UNDERSTANDING A LARGE-SCALE IPTV NETWORK VIA SYSTEM LOGS

A Thesis
Presented to
The Academic Faculty

by

Tongqing Qiu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology
August 2011

UNDERSTANDING A LARGE-SCALE IPTV NETWORK VIA SYSTEM LOGS

Approved by:

Prof. Jun (Jim) Xu, Adviser
School of Computer Science
Georgia Institute of Technology

Prof. Mostafa H. Ammar
School of Computer Science
Georgia Institute of Technology

Prof. Nick Feamster
School of Computer Science
Georgia Institute of Technology

Prof. Xiaoli Ma
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Jia Wang
Network Measurement and Engineering Research Department
AT&T-Lab Research

Date Approved: August 2011

To Ivy.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my adviser Jun (Jim) Xu, who encouraged me to take up Ph.D. study in Georgia Tech, and provided kind guidance and consistent support throughout my entire study.

I would like to thank Jia Wang, together with Zihui Ge, Dan Pei, who directed and helped my research on fork and join networks during my internship at AT&T research, and during the years afterwards to see it through. I want to thank Nan Hua who practically helped me starting from my first paper. I would like to thank all my co-authors, whose participation are invaluable for the many projects I was involved in, and from whom I have learned a lot. I would like to thank my thesis committee, who showed interest in my work and offered valuable suggestions.

My special thanks to my loving wife, Ivy, who steadfastly supported my study and shouldered majority of the burden of the family.

TABLE OF CONTENTS

DEDICATION . . . . . **iii**

ACKNOWLEDGEMENTS . . . . . **iv**

LIST OF TABLES . . . . . **viii**

LIST OF FIGURES . . . . . **ix**

SUMMARY . . . . . **xii**

I INTRODUCTION **1**

1.1 Our Contributions 3

1.1.1 Router Syslog Mining 3

1.1.2 Set-top Box Logs Study 3

1.1.3 Remarks 4

1.2 Related Studies Overview 5

1.2.1 IPTV Measurement Study 5

1.2.2 System Log Study 5

1.3 Organization of Dissertation 6

II BACKGROUND **8**

2.1 Overview of IPTV Architecture 8

2.2 Router Syslog 10

2.3 STB logs 13

III ROUTER SYSLOG MINING **14**

3.1 Introduction 14

3.2 System Design Overview 17

3.2.1 Offline Domain Knowledge Learning 19

3.2.2 Online SyslogDigest System 22

3.3 Syslog Mining Methodologies 24

3.3.1 Offline Learning Methodologies 24

3.3.2 Online System Methodologies 33
3.4 Evaluation 36
3.4.1 Evaluation Methodology 37
3.4.2 Components Effectiveness 38
3.4.3 Performance of SyslogDigest 43
3.5 Applications 46
3.5.1 Complex network troubleshooting 46
3.5.2 Network health monitoring and visualization 48
3.6 Related Work 50
3.7 Summary and Future Work 51
IV MODELING IPTV CHANNEL POPULARITY 52
4.1 Introduction 52
4.2 Measurement Results of Channel Popularity 54
4.2.1 Distribution of Channel Popularity 56
4.2.2 Correlation between Channel Accesses and Channel Dwell Time 57
4.2.3 Temporal Dynamics of Channel Popularity 58
4.2.4 Multi-scale Property of Channel Popularity Similarity 61
4.3 Modeling Channel Popularity 63
4.3.1 Zipf-like Model 63
4.3.2 Mean Reversion Model 63
4.3.3 Forecasting Channel Popularity 65
4.4 Multi-class Popularity Modeling 66
4.4.1 Grouping STBs 67
4.4.2 Measuring Difference in Channel Preferences of STB Groups 71
4.4.3 Identifying Best Grouping Methods 73
4.4.4 Explaining Channel Popularity Dynamics 75
4.4.5 Modeling and Simulation 78
4.5 Related Work 79

4.6 Summary and Future Work 80

V MODELING IPTV USER BEHAVIOR 82

5.1 Introduction 82

5.2 Analyzing User Activities 85

5.2.1 Turning STBs On and Off 85

5.2.2 Switching Channels 88

5.2.3 Channel Popularity 89

5.3 Modeling User Activities 92

5.3.1 Modeling Session Length 92

5.3.2 Modeling Time-Varying Rates 95

5.3.3 Modeling Channel Popularity Distribution 97

5.3.4 Modeling Channel Popularity Dynamics 98

5.4 SimulWatch: A Workload Generator 100

5.4.1 SimulWatch Design 100

5.4.2 Evaluation 102

5.5 Related Work 110

5.6 Summary and Future Work 111

VI CONCLUSION 113

REFERENCES 115

VITA 121

LIST OF TABLES

1 Syslog messages example 10

2 Syslog messages example (cont.) 10

3 Toy Example. Router r1's interface Serial1/0.10/10:0 is connected to r2's inteface Serial1/0.20/20:0. 18

4 The messages belong to the same message type (BGP-5-ADJCHANGE) 26

5 Sub message types of BGP-5-ADJCHANGE 26

6 Sensitivity of minimal support SP_{min} value 38

7 Parameter setting in SyslogDigest 43

8 Effectiveness (compression ratio) of three digest methodologies. T: temporal based, R: rule based, C: cross router 44

9 Classification of top 150 IPTV channels 71

10 Channel preferences of STB groups based on PREF. 71

11 Symmetric uncertainty between PREF and different grouping methods 74

12 Stability of different grouping methods 74

13 Channel preferences of STB groups based on WT-D. 75

14 Model parameters for session length distributions 93

15 Modeling parameters for event rates 95

16 Goodness-of-fit scores for session length and channel popularity distributions 105

17 RMSE when modeling the time-varying rate 105

LIST OF FIGURES

1 IPTV Architecture 9

2 SyslogDigest Architecture. 20

3 Sub type tree construction example. 25

4 Location hierarchy 29

5 Controller up/down example. 31

6 TCP bad authentication example 31

7 The impact of parameter SP_{min} and $Conf_{min}$, in dataset A. 40

8 The impact of parameter W, when $Conf_{min} = 0.8$ and $SP_{min} = 0.0005$. 40

9 The number of rules over 12 weeks, dataset A. 41

10 The number of rules over 12 weeks, dataset B. 41

11 The impact of varying value of α on the compression ratio ($\beta = 2$). . 42

12 The impact of varying value of β on the compression ratio ($\alpha = 0.05$ for dataset A and $\alpha = 0.075$ for dataset B). 43

