temp_100pkt_duration_vector[$temp_100pkt_duration_counter++] = $sample[5];
            }
            if( ($functiontype == 1)&&( exists $homeAP{ $sample[1] } )){
                if( ($homeAP{ $sample[1] } == $sample[2]) && ( $sample[3] >= 100 )){
                    $temp_100pkt_fromHome_traffic_vector[$temp_100pkt_fromHome_traffic_counter++] =
$sample[4];
                    $temp_100pkt_fromHome_duration_vector[$temp_100pkt_fromHome_duration_counter++]
= $sample[5];
                }
            }
        }
        $array[$counter_in_array][0] = $sample[1];
        $array[$counter_in_array][1] = mean( @temp_traffic_vector);
        $array[$counter_in_array][2] = median( @temp_traffic_vector);
        $array[$counter_in_array][3] = my_std( @temp_traffic_vector);
        $array[$counter_in_array][4] = mySum( @temp_traffic_vector);
        $array[$counter_in_array][5] = lower_conf_interval( @temp_traffic_vector);
        $array[$counter_in_array][6] = upper_conf_interval( @temp_traffic_vector);
        $array[$counter_in_array][7] = mean( @temp_duration_vector);
        $array[$counter_in_array][8] = median( @temp_duration_vector);
        $array[$counter_in_array][9] = my_std( @temp_duration_vector);
        $array[$counter_in_array][10] = lower_conf_interval( @temp_duration_vector);
        $array[$counter_in_array][11] = upper_conf_interval( @temp_duration_vector);
        $array[$counter_in_array][12] = $temp_traffic_counter;

        if( $temp_100pkt_traffic_counter > 0 ){
            $array[$counter_in_array][13] = mean( @temp_100pkt_traffic_vector);
            $array[$counter_in_array][14] = median( @temp_100pkt_traffic_vector);
```

```
            $array[$counter_in_array][15] = my_std( @temp_100pkt_traffic_vector);
            $array[$counter_in_array][16] = mySum( @temp_100pkt_traffic_vector);
            $array[$counter_in_array][17] = mean( @temp_100pkt_duration_vector);
            $array[$counter_in_array][18] = median( @temp_100pkt_duration_vector);
            $array[$counter_in_array][19] = my_std( @temp_100pkt_duration_vector);
            $array[$counter_in_array][20] = $temp_100pkt_traffic_counter;
        }
        else{
            $array[$counter_in_array][13] = -1;
            $array[$counter_in_array][14] = -1;
            $array[$counter_in_array][15] = -1;
            $array[$counter_in_array][16] = -1;
            $array[$counter_in_array][17] = -1;
            $array[$counter_in_array][18] = -1;
            $array[$counter_in_array][19] = -1;
            $array[$counter_in_array][20] = -1;
        }
        if( $temp_100pkt_fromHome_traffic_counter > 0 ){
            $array[$counter_in_array][21] = mean( @temp_100pkt_fromHome_traffic_vector);
            $array[$counter_in_array][22] = median( @temp_100pkt_fromHome_traffic_vector);
            $array[$counter_in_array][23] = my_std( @temp_100pkt_fromHome_traffic_vector);
            $array[$counter_in_array][24] = mean( @temp_100pkt_fromHome_duration_vector);
            $array[$counter_in_array][25] = median( @temp_100pkt_fromHome_duration_vector);
            $array[$counter_in_array][26] = my_std( @temp_100pkt_fromHome_duration_vector);
            $array[$counter_in_array][27] = $temp_100pkt_fromHome_traffic_counter;
        }
        else{
            $array[$counter_in_array][21] = -1;
            $array[$counter_in_array][22] = -1;
            $array[$counter_in_array][23] = -1;
            $array[$counter_in_array][24] = -1;
            $array[$counter_in_array][25] = -1;
            $array[$counter_in_array][26] = -1;
            $array[$counter_in_array][27] = -1;
        }
        $counter_in_array++;
        close( INPUT );
    }
    else {
        # do nothing
    }
} # end of for loop counting clients

$output_filename = "perClientStatistcs2.dat";
open( OUTPUT, ">$output_filename");
    for($i=0; $i<$counter_in_array; $i++){
        for( $j=0; $j<27; ++$j ) {
            print OUTPUT "$array[$i][$j]." ";
        }
        print OUTPUT "$array[$i][$j]."\n";
    }
close( OUTPUT );

# ------------------------------------------------------------------------------
# Functions
# ------------------------------------------------------------------------------

sub getHomeAP{
    open( HOMEAP, "HomeAPwith75Thres.txt");
    while (<HOMEAP>){
        next if (/^\s*$/); # skip empty lines
        # Parse client polling line
        my ( $client, $ap ) = split;
```

```
        $homeAP{ $client } = $ap;
    }
    close( HOMEAP );
}
sub checkValidRange{
    my $cur_poll = $_[ 0 ];
    #From [Wed Apr 13 19:05:58 2005] to [Thu Apr 21 05:18:16 2005]
    if( ($cur_poll > 1113408358 )&&($cur_poll < 1114049896) ){
        return 1;
    }
    else{
        return 0;
    }
}

# ------------------------------------------------------------------------
# Stats Functions
# ------------------------------------------------------------------------
sub mySum {
  my ($sum, $elem);
  $sum = 0;
  foreach $elem (@_) {
    $sum += $elem;
  }
  return($sum);
}
sub mean {
        my(@data)=@_;
        my $sum;
        foreach(@data) {
                $sum+=$_;
        }
        return($sum/@data);
}
sub median {
        my(@data)=sort { $a <=> $b} @_;
        if (scalar(@data)%2) {
                return($data[@data/2]);
        } else {
                my($upper, $lower);
                $lower=$data[@data/2];
                $upper=$data[@data/2 - 1];
                return(mean($lower, $upper));
        }
}
sub std_dev {
        my(@data)=@_;
        my($sq_dev_sum, $avg)=(0,0);

        $avg=mean(@data);
        foreach my $elem (@data) {
                $sq_dev_sum+=($avg-$elem)**2;
        }
        return(sqrt($sq_dev_sum/@data-1));
}
sub my_std {
    my(@data) = @_;
    my $avg = mean(@data);
    my $sum = 0;
    my $counter = 0;

    foreach my $elem ( @data) {
        $sum += ($elem-$avg)**2;
```

```
        $counter++;
    }
    --$counter;
    if( $counter != 0 ) {
        $sum = $sum / $counter;
        $sum = sqrt($sum);
    }
    else {
        # if one vector contains only one element then the std of that vector is the value of that
one element.
        $sum = $data[0];
    }
    return $sum;
}
sub lower_conf_interval {
    my (@lower_conf_interval_temp) = @_;
    my $mean_val = mean(@lower_conf_interval_temp);
    my $std_val = my_std(@lower_conf_interval_temp);
    my $constant = 1.96;
    # for 95% confidence interval the constant in the equation for the confidence interval must ne
1.96
    my $array_len = scalar(@lower_conf_interval_temp);

    return $mean_val - ( ($constant*$std_val)/sqrt($array_len) ) ;
}
sub upper_conf_interval {
    my (@upper_conf_interval_temp) = @_;
    my $mean_val = mean(@upper_conf_interval_temp);
    my $std_val = my_std(@upper_conf_interval_temp);
    my $constant = 1.96;
    # for 95% confidence interval the constant in the equation for the confidence interval must ne
1.96
    my $array_len = scalar(@upper_conf_interval_temp);

    return $mean_val + ( ($constant*$std_val)/sqrt($array_len) ) ;
}
```

getTopAps.m

```
%-------------------------------------------------------------------------------%
% DESCPRIPTION :
%   Function used to locate hotspots among all APs
% INPUT
%    Traffic: matrix containing the traffic (SNMP) data for all Aps
%             rows correspond to Aps and columns to time slots
%    holes: vector containing invalid APIDs
% OUTPUT
%   top: vector containing hotspots
%   topTraffic: matrix containing traffic time series for the corresponding hotspots
%-------------------------------------------------------------------------------%
function [top,topTraffic] = getTopAps( Traffic, holes )
    %input daily traffic

        [ r, c ] = size( Traffic );

        numTopDAY = 65;
        numTopHOUR= 65;

        %%%% GET TOP HOUR APS
        for i = 1:numTopDAY
            topHOUR( i, 1 ) = 0;    %ap
            topHOUR( i, 2 ) = 0;    %top hourly traffic for that AP
```

```
        end

        for j = 1:c
            curAP = j;
            if( find( holes(:,1) == curAP ) )
                continue;
            end
            for i = 1:r
                curTraffic = Traffic( i, j );

                if( curTraffic > topHOUR( 1, 2 ))
                    foundTop = find( topHOUR( :, 1 ) == curAP );
                    if( foundTop )
                        topHour( foundTop, 2 ) = curTraffic;
                        topHOUR = sortrows( topHOUR, [2] );
                    else
                        topHOUR( 1, 1 ) = curAP;
                        topHOUR( 1, 2 ) = curTraffic;
                        topHOUR = sortrows( topHOUR, [2] );
                    end
                end
            end
        end
        topHOUR;
        %%%% GET TOP DAY APS

        for j = 1:c
            curAP = j;
            startRow = 1;
            endRow = 24;
            curAPTopDay = 0;
            if( find( holes(:,1) == curAP ) )
                continue;
            end
            while( startRow < r - 23 )
                curDAY = Traffic( startRow:endRow, j );

                curDayTraffic = sum( curDAY );
                if( curDayTraffic > curAPTopDay )
                    curAPTopDay = curDayTraffic;
                end

                startRow = startRow + 24;
                endRow = endRow + 24;
            end
            APTopDay( curAP, 1 ) = curAP;
            APTopDay( curAP, 2 ) = curAPTopDay;
        end