13 Number of event digests and active rules per day for dataset A. . . . 44

14 The digest result per router of dataset A. 45

15 Visualization based on SyslogDigest output. 48

16 Visualization based on raw syslog data. 49

17 The number of online STBs for each hour during a week. 55

18 The number of channel switches for each hour during a week. 55

19 CDF of channel popularity. 55

20 Channel popularity distribution (varying time period). 56

21 Channel popularity distribution (varying start time). 56

22 The correlation between channel access frequency and dwell time. . . 58

23 The dynamics of channel popularity of kids channel K, 1 point every 15 minutes 59

24 The dynamics of channel popularity of news channel N, 1 point every 15 minutes 60

25 The distribution of CoV 60

26 The distribution of the slope in ACF 61
27 The average cosine similarity for different aggregation time scales . . 62
28 Channel popularity distribution. 63
29 Cosine similarity using a simulated trace based on the mean reversion process. 65
30 Dissimilarity vs. K when we use TV watching time as grouping feature. 69
31 Time-of-day dynamics for news channels, comparing WT-D with all watchers 76
32 Time-of-day dynamics for kids channels, comparing WT-D with all watchers 76
33 Population mix for each group based on WT-D 77
34 The cosine similarity function when varying lag. The solid line represents the real trace, the dash line represents the model. Top: single-class, bottom: multi-class. 78
35 CCDF of the length for on-, off-, and channel-sessions 86
36 Number of on-line STBs 87
37 The normalized switching-on, switching-off, and channel switching events (one-minute granularity) 87
38 CDF of channels popularity 90
39 Ratio of change in popular channels, as seen over hours in a single day. 91
40 Channel popularity distribution change (hourly) 91
41 QQ plots comparing models and real traces 92
42 The time-varying rates in frequency domain 94
43 Modeling aggregate event rate 94
44 Find the optimal number of spikes 96
45 Fitting the channel popularity distribution 97
46 ON-OFF model 100
47 Comparison of the session-length distribution. CCDFs for the real trace and generated workload closely match in all cases. 103
48 Comparison of the aggregate event rate. The real-trace results are on the top, and the workload results are on the bottom. 103

49 Channel popularity distributions for the real trace and the generated workload. 106

50 Number of on-line STBs over time. The results from real trace and workload closely match. 106

51 Multi-class population model captures the change of channel popularity over time (hourly) 107

52 Time-of-day dynamics for a popular kids channel, based on multi-class synthetic trace 108

53 Population mix for each group, based on multi-class synthetic trace . 109

54 Case study. The results from real trace and workload closely match. 109

SUMMARY

Recently, there has been a global trend among the telecommunication industry on the rapid deployment of IPTV (Internet Protocol Television) infrastructure and services. While the industry rushes into the IPTV era, the comprehensive understanding of the status and dynamics of IPTV network lags behind. Filling this gap requires in-depth analysis of large amounts of measurement data across the IPTV network. One type of the data of particular interest is device or system log, which has not been systematically studied before. In this dissertation, we will explore the possibility of utilizing system logs to serve a wide range of IPTV network management purposes including health monitoring, troubleshooting and performance evaluation, etc. In particular, we develop a tool to convert raw router syslogs to meaningful network events. In addition, by analyzing set-top box (STB) logs, we propose a series of models to capture both channel popularity and dynamics, and users' activity on the IPTV network.

CHAPTER I

INTRODUCTION

In the past several years, there has been a global trend among telecommunication companies on the rapid deployment of IPTV (Internet Protocol Television) infrastructure and services, in which live TV streams are encoded as a series of IP packets and delivered to users through the residential broadband access network and set-top boxes (STB). These IPTV services include commercial grade multicasting TV, video on demand (VoD), triple play, voice over IP (VoIP), and Web/email access, well beyond traditional cable television services [51].

Large-scale IPTV networks are designed with the goal of providing high availability and Quality of Service (QoS) while keeping the operational complexity and cost low. Meeting this goal requires the detailed knowledge of network status and dynamics. While the industry rushes into the IPTV era, what lags behind is a comprehensive understanding of the IPTV networks. We believe that the understanding includes at least two parts: (1) to understand the network itself (e.g., network monitoring and troubleshooting); (2) to understand the end-users' activity (e.g., channel switching) and impact (e.g., bandwidth consumption).

To understand both aspects in a large-scale network (not limited to IPTV) in general, an Internet Service Provider (ISP) usually collects a large amount of measurement data across the network, including device log files, traps and alarms on different network layers, active or passive service performance measures, etc. In this thesis, we focus primarily on one data source – *logs*, which are emitted by network devices, operating systems, applications and various other types of intelligent or programmable devices. Although these logs are usually created by the software developers to aid in

the debugging of the operation of these applications, they can be used by the network service providers for various management tasks.