        APTopDay = sortrows( APTopDay, [2] );

        for i = 1:numTopDAY
            APTopDaySel( i, 1 ) = APTopDay( c + 1 - i, 1 );
            APTopDaySel( i, 2 ) = APTopDay( c + 1 - i, 2 );
        end
        APTopDaySel;

        %%%% Final Selection
        sizeTop = 1;
        top( 1, 1 ) = 0;
        top( 1, 2 ) = 0;
        for i = 1:numTopDAY
            curAP = APTopDaySel( i, 1 );
```

```
            found = find( topHOUR(:,1) == curAP );
            if( found )
                top( sizeTop, 1 ) = APTopDaySel( i, 1 );
                top( sizeTop, 2 ) = APTopDaySel( i, 2 );
                top( sizeTop, 3 ) = topHOUR( found, 2);
                sizeTop = sizeTop + 1;
            end
        end

        top = sortrows( top, [2] );

        [ r, c ] = size(top);

        for i = 1:r
            curHotspot = top( i, 1 );
            topTraffic( :, i ) = Traffic( :, curHotspot );
        end
```

movingAverage.m

```
%-------------------------------------------------------------------------------%
% DESCPRIPTION :
%   Function used to locate hotspots among all APs
% INPUT
%    Traffic: matrix containing the traffic (SNMP) data for all Aps
%             rows correspond to Aps and columns to time slots
%    window: window for which the moving average is calculated
% OUTPUT
%   smooth: smoothed values produced
%-------------------------------------------------------------------------------%

function smooth = movingAverage2( Traffic, window )

    [ r, c ] = size( Traffic );

    for j = 1:c
        index = 1;
        for i = 1:r
            if( window == 3 )
                if( i == 1 )
                    smooth( i, j ) = (Traffic( i, j ) + Traffic( i+1, j ) + Traffic( i+2, j ))/3;
                elseif( i == r )
                    smooth( i, j ) = Traffic( i-2, j ) + Traffic( i-1, j ) + Traffic( i, j )/3;
                else
                    smooth( i, j ) = Traffic( i-1, j ) + Traffic( i, j ) + Traffic( i+1, j ) /3;
                end
            elseif( window == 5 )
                if( i == 1 )
                    smooth( i, j ) = (Traffic( i, j ) + Traffic( i+1, j ) + Traffic( i+2, j ))/3;
                elseif( i == 2 )
                    smooth( i, j ) = (Traffic( i-1, j ) + Traffic( i, j ) + Traffic( i+1, j ) +
Traffic( i+2, j ) )/4;
                elseif( i == r )
                    smooth( i, j ) = Traffic( i-2, j ) + Traffic( i-1, j ) + Traffic( i, j )/3;
                elseif( i == r - 1 )
                    smooth( i, j ) = Traffic( i-2, j ) + Traffic( i-1, j ) + Traffic( i, j ) +
Traffic( i+1, j )/4;
                else
                    smooth( i, j ) = Traffic( i-2, j ) + Traffic( i-1, j ) + Traffic( i, j ) +
Traffic( i+1, j ) + Traffic( i+2, j ) /5;
                end
```

```
            else
                smooth( 1, 1 ) = -1;
            end
        end
    end
```

predictAP2ratioRegressW4.m

```
%-------------------------------------------------------------------------%
% DESCPRIPTION :
%   P3 Algorithm
% INPUT
%    Traffic: matrix containing the traffic (SNMP) data for all Aps
%             rows correspond to Aps and columns to time slots
%    Visits: matrix containing the visits (SNMP) data for all Aps
%             rows correspond to Aps and columns to time slots
%    week_num: weeks selected to apply the algorithm
%    holes: vector containing invalid APIDs
%    window: window used in the moving average component of the algorithm
%    mode: if mode equals 'all' then the algorithm is applied over all hotspots
%    hotspots: vector containing hotspots in the corresponding dataset
%    multi_step: boolean value - if 1 then the algorithm is multi steo otherwise
%                single step method is used
% OUTPUT
%   P3_FINAL: [ Apid; meanP3val; medianP3val; stdP3val ]
%-------------------------------------------------------------------------%

    %added visits in the formula....
    function [P3_FINAL,debugT] = predictAP2ratioRegressW4( Traffic, Visits, week_num, holes,
window, mode, hotspots, multi_step )
    %function [P3,debug] = predictAP2ratioRegressW2( Traffic, week_num, holes, window, mode,
rWeights, multi_step )