In an IPTV network, there are two important types of system logs: (1) router syslogs that a router passively generates to describe a wide range of events observed by it. (2) STB logs that describe the end-users' activities like turning on or off the STB. These two logs, however, have not been effectively used from network operators' perspective. For instance, detailed router syslog messages are typically examined manually or through inflexible tools only when required by an on-going troubleshooting investigation or when given a narrow time range and a specific router under suspicion. Similarly, STB logs are used only by Cha et al. before to take the initial user behaviors study without comprehensive modeling [12].

Extracting useful information from raw logs is far from a trivial task. The first challenge lies in the complexity and variety of *each* log. The syntax and semantics of data within log messages (e.g., router syslogs) are usually application or vendor-specific. Log message format or content may not always be fully documented. The second challenge is the subtle patterns buried in *multiple* logs. The pattern can be the implicit association between any two logs, or the underlying model that all logs follow.

The goal of this dissertation is to explore the possibility of utilizing previously underutilized logs in the IPTV network by discovering the embedded patterns to serve network management tasks such as health monitoring, troubleshooting, and performance evaluation. In particular, we analyze router syslogs and STB logs to unveil IPTV network status and dynamics from both network and end-user's perspectives. We explain our contributions in details as follows.

1.1 Our Contributions

1.1.1 Router Syslog Mining

IPTV router syslog messages are essentially free-form text with only a minimal structure, and their formats vary among different vendors and router OSes. Furthermore, since router syslogs are aimed for tracking and debugging router software/hardware problems, they are often too low-level from network service management perspectives. Due to their sheer volume, detailed router syslog messages are typically examined only when required by an on-going troubleshooting investigation or when given a narrow time range and a specific router under suspicion. We design a SyslogDigest system that can automatically transform and compress such low-level minimally-structured syslog messages into meaningful and prioritized high-level network events, using powerful machine learning techniques tailored to our problem domain. These events are three orders of magnitude fewer in number and have much better usability than raw syslog messages. We demonstrate that they provide critical input to IPTV network troubleshooting, and health monitoring and visualization.

1.1.2 Set-top Box Logs Study

Different from router syslogs, STB logs are well formatted to record all users activities with respect to turning on and off STBs, and switching channels. The challenging part is therefore no longer the variety of logs syntax but the subtle model underlying the logs. We first model one primary aspect of IPTV system – channel popularity and then model all user activities.

1.1.2.1 Channel Popularity Modeling

Understanding the channel popularity or content popularity is an important step in the workload characterization for modern information distribution systems such as World Wide Web, P2P file- sharing systems, IPTV networks, video-on-demand (VOD) systems, content distribution networks, publish/subscribe systems, and RSS

feeds distribution systems. We focus on analyzing the channel popularity in the context of IPTV. In particular, we aim at capturing two important aspects of channel popularity the distribution and temporal dynamics of the channel popularity. We conduct an in-depth analysis on channel popularity on a large collection of user channel access data from a nation-wide commercial IPTV network. Based on the findings in our analysis, we identify a stochastic model that finds good matches in all attributes of interest with respect to the channel popularity. Furthermore, we propose a method to identify subsets of user population with inherently different channel interests. By tracking the changes of population mixtures among different user classes, we extend our model to a multi-class population model, which enables us to capture the moderate diurnal popularity patterns exhibited by some channels.

1.1.2.2 User Activity Modeling

We perform an in-depth study on several intrinsic characteristics of IPTV user activities by analyzing STB logs collected from an operational nation-wide IPTV system. We further generalize the findings and develop a series of models to capture both the probability distribution and time-dynamics of user activities. We then combine theses models to design an IPTV user activity workload generation tool called SIMULWATCH, which formalizes and quantizes user activities as state transitions. It takes a small number of input parameters to generate synthetic workload traces that mimic a set of real users watching IPTV. We validate all the models and the prototype of SIMULWATCH using the real traces. In particular, we show that SIMULWATCH can model both unicast and multicast traffic accurately, establishing itself as a useful tool in driving the performance study in IPTV systems.

1.1.3 Remarks

Although our studies focus on IPTV network, the methodologies we applied and the tools we developed can be extended to other applications. For instance, SyslogDigest,

the vendor/network independent tool, can be directly used in other networks (e.g., general purpose backbone). The stochastic model we designed for channel popularity may also be applied to other applications with respect to content popularity.

1.2 Related Studies Overview

1.2.1 IPTV Measurement Study

Since IPTV is a relatively new application in the Internet, very few measurement and management researches have been done in this domain. From end-users' perspective, Cha et al. [12] present the first large-scale study of user behaviors in an IPTV system. They characterize user behaviors such as channel popularity and dynamics, viewing sessions, geographic locality and channel surfing probabilities. Our research takes a step further: modeling the user activities based on the measurement study and designing a workload generator, which can be used to evaluate various aspects of IPTV system design and performance with respect to realistic user workload. Whereas our work focuses on IPTV services running on top of a provider backbone, there are a number of peer-to-peer (P2P) based IPTV systems [63, 32] and the measurement study that focuses on P2P IPTV systems [23, 43]. From a network perspective, Mahimkar et al. [35] focuses on characterizing performance impairments and faults in a large-scale IPTV system. While our system (i.e. SyslogDigest) is not specifically designed for troubleshooting, it can benefit complex troubleshooting tasks significantly.

1.2.2 System Log Study

Different learning techniques are proposed to extract knowledge from logs to serve other applications (besides IPTV). Here we provide an overview of these researches from both application and methodology angles.

Applications The most widely applied domain of log learning is World Wide Web. Web log mining deals with the extraction of knowledge from server log files consisting of the (textual) logs that are collected when users access Web servers and

might be represented in standard formats [19]. Typical applications of web log mining are Web personalization [18, 64], adaptive Web sites [40], and user modeling [49, 62, 57]. Besides Web, there are also several studies on failure or anomaly detection based on logs of general purpose servers [39, 54, 53, 33]. In contrast, we are the first to apply learning methods to logs in an IPTV network environment. Moreover, our log study is network-wide, which involves multiple distributed locations instead of a single central server.