        [ r, c ] = size( Traffic );
        APid = hotspots(:,1);

        week_start = ((week_num-1)*7+5)*24 + 1;
        week_end = ((week_num-1)*7+5+5)*24;
        week = Traffic( week_start:week_end, : );
        weekV = Visits( week_start:week_end, : );

        tWeights = getRegressionWeights4( Traffic, week, holes, window, hotspots );
        vWeights = getRegressionWeights4( Visits, weekV, holes, window, hotspots );
        logWeights = getRegressionWeights4b( Traffic, Visits, tWeights, vWeights, week, weekV,
holes, window, hotspots );

        for j = 1:c
            curAP = j;
            [APMeanTot,APMeanTotPerHour] = ratioHelper( Traffic, r, curAP, week_num, multi_step );
            [APMeanTotV,APMeanTotPerHourV] = ratioHelper( Visits, r, curAP, week_num, multi_step );
            %V1
            for k = 1:24
                V1( k, j ) = mean( APMeanTot( :, k ) );
                VV1( k, j ) = mean( APMeanTotV( :, k ) );
            end
            %V2
            hour2 = 1;
            day2 = 1;
            for n = 1:120
                hmean = mean( APMeanTotPerHour( :, n ) );
                hmeanV = mean( APMeanTotPerHourV( :, n ) );
```

```
            col2 = (curAP-1)*5 + day2;
            V2( hour2, col2 ) = hmean;
            VV2( hour2, col2 ) = hmeanV;
            if( hour2 == 24 )
                hour2 = 0;
                day2 = day2 + 1;
            end
            hour2 = hour2 + 1;
        end
    end

    %APMeanTot size = (5*5)x24 [ 5*5 indicates 5days(weekdays) for 5 weeks ]
    %V1 size = 24*488 [ each row contains the mean traffic for that specific hour ]

    [ r, c ] = size( week );
    holesFoundSize = 1;
    sizeP3 = 1;
    debugSIZE = 1;
    debugTsize = 1;
    for j = 1:c
        curAP = j;
        hour3 = 1;
        day3 = 1;
        posP3 = 0;
        tot = 0;

        for h = 1:window
            history( h, 1 ) = 0;
            historyV( h, 1 ) = 0;
        end
        if( find( holes(:,1) == curAP ) )
            %do nothing
        else
            for i = 1:r
                curTraffic = week( i, j );
                curVisits = weekV( i, j );

                meanHistory = mean( history( :, 1 ) );
                meanHistoryV = mean( historyV( :, 1 ) );

                found = find( tWeights( :, 1 ) == curAP );
                foundV = find( vWeights( :, 1 ) == curAP );
                foundL = find( logWeights( :, 1 ) == curAP );

                if( found )
                    tweight1 = tWeights( found, 2 );
                    tweight2 = tWeights( found, 3 );
                    tweight3 = tWeights( found, 4 );
                else
                    tweight1 = 0;
                    tweight2 = 0;
                    tweight3 = 0;;
                end
                if( foundV )
                    vweight1 = vWeights( foundV, 2 );
                    vweight2 = vWeights( foundV, 3 );
                    vweight3 = vWeights( foundV, 4 );
                else
                    vweight1 = 0;
                    vweight2 = 0;
                    vweight3 = 0;;
                end
                if( foundL )
```

```
                        lweight1 = logWeights( foundL, 2 );
                        lweight2 = logWeights( foundL, 3 );
                    else
                        lweight1 = 0;
                        lweight2 = 0;
                    end

                    avgT = tweight2*V1( hour3, curAP ) + tweight3*V2( hour3, (curAP-1)*5 + day3 ) +
tweight1*meanHistory;
                    %%%%%%%%%
                    debugT( debugTsize, 1 ) = avgT;
                    debugTsize = debugTsize + 1;
                    %%%%%%%%%
                    avgV = vweight2*VV1( hour3, curAP ) + vweight3*VV2( hour3, (curAP-1)*5 + day3 )
+ vweight1*meanHistoryV;
                    avg = lweight1*log( avgT ) + lweight2*log( avgV );
                    lweight1
                    lweight2

                    if( curTraffic ~= 0 )
                        if( mode == 'all')
                            found = 1;  %ALL APS
                        else
                            found = find( APid( :, 1 ) == curAP );
                        end
                        if( found )
                            value = (abs(avg - log(curTraffic)))/log(curTraffic);

                            P3( sizeP3, 1  ) = j;
                            P3( sizeP3, 2  ) = value;
                            sizeP3 = sizeP3 + 1;
                        end
                    end

                    if( hour3 == 24 )
                        hour3 = 0;
                        day3 = day3 + 1;
                    end
                    hour3 = hour3 + 1;