Methodology A number of detailed techniques are proposed to extract the desired knowledge of system logs. We discuss two popular types of methods here. The first one is association rule mining, a method to discover interesting relationships between different logs. This method is widely used in web usage mining [19]. For example, Yang et al. apply classic association rule mining to discovering web access patterns for WWW caching and prefetching design [57]. While our method also discovers the association rules among different types of syslogs, it, however, considers both temporal and spatial constraint. The second technique is stochastic modeling, which formalizes the generation of a set of (sequential) logs as the stochastic process. A well-known ON-OFF model is proposed to model LAN traffic [49] and BGP updates [65]. Our model of IPTV user activity is also based on the ON-OFF mode, but it is more sophisticated, in terms of states and transition choices.

1.3 Organization of Dissertation

The rest of the thesis is organized as follows: Chapter 2 presents a background of IPTV network infrastructure and logs we studied. Chapter 3 presents the design of SyslogDigest tool for converting the level-low syslog messages to high-level network events. Chapter 4 presents our study and model of channel popularity and dynamics based on STB logs. Chapter 5 presents our design of workload generator based on modeling user activities in IPTV network. Finally, Chapter 6 concludes the thesis

with a summary of contributions.

CHAPTER II

BACKGROUND

In this chapter we will present the basic architecture of an IPTV system and data sets that will be used in the rest of this dissertation.

2.1 Overview of IPTV Architecture

Figure 1 shows a typical IPTV service system [42]. The SHO (Super Hub Office), the primary source of television content, digitally encodes video streams received externally (e.g., via satellite) and transmits them to multiple VHOs (Video Hub Offices) through a high-speed IP backbone network. The VHOs, each responsible for a metropolitan area, in turn acquire additional local contents (e.g., local news), perform some further processing (e.g., advertisement insertion) and transmit the processed TV streams to end users upon request through access network[1]. Inside a residential home, RG (Residential Gateway) connects to a modem and one or more STBs (Set-Top Boxes) with coaxial cable, receiving and forwarding all data, including live TV streams, STB control traffic, VoIP and Internet data traffic, into and out of the subscriber's home. Finally behind an STB, there connects a TV.

In order to leverage the one-to-many nature of IPTV traffic and ease the bandwidth requirement, video streams are typically delivered using IP multicast in both backbone and access networks. Depending on the TV channel and the codec used, the bit rate of each video stream varies widely from around 1.5Mbps (SDTV with H.264) to around 15Mbps (HDTV with MPEG2). The latency of channel switch is due to

[1]Typically a VHO connects to RGs through a protected optical network using fiber-to-the-node (FTTN) or fiber-to-the-premises (FTTP) technologies. We omit showing the network elements in between as they are not the focus of our work.

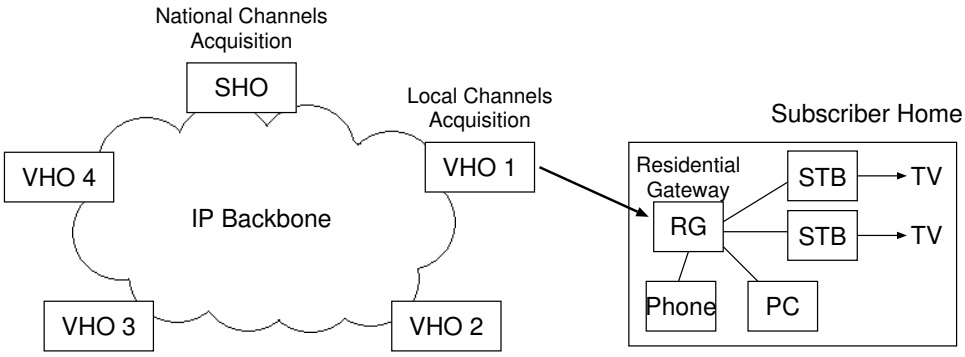


Figure 1: IPTV Architecture

both the multicast group management using IGMP and the video decoding dependency (e.g., waiting for a next I-frame). They could add up to a few seconds, which might make the audience chafe at the bit. This limitation is likely to motivate IPTV users to perform more targeted channel switches than random or sequential channel scans compared to users from conventional TV systems. To address/alleviate this problem[2], some IPTV providers have adopted a fast-channel-switch mechanism in which a server in the VHO sends the STB a unicast video stream (often at a rate higher than multicast rate to avoid long decoding latency) while the STB catches up with the multicast TV stream [44].

Similar to conventional TV users, IPTV users use a vendor/provider customized remote controller to control the STB. For example, one may use *Up/Down* buttons to sequentially switch channels, use *Return* button to jump back to the channel previously watched, or enter a channel number to jump directly to a specific channel. On the other hand, IPTV providers often support additional features, some of which are not offered in conventional TV services. For example, many IPTV providers add the capability for a small number of user-defined *favorite channel list*, so that one can easily switch between or scan through the favorite channels. Furthermore, most STBs support the DVR (Digital Video Recording) feature, in which with the help of a local

[2]How to reduce this delay is an active research area.

Table 1: Syslog messages example

Message	Vendor	Message timestamp	Router	Message-type/error-code
m1	V1	2010-01-10 00:00:15	r1	LINEPROTO-5-UPDOWN
m2	V1	2010-01-10 00:00:15	r5	LINK-3-UPDOWN
m3	V1	2010-01-10 00:00:15	r8	SYS-1-CPURISINGTHRESHOLD
m3	V1	2010-01-10 00:00:26	r8	SYS-1-CPUFALLINGTHRESHOLD
m4	V2	2010-01-10 00:00:23	ra	SNMP-WARNING-linkDown
m5	V2	2010-01-10 00:00:24	rb	SVCMGR-MAJOR-sapPortStateChangeProcessed
m6	V2	2010-01-10 00:00:26	ra	SNMP-WARNING-linkup

Table 2: Syslog messages example (cont.)

Message	Detailed message
m1	Line protocol on Interface Serial13/0.10/ 20:0, changed state to down
m2	Interface Serial2/0.10/2:0, changed state to down
m3	Threshold: Total CPU Utilization(Total/Intr): 95%/1%, Top 3 processes (Pid/Util): 2/71%, 8/6%, 7/3%
m3	Threshold: Total CPU Utilization(Total/Intr) 30%/1%.
m4	Interface 0/0/1 is not operational
m5	The status of all affected SAPs on port 1/1/1 has been updated.
m6	Interface 0/1/0 is operational

hard drive, a user can pause, rewind, fast forward (up to live play), and record the TV program being played. Some IPTV providers support one channel being recorded to DVR while another channel being played live on TV. Also depending on the IPTV provider, IPTV users can enjoy many advanced features such as Picture-In-Picture (PIP), on-line gaming and chatting, and personalized web services on their TVs.

2.2 Router Syslog

In this section, we provide an overview of the syntax and semantics of router syslog messages. Similar to syslogs on computer servers, router syslogs are the messages that routers generate to record the hardware and software conditions observed by them, such as link and protocol-related state changes (e.g., down or up), alarming environmental measurements (e.g., high voltage or temperature), and warning messages (e.g., triggered when BGP neighbors send more routes than the router is configured to allow). Although syslog messages are intended primarily for tracking and debugging router software and hardware problems, they can be extremely valuable to network

operators in managing networked services and troubleshooting network incidents. For this reason syslogs are usually collected on all routers inside a network, especially an ISP network, and a syslog (transmission) protocol [20] is standardized and widely supported by router vendors to transmit syslog messages from routers to one or more syslog collector(s).

While the syslog protocol – for transmitting syslog messages – is standardized, the syslog messages themselves are not. They are essentially free-form texts, the syntax and semantics of which vary among router vendors and router operating systems. Table 1 and 2shows a few examples of syslog messages from two router vendors. We can observe only a minimal structure in a syslog message: (1) a timestamp indicating when the message is generated, (2) the identifier of the router that generates the message (called *originating router*), (3) message type, also known as the error code, indicating the nature of the problem, and (4) detailed message information generated by the router OS. In order to correlate syslog messages across routers, the clocks (for generating the timestamps) on these routers need to be synchronized often through the Network Time Protocol (NTP), which is the case in our studies.

The detailed message information (aforementioned field (4)) in router syslogs is quite ad hoc in nature. They are simply free-form texts "printf"-ed by router operating systems with detailed information such as the location, state/status, or measurement readings of a alarming condition embedded in them. For example, in the first message (m1), (`Line protocol on Interface Serial13/0.10/20:0, changed state to down`), the `Serial13/0.10/20:0` part indicates the network interface at which the layer-2 line protocol (PPP) has been impacted and the `down` part indicates the status/state of the line protocol. The rest of it, `Line protocol on Interface ..., changed state to ...`, can be viewed as the sub type for this type of syslog message. It is worth noting that there are often multiple sub types associated with the same syslog type (error code). The combination of syslog type

and this sub type can be used as a *template* to uniquely identify the class of network conditions that the syslog message describes.

In some syslog versions, the error code field contains severity information. For example, in the first four lines (vendor V1 syslogs) of Table 1, the number between two "-" symbols is the severity level of the messages – the smaller the number is, the more important the message is *perceived by the originating router*. It is important to note, however, that the severity level of a syslog message is assigned by the equipment vendor based on the perceived impact of the observed event on the performance and proper functioning of the local network elements. It does not in general translate into the severity of the impact that this event will have on the performance and proper functioning of the overall network and therefore cannot be directly used for to rank/order the importance of events for network service management purposes. For example, syslog messages concerning router CPU utilization rising above or falling below a given threshold (lines 3 and 4 in Table 2) have been considered more important (smaller in severity number) than those concerning an adjacent link changing its state to "down" (line 2 in Table 2) in some router OS. Network operators would certainly disagree in this case.

Router configuration tools usually allow network operators to specify a threshold such that potential syslog messages with severity levels above or equal to it will be recorded. In this study, we collect syslogs at such "informational" level (usually the default setting). Depending on the network conditions, the amount of router syslog messages in an operational network varies. In the large-scale ISP network (hundreds to thousands routers) that we study in the dissertation, there are typically hundreds of thousands to millions of messages per day.

2.3 STB logs

In IPTV networks, the set-top box is a small computer providing two-way communications on an IP network and decoding the video streaming media. IP set-top boxes have a built in home network interface which can be Ethernet or one of the existing wire home networking technologies such as HomePNA or the ITU-T G.hn standard, which provides a way to create a high-speed (up to 1 Gigabit/s) Local area network using existing home wiring (power lines, phone lines, and coaxial cables).

The software running on STB keeps logging the events like STB power on/off, channel switching, system crashing, etc. We particularly focus on the logs that are directly related to users activities. The STB logs we use in this dissertation are collected from a large scale IPTV provider in the United States, which has over one million subscribers and over two million STBs spread throughout four different time zones. As a privacy protection, only anonymous data was used in this study; no information that could be used to directly or indirectly identify individual subscribers was included. By collecting data from the anonymous STB logs, we were able to model activities such as turning on/off STBs, switching channels, and playing live or recorded TV program. In particular, we associate each activity recorded in the anonymous STB logs with its origin STB and a timestamp (which is at the precision of one second). To account for different time zones, we map the STBs to their metropolitan area and convert the associated timestamps into their local time.