                    %update history
                   if( window >= 2 && window <= 6 )
                        history( 1, 1 ) = history( 2:window, 1 );
                        history( window, 1 ) = curTraffic;
                        historyV( 1, 1 ) = historyV( 2:window, 1 );
                        historyV( window, 1 ) = curVisits;
                    else
                        disp( 'wrong WINDOW SIZE');
                    end
                end
            end
        end

    debug = unique(P3(:,1));
    [ r, c ] = size( P3 );

    prevAP = P3( 1, 1 );
    tabSize = 1;
    P3size = 1;
    for i = 1:r
        curAP = P3( i, 1 );

        if( curAP ~= prevAP )
```

```
            P3_FINAL( P3size, 1 ) = prevAP;
            P3_FINAL( P3size, 2 ) = mean( ratioTab(:,1) );
            P3_FINAL( P3size, 3 ) = median( ratioTab(:,1) );
            P3_FINAL( P3size, 4 ) = std( ratioTab(:,1) );

            clear ratioTab;
            P3size = P3size + 1;
            tabSize = 1;
        end
        ratioTab( tabSize, 1 ) = P3( i, 2 );
        tabSize = tabSize + 1;

        prevAP = curAP;
    end
    P3_FINAL( P3size, 1 ) = prevAP;
    P3_FINAL( P3size, 2 ) = mean( ratioTab(:,1) );
    P3_FINAL( P3size, 3 ) = median( ratioTab(:,1) );
    P3_FINAL( P3size, 4 ) = std( ratioTab(:,1) );


    stats(1,1) = mean( P3(:,2));
    stats(2,1) = median( P3(:,2));
    stats(3,1) = std( P3(:,2));
```

NAMSA2RemOutliers.m

```
%-------------------------------------------------------------------------------%
% DESCPRIPTION :
%   NAMSA Algorithm
% INPUT
%    traffic: matrix containing the traffic (SNMP) data for all Aps
%             rows correspond to Aps and columns to time slots
%    visits: matrix containing the visits (SNMP) data for all Aps
%             rows correspond to Aps and columns to time slots
%    week_num: weeks selected to apply the algorithm
%    mode: if mode equals 'all' then the algorithm is applied over all hotspots
% OUTPUT
%   namsa_final: [ Apid; meanNAMSAval; medianNAMSAval; stdNAMSAval ]
%   residuals: vector containing the residual values
%-------------------------------------------------------------------------------%
    function [namsa_final,residuals] = NAMSA2RemOutliers( traffic, visits, week_num, mode, APid )

        if( week_num ~= 0 )
            week_start = ((week_num-1)*7+5)*24 + 1;
            week_end = ((week_num-1)*7+5+5)*24;
            traffic = traffic( week_start:week_end, : );
            visits = visits( week_start:week_end, : );
        end

        [ rt, ct ] = size( traffic );
        [ rv, cv ] = size( visits );

    %   trafficRemOutliers = matrixRemoveOutliers( traffic, 1.96 );
    %   visitsRemOutliers = matrixRemoveOutliers( visits, 1.96 );

        normTraffic = traffic.^(1/4);
        normVisits = visits.^(1/4);
    %   normTrafficRemOutliers = trafficRemOutliers.^(1/4);
    %   normVisitsRemOutliers = visitsRemOutliers.^(1/4);

        normTrafficRemOutliers = matrixRemoveOutliers( normTraffic, 1.96 );
```

```
        normVisitsRemOutliers = matrixRemoveOutliers( normVisits, 1.96 );


        namsa_resSize = 1;
        if( rt ~= rv || ct ~= cv )
            disp 'Invalid Input'
        else
            for j = 1:ct
                %get regression weights
                y = normTrafficRemOutliers( 2:rt, j );
                X(:,1) = ones( rt-1, 1 );
                X(:,2) = normTrafficRemOutliers( 1:rt-1, j );
                X(:,3) = normVisitsRemOutliers( 1:rt-1, j );
                rweights = regress( y, X );
                rw_a = rweights( 1, 1 );
                rw_b = rweights( 2, 1 );
                rw_c = rweights( 3, 1 );

                residualsSize = 1;
                curAP = j;
                for i = 2:rt
                    curTraffic = traffic( i, j );
                    curVisits = visits( i, j );
                    prevTraffic = normTrafficRemOutliers( i-1, j );
                    prevVisits = normVisitsRemOutliers( i-1, j );

                    avg = rw_a + rw_b*prevTraffic + rw_c*prevVisits;

                    avgR = avg^4;
                    res_val = curTraffic - avgR;
                    residuals( i-1, curAP ) = res_val;

                    if( curTraffic ~= 0 )
                        if( mode == 'all')
                            found = 1;  %ALL APS
                        else
                            found = find( APid( :, 1 ) == curAP );
                        end
                        if( found )
                            avgf = avg^4;
                            value = (abs(avgf - curTraffic))/curTraffic;

                            namsa_res( namsa_resSize, 1  ) = j;
                            namsa_res( namsa_resSize, 2  ) = value;
                            namsa_resSize = namsa_resSize + 1;
                        end
                    end
                end
            end
        end

    [ r, c ] = size( namsa_res );
    prevAP = namsa_res( 1, 1 );
    tabSize = 1;
    namsaSize = 1;
    for i = 1:r
        curAP = namsa_res( i, 1 );

        if( curAP ~= prevAP )
            namsa_final( namsaSize, 1 ) = prevAP;
            namsa_final( namsaSize, 2 ) = mean( ratioTab(:,1) );
            namsa_final( namsaSize, 3 ) = median( ratioTab(:,1) );
            namsa_final( namsaSize, 4 ) = std( ratioTab(:,1) );
```