CHAPTER III

ROUTER SYSLOG MINING

3.1 Introduction

Router syslogs are messages that a router logs to describe a wide range of events observed by it. Although router syslogs are primarily designed and intended for router vendors to track and debug router software/hardware problems, they are also widely used by the network service providers as one of the most valuable data sources in monitoring network health and in troubleshooting network faults and performance anomalies. However, working with raw syslog messages is not easy from network service management prospectives. First, router syslog messages are essentially free-form text with only a minimal structure. The type of information that is logged and its formats vary among different vendors and router operating systems. Second, router syslog messages are often too low-level. They do not directly translate into what actually happened in the network (i.e., network events) without meaningful abstraction and aggregation. Third, not every router syslog message is an indication of an occurrence of an incident that could potentially impact network services. For example, some router syslog messages are generated purely for debugging purposes and have no implications on network services.

Although large ISP network, consisting of thousands of routers, is expected to observe millions of information-rich syslog messages per day, the lack of sentence structures in log messages and relational structures across messages prevents router syslogs from being utilized to its fullest extent in various network management applications. Network monitoring systems typically rely on the input of domain knowledge

to be able to focus on a rather small (yet deemed important) subset of syslog messages. For example, commercial network management tools such as Lonix [1] and NetCool [2] focus on a small set of messages concerning network faults. The syntax and the relations of these syslog messages are explicitly captured to allow for automated parsing and understanding. When certain patterns of syslog messages are observed in the network, alarms are triggered and operation tickets are issued. The parsers and the message relationship models therein need to be constantly updated to keep up with network changes. For example, routers upgraded to a new operating system may introduce new syslog message formats and hence require a new parser. Network issues can also fly under the radar if they do not match syslog patterns already programmed. Meanwhile, when troubleshooting a network event, network operators have to focus on a narrow time-window and a specific router in examining the raw syslog messages in detail, in order to avoid being overwhelmed by the number of syslog messages. It is a very time-consuming and inefficient process when a complicated network incident involves a large number of messages. Moreover, by limiting to a small scope, operators lose the sight of "big picture", such as the information regarding the frequency or the scope of the kind of network events under investigation. Finally, network auditing and trend analysis systems have to rely on rather simple frequency statistics when it comes to router syslogs. For example, MERCURY [37] detects network behavior changes due to network upgrades by tracking the level-shift of message frequencies of individual syslogs. Knowing the relationship across syslog messages would make the result much more meaningful.

In this work, we focus on proactively mining network events from router syslogs. In particular, we design an automated system (called SyslogDigest) that transforms and compresses massive low-level minimally-structured syslog messages into a small number of meaningful and prioritized high-level network events. SyslogDigest is vendor/network independent and does not require domain knowledge on expected

network behaviors. It automatically identifies signatures of different types of syslog messages in terms of both their syntax and temporal patterns (e.g., interarrival time of each type of syslog messages). In addition, SyslogDigest learns association rules between different types of syslog messages both on the same router and across routers. The combination of the signatures and association rules of syslog messages enables us to group them into meaningful network events. Furthermore, SyslogDigest prioritizes network events using a number of severity/importance measures. This allows network operators to quickly identify and focus on important network events. More importantly, this enables complex network troubleshooting, in which thousands to tens of thousands of syslog messages on multiple routers may be related to a single network event and may need to be identified out of millions of syslog messages for examination. SyslogDigest systematically classifies and groups these syslog messages into a single meaningful event, making obsolete the long and error-prone manual grouping process in the current practice. This automated grouping capability not only enables monitoring overall network health and tracking the appearance and evolvement of network events, but also allows for much better network visualization, since visualizing such network events provides a much clearer and more accurate big picture of what happened in the network than visualizing raw syslog messages.

We apply SyslogDigest to router syslog messages collected from operational IP networks like IPTV backbone network. We show that SyslogDigest outputs prioritized network events that are over three orders of magnitude fewer in number and have much better usability than raw syslog messages. Using real applications, we further demonstrate that they provide critical input to not only network troubleshooting but also network health monitoring and visualization.

To summarize, we make four major contributions in this work.

1. We designed an automated tool SyslogDigest that transforms massive volume of router syslog messages into a much smaller number of meaningful network

events.

2. We developed effective techniques tailored to our problem domain to systematically identify signatures of syslog messages, learn association rules that capture network behaviors over time, group relevant raw syslog messages across multiple routers into network events, and label and prioritize network events based on their nature and severities.

3. We conducted large-scale experiments on real router syslog data collected from two large operational IP networks and demonstrated that SyslogDigest is able to transform millions of syslog messages into network events that are over three orders of magnitude fewer in number and smaller in size.

The remainder of this chapter is organized as follows. Section 3.2 presents an overview of SyslogDigest system and Section 3.3 describes the detailed methodologies it uses. Section 3.4 presents the evaluation results based on router syslog data collected from two large operational IP networks. We present some applications of SyslogDigest in Section 3.5 and related work in Section 3.6. Finally, Section 3.7 concludes the chapter.