```
            clear ratioTab;
            namsaSize = namsaSize + 1;
            tabSize = 1;
        end
        ratioTab( tabSize, 1 ) = namsa_res( i, 2 );
        tabSize = tabSize + 1;

        prevAP = curAP;
    end
    namsa_final( namsaSize, 1 ) = prevAP;
    namsa_final( namsaSize, 2 ) = mean( ratioTab(:,1) );
    namsa_final( namsaSize, 3 ) = median( ratioTab(:,1) );
    namsa_final( namsaSize, 4 ) = std( ratioTab(:,1) );


    stats(1,1) = mean( namsa_res(:,2));
    stats(2,1) = median( namsa_res(:,2));
    stats(3,1) = std( namsa_res(:,2));
```

spectrum_analysis.m

```
%-------------------------------------------------------------------------------%
% DESCPRIPTION :
%   Perform a spectrum analysis in a given vector dataset
% INPUT
%    HTraffic: matrix containing the traffic (SNMP) data for all Clients
%              rows correspond to Clients and columns to time slots
%    Client: client for which the spectrum analysis is performed
%    parCorrOn _num: boolean value - if '1' then partial correlation analysis
%                    is performed
% OUTPUT
%   cycle: The resulting dominant period
%-------------------------------------------------------------------------------%

    function cycle = spectrum_analysis( HTraffic, Client, parCorrOn )

        %HTraffic format 7751x3610
        %Rows correspond to specific clients - columns contain ts values
        %Column 1 contains client ID - columns 2:361 contain ts values

        [r,c] = size( HTraffic );
        cl_index = find( HTraffic( :, 1 ) == Client );
        cl_index
        if( cl_index )
            x = HTraffic( cl_index:cl_index, 2:c );

            Y = fft(x);
            N = length(Y);
            Y(1) = [];
            power = abs(Y(1:N/2)).^2;
            nyquist = 1/2;
            freq = (1:N/2)/(N/2)*nyquist;
            period = 1./freq;

            [mp,index] = max(power);
            cycle = period(index);

            if( parCorrOn == 1 )
                subplot(2,1,1)
            end
            plot(period,power), grid on
```

```
            s = sprintf('Spectrum Analysis Client[%d] - Cycle=%d', Client,cycle)
            title( s )
            ylabel('Power')
            xlabel('Period')

            if( parCorrOn == 1 )
                subplot(2,1,2)
                parcorr( x, 24 )
                s = sprintf('Partial Autocorrelation Function Client[%d]', Client)
                title( s )
            end
        else
            disp 'Invalid client ID'
        end
```

chiSquareTest.m

```
%-------------------------------------------------------------------------------%
% DESCPRIPTION :
%   Perform a ChiSquare Test
% INPUT
%    Input : Input data are expected in columns 3 and 4 respectively and column
%            five should be filled with zero vals if the corresponding row is valid
%            is performed
%-------------------------------------------------------------------------------%

function [chi2 critical df] = chiSquareTest( a );

% read individual columns of interest from a - in paraller filter elements with
% "invalid" values
w1 = a(:,3); len1 = length(w1);
dist1 = a(:,4);
valid1 = a(:,5);
ww1 = find(valid1 < 0);
w1(ww1) = NaN.*w1(ww1); w1 = w1(~isnan(w1));
dist1(ww1) = NaN.*dist1(ww1);dist1 = dist1(~isnan(dist1));

% carry out the chi-square independence test for the contingency table
% need to determine bins -try to make them equal size
nbinsx = 10;
nbinsy = 10;
alpha = 0.01;

% 1st dataset
w1s = sort(w1);
indx = round([1/nbinsx:1/nbinsx:1]*length(w1s));
binsx = w1s(indx);
% check whether some of the binsx values are the same and adapt bins
binsxx(1) = binsx(1);
ind2=1;
for ind1=2:length(binsx)
    if binsx(ind1) > binsx(ind1-1)
        ind2=ind2+1;
        binsxx(ind2) = binsx(ind1);
    end
end
%binsxx

dist1s = sort(dist1);
indy = round([1/nbinsy:1/nbinsy:1]*length(dist1s));
binsy = dist1s(indy);
```

```
binsyy(1) = binsy(1);
ind2=1;
for ind1=2:length(binsy)
    if binsy(ind1) > binsy(ind1-1)
        ind2=ind2+1;
        binsyy(ind2) = binsy(ind1);
    end
end
%binsyy