3.2 System Design Overview

The goal of SyslogDigest is to automatically transform and compress low-level minimally-structured syslog messages into meaningful and prioritized high-level network events. Our key observation is that one single (sometimes intermittent) condition on a network element, such as an network interface, router hardware, and router software, can often trigger a large number of different messages across time, protocol layers, and routers. For example, messages $m1$ to $m16$ in Table 3 are all triggered by the same network condition: the link between routers $r1$ and $r2$ flapped a couple of times. Based on this observation, our basic idea is to automatically construct a *network event*

Table 3: Toy Example. Router r1's interface Serial1/0.10/10:0 is connected to r2's inteface Serial1/0.20/20:0.

index	timestamp	router	message-type/error-code	detailed message
m1	00:00:00	r1	LINK-3-UPDOWN	Interface Serial1/0.10/10:0, changed state to down
m2	00:00:00	r2	LINK-3-UPDOWN	Interface Serial1/0.20/20:0, changed state to down
m3	00:00:01	r1	LINEPROTO-5-UPDOWN	Line protocol on Interface Serial1/0.10/10:0, changed state to down
m4	00:00:01	r2	LINEPROTO-5-UPDOWN	Line protocol on Interface Serial2/0.20/20:0, changed state to down
m5	00:00:10	r1	LINK-3-UPDOWN	Interface Serial1/0.10/10:0, changed state to up
m6	00:00:10	r2	LINK-3-UPDOWN	Interface Serial1/0.20/20:0, changed state to up
m7	00:00:11	r1	LINEPROTO-5-UPDOWN	Line protocol on Interface Serial1/0.10/10:0, changed state to up
m8	00:00:11	r2	LINEPROTO-5-UPDOWN	Line protocol on Interface Serial2/0.20/20:0, changed state to up
m9	00:00:20	r1	LINK-3-UPDOWN	Interface Serial1/0.10/10:0, changed state to down
m10	00:00:20	r2	LINK-3-UPDOWN	Interface Serial1/0.20/20:0, changed state to down
m11	00:00:21	r1	LINEPROTO-5-UPDOWN	Line protocol on Interface Serial1/0.10/10:0, changed state to down
m12	00:00:21	r2	LINEPROTO-5-UPDOWN	Line protocol on Interface Serial2/0.20/20:0, changed state to down
m13	00:00:30	r1	LINK-3-UPDOWN	Interface Serial1/0.10/10:0, changed state to up
m14	00:00:30	r2	LINK-3-UPDOWN	Interface Serial1/0.20/20:0, changed state to up
m15	00:00:31	r1	LINEPROTO-5-UPDOWN	Line protocol on Interface Serial1/0.10/10:0, changed state to up
m16	00:00:31	r2	LINEPROTO-5-UPDOWN	Line protocol on Interface Serial2/0.20/20:0, changed state to up

by grouping together *related* messages, *i.e.*, those triggered by the same network condition, and then prioritize network events using a number of severity/importance measures. The related messages usually exhibits certain temporal or spatial pattern. The challenge is how to automatically determine such patterns or the lack of them among syslog messages without relying on domain knowledge being manually input and updated (by network operators). SyslogDigest accomplishes this goal through a two-step process. In this first step, an *offline domain knowledge learning* component automatically extracts relevant domain knowledge from raw syslog data. In the second step, an *online processing* component will rely on such acquired domain knowledge and other available information (e.g., temporal closeness of messages) to finally group related messages into high-level events and present the prioritized results. In the rest of this section, we will provide a high-level overview of SyslogDigest's architecture, shown in Figure 2, and the functionalities of its components, by working out a running example shown in Table 3, where 16 syslog messages are eventually grouped into 1 high-level network event.

3.2.1 Offline Domain Knowledge Learning

The offline domain knowledge learning component automatically acquires domain knowledge such as the syntax/semantics of syslog messages, the relationships among various message templates, and the detailed location information obtained from raw syslog data. The offline domain knowledge learning proceeds as follows.

First, it automatically extract message templates from the historical syslog messages. To deal with the challenge posed by minimally-structured messages from different vendors and/or different OS versions that adopt different template syntax/semantics systems, we develop an effective signature identification technique in which messages are decomposed into whitespace-separated words and a frequent word