[chi2 critical df]=chi2D_contingency(w1,dist1,binsxx,binsyy,alpha);
fprintf('Chi-square test for the contingency table of edge weights and distances\n');
fprintf('Chi-square statistic T = %f critical_value at alpha=%f c=%f for df=%d
\n\n',chi2,alpha,critical,df);

clear binsx binsxx indx binsy binsyy indy ind1 ind2;
```

predictExponential.m

```
%-------------------------------------------------------------------------------%
% DESCPRIPTION :
%   Predict traffic using the correlation relationship between traffic and visits
%   in the log scale
% INPUT
%    Visits: matrix containing the visits data for all APs
%            rows correspond to APs and columns to time slots
%    Traffic: matrix containing the traffic (SNMP) data for all APs
%            rows correspond to APs and columns to time slots
%    predict_start, predict_end : week numbers corresponding to the time period
%                                 used as history window
%    hotspots: vector containing the hotspot Aps
% OUTPUT
%   cycle: The resulting dominant period
%-------------------------------------------------------------------------------%


function [ExpRatioRes,absDiff] = predictExponential( Flows, Traffic, predict_start, predict_end,
hotspots )

        %Get Prediction Period
        TrafficPredict = Traffic( predict_start:predict_end-1,:);
        FlowsHistory = Flows( predict_start:predict_end-1,:);

        [ r, c ] = size( Flows );
        [ rt, ct ] = size( TrafficPredict );

        bweight = flows2traffic( Flows, Traffic, predict_start, hotspots, 2, 1 )
        sizeP3 = 1; absDiffSize = 1;
        for curAP = 1:ct
            isHotspot = find( hotspots(:,1 ) == curAP );
            if( isHotspot )
                for curRow = 2:rt
                    curTraffic = TrafficPredict( curRow, curAP );
                    prevFlows = FlowsHistory( curRow-1, curAP );
                    %T = e^b*V^a
                    weightIndex = find( bweight(:,1 ) == curAP );
                    alpha = bweight( weightIndex, 2 );
                    beta = bweight( weightIndex, 3 );
                    predTraffic = exp( beta )*(prevFlows.^alpha);

                    if( curTraffic ~= 0 )
                        value = (abs(predTraffic - curTraffic))/curTraffic;

                        P3( sizeP3, 1  ) = curAP;
```

```
                        P3( sizeP3, 2  ) = value;
                        sizeP3 = sizeP3 + 1;
                    end
                    absDiff( absDiffSize, 1 ) = (abs(predTraffic - curTraffic));
                    absDiff( absDiffSize, 2 ) = predTraffic;
                    absDiff( absDiffSize, 3 ) = curTraffic;
                    absDiffSize = absDiffSize + 1;
                end
            end
        end
            [ r, c ] = size( P3 );

    prevAP = P3( 1, 1 );
    tabSize = 1;
    P3size = 1;
    for i = 1:r
        curAP = P3( i, 1 );

        if( curAP ~= prevAP )
            ExpRatioRes( P3size, 1 ) = prevAP;
            ExpRatioRes( P3size, 2 ) = mean( ratioTab(:,1) );
            ExpRatioRes( P3size, 3 ) = median( ratioTab(:,1) );
            ExpRatioRes( P3size, 4 ) = std( ratioTab(:,1) );

            clear ratioTab;
            P3size = P3size + 1;
            tabSize = 1;
        end
        ratioTab( tabSize, 1 ) = P3( i, 2 );
        tabSize = tabSize + 1;

        prevAP = curAP;
    end
    ExpRatioRes( P3size, 1 ) = prevAP;
    ExpRatioRes( P3size, 2 ) = mean( ratioTab(:,1) );
    ExpRatioRes( P3size, 3 ) = median( ratioTab(:,1) );
    ExpRatioRes( P3size, 4 ) = std( ratioTab(:,1) );
```

overlapping_intervals.m

```
%-------------------------------------------------------------------------------%
% DESCPRIPTION :
%   Check whether data vectors v1 and v2 have overlapping confidence intervals
% INPUT
%    v1: data vector 1
%    v2: data vector 2
% OUTPUT
%   overlap: value is 1 if v1 and v2 have overlapping confidence intervals. 0 otherwise
%-------------------------------------------------------------------------------%

    function [overlap,trend] = overlapping_intervals( v1, v2 )

        l1 = length( v1 ); l2 = length( v2 ); overlap = 0;
        [confV1L,confV1R] = confidenceInterval( v1 );
        [confV2L,confV2R] = confidenceInterval( v2 );

        if( ((mean( v1 ) > mean( v2 )) && (confV2R > confV1L)) || ( (mean( v2 ) > mean( v1 )) &&
(confV1R > confV2L)))
            overlap = 1;
        end
        if( (overlap == 0) && (mean( v1 ) > mean( v2 ) ))
            trend = 2; %ascending
```

```
        else
            trend = 1; %descending
        end
```

confidenceInterval.m

```
%-------------------------------------------------------------------------------%
% DESCPRIPTION :
%   Calculate confidence intervals for given data vector
% INPUT
%    vector: input data
% OUTPUT
%   confL - confR: left and right confidence intervals respectively
%-------------------------------------------------------------------------------%

    function [confL,confR] = confidenceInterval( vector )

        X = mean( vector );
        n = length( vector );
        SS = 0; c = 1.96;
        for i = 1:n
            SS = SS + (vector( i ) - X).^2;
        end
        SS = (1/(n-1))*SS;
        S = sqrt( SS );
        confL = X - c*S/sqrt( n );
        confR = X + c*S/sqrt( n );
```

levelShiftConfInt.m

```
%-------------------------------------------------------------------------------%
% DESCPRIPTION :
%   Function levelShiftConfInt
%   last modification 4/10/05
%       -> introducing a method to locate level shifts based on confidence intervals
% INPUT
%    traffic_vec : Traffic matrix column corresponding to the AP analyzed
%       index : index in the traffic_vec matrix containing the value tested
%       meanTrafficVal : threshold used to set the minimum diffence for the method metric for
%       two possible consecutive levels
% OUTPUT
%   CurLevel : level corresponding to the traffic_vec[index] value tested
%-------------------------------------------------------------------------------%

function [CurLevel, level_shift_index_asc, level_shift_index_desc ] = levelShiftConfInt( traffic_vec,
index )
        trafficLen = length( traffic_vec );
        possible_shift_min_size = 0;
        L2 = [];
        if( trafficLen < 5 )
            disp 'Cannot Initiate Level Detection - Input Vector is too small'
        elseif( index <= 5 )
            L1 = traffic_vec( 1:index );
            L2_start = index + 1;
        else
            %Initialize L1
            L1_start = 1; L1_end = 3;
            maxL1Size = 12;
            %Initialize L2
            L2_start = 4; L2_end = 4;
            minL2Size = 3;
            %Initialize flags
```

```
level_shift_asc_size = 1; level_shift_desc_size = 1;
trigger = 0; overlap = 1;
foundLargerVal = 0; foundSmallerVal = 0;;

while( (L2_end < index)&&(L2_end < trafficLen))
    if( trigger == 0 && possible_shift_min_size == 0 )
        %ADVANCE L1
        if( (L1_end - L1_start) >= maxL1Size )  %maxSize reached
            L1_start = L1_start + 1;
        end
        L1_end = L1_end + 1;
        clear L1;
        L1 = traffic_vec( L1_start:L1_end, 1 );
        %ADVANCE L2
        L2_start = L1_end + 1;
        L2_end = L2_start;
        clear L2;
        L2 = traffic_vec( L2_start:L2_end, 1 );

        if( find( L1( :, 1 ) >= L2( 1, 1 ) ) )
            foundLargerVal = 1;
        end
        if( find( L1( :, 1 ) <= L2( 1, 1 ) ) )
            foundSmallerVal = 1;
        end
        if( foundLargerVal == 0 || foundSmallerVal == 0 )
            trigger = 1;
        end
        foundLargerVal = 0; foundSmallerVal = 0;
    else
        %ADVANCE L2
        L2_end = L2_end + 1;
        if( L2_end < trafficLen )   %reached the end of traffic_vec
            clear L2;
            L2 = traffic_vec( L2_start:L2_end, 1 );
        else
            break;
        end
        if( L2_end - L2_start < minL2Size )
            possible_shift_min_size = 1;
        else
            possible_shift_min_size = 0;
            [overlap,trend] = overlapping_intervals( L1, L2 );
            if( overlap == 0 )
                %New Level Found
                L1_start = L2_start;
                L1_end = L2_end;
                L2_start = L1_end + 1;
                L2_end = L2_start;
                L1 = traffic_vec( L1_start:L1_end, 1 );
                L2 = traffic_vec( L2_start:L2_end, 1 );
                if( trend == 1 )
                    level_shift_index_asc( level_shift_asc_size, 1 ) = L1_start;
                    level_shift_asc_size = level_shift_asc_size + 1;
                elseif( trend == 2 )
                    level_shift_index_desc( level_shift_desc_size, 1 ) = L1_start;
                    level_shift_desc_size = level_shift_desc_size + 1;
                else
                    disp 'Invalid trend found'
                end
            else
                %disp 'Overlapping'
            end
```

```
                        trigger = 0;
                    end
                end
            end
        end
        if( L2_start <= index )
            %Return L1-L2 Level [aggregate]
            s2 = length( L2 );
            s1 = length( L1 );
            CurLevel( 1:s1, 1 ) = L1;
            CurLevel( (s1+1):(s1+s2), 1 ) = L2;
        else
            %Return CurLevel [L1]
            CurLevel = L1;
        end
```

ccdf.m

```
%-------------------------------------------------------------------------------%
% DESCPRIPTION :
%   Plot CCDF based on empirical CDF
% INPUT
%    inputvec: input data vector
% OUTPUT
%   vec : CDF values computed
%-------------------------------------------------------------------------------%

function [vec,xvals] = ccdf( inputvec )
    [vec,xvals] = ecdf( inputvec );
    plot(xvals, 1-vec, 'r.')
```