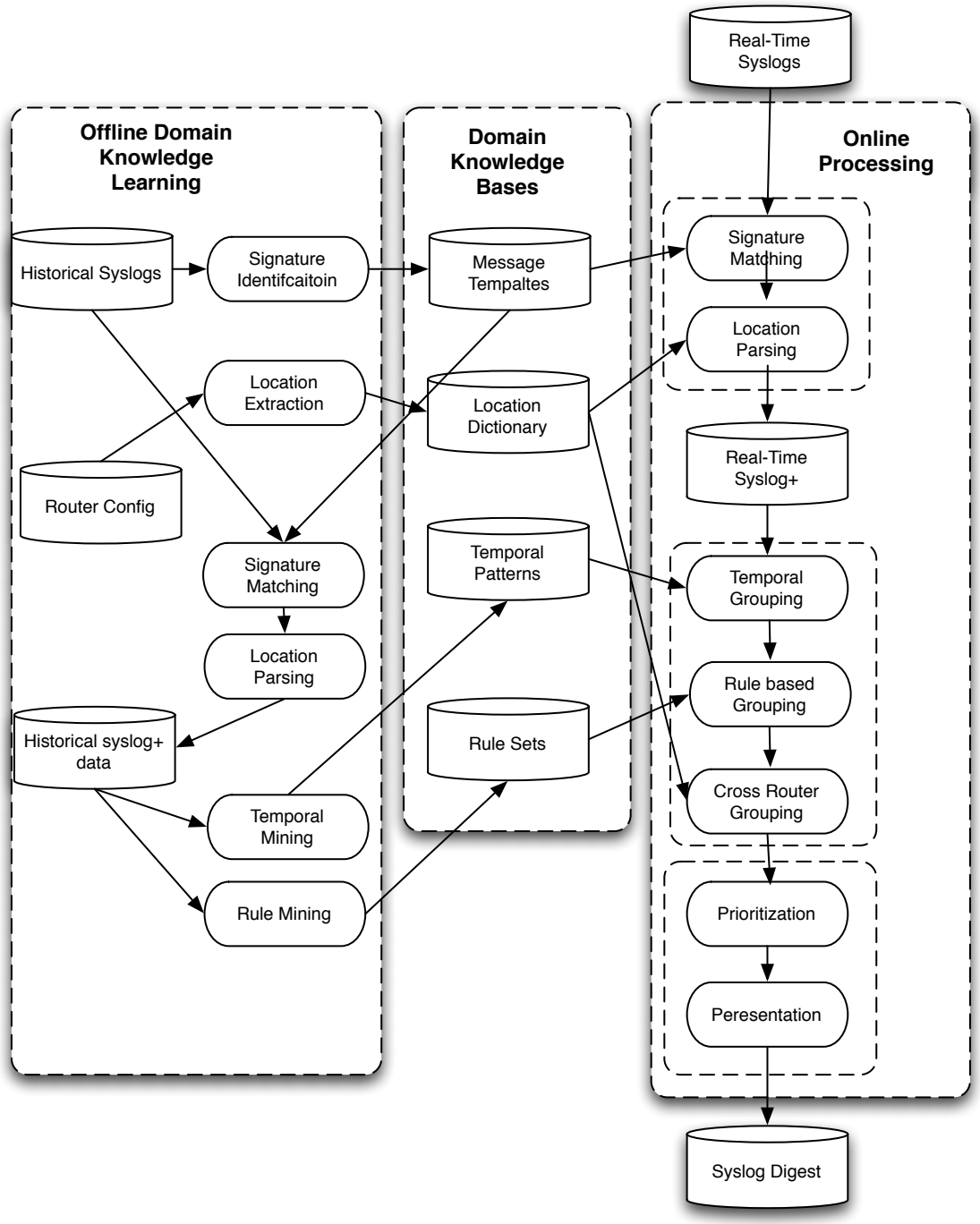


Figure 2: SyslogDigest Architecture.

(excluding those words denoting specific information like locations) sequence is considered as a template. For example, without going into the details of the signature identification technique, it is intuitive to see the following template can be extracted: `t1: LINK-3-UPDOWN Interface ..., changed state to down`, `t2: LINEPROTO-5-UPDOWN Line protocol on Interface ..., changed state to down`, `t3: LINK-3-UPDOWN Interface ..., changed state to up`, and `t4: LINEPROTO-5-UPDOWN Line protocol on Interface ..., changed state to up`.

Second, in order to parse the location information embedded and hidden in syslog messages, SyslogDigest learns and builds a "dictionary" for the locations in each router's syslog messages based on its configuration files (i.e., router configs). One might be tempted to parse the location information purely based on the vendor manual description for each message. This can however be overly expensive due to the high diversity of message formats and large number of messages. Our solution is based on one key observation: a router almost always writes to syslog messages only the location information it knows, i.e, those configured in the router. Our solution is therefore to parse router configs (much better formatted and documented than syslog messages) to build a dictionary of its locations offline. With router configs, we also build the mapping between different locations, e.g., from an interface name to its IP address, and the hierarchical location relationship between interfaces, ports and linecards, and network topology such as the interfaces connecting two routers. These mappings enable us to group syslog messages with related locations. In the example shown in Table 3, the location dictionary will contain interfaces `r1,Serial1/0.10/10:0` and `r2,Serial1/0.20/20:0` and also the information that these two interfaces are connected to each other.

Third, to learn the relationship among different templates, we first augment each historical syslog message with additional information, including message template

and its location information, by matching it with the templates and locations previously learned. In our example, $m1$ is augmented as `m1|t1|r1,Serial1/0.10/10:0` (with template `t1` and location `r1,Serial1/0.10/10:0` appended), and $m4$ is augmented as `m4|t2|r2,Serial1/0.20/20:0`, and so on. The resulting messages are called **Syslog+** messages in our system. We then apply association rule mining techniques to Syslog+ messages to learn the relationships (i.e., associations) among different messages with different templates. A rule of thumb is that if two messages frequently occur *close* enough in time and at *related* locations (postpone the details to Section 3.3), they are considered *associated* and should be grouped together. For example, if syslog messages in templates $t1$ and $t2$ often happen close together, the association $t1, t2$ will be declared.

Finally, SyslogDigest learns the temporal patterns of each template from Syslog+ message. The intuition is that messages with same template can appear periodically (e.g., due to various network timers), and if so, these events can be grouped together. Such kind of periodicity can be learned offline through measurements of corresponding interarrival times and predictions based on their linear regression.

The above domain knowledge learning process will be periodically run (offline) to incorporate the latest changes to router hardware and software configurations, and the acquired domain knowledge will be used as input to the online SyslogDigest System.

3.2.2 Online SyslogDigest System

The online system takes the real-time syslog message stream as well as the previously learned domain knowledge as input, and output meaningful prioritized network

events, in a two-step process. The first step is to augment the real-time syslog messages with template and location information just like in the aforementioned offline process, and output the (augmented) Syslog+ messages in an online fashion.

The second step is to group related Syslog+ messages together to construct meaningful network events. We propose three grouping methods: **temporal grouping, rule-based grouping and cross-router grouping**. **Temporal grouping** targets at messages with the same template on the same router. It groups together messages that have the same template and happen periodically, where such periodicity (temporal patterns) is already detected during the offline temporal mining. In our example, it is intuitive to see that after the temporal grouping, $m1, m5, m9, m13$ (with common template $t1$ and common location `r1,Serial1/0.10/10:0`) are grouped together. So are $(m2, m6, m10, m14)$, $(m3, m7, m11, m15)$, and $(m4, m8, m12, m16)$. **Rule-based grouping** targets the messages with different templates on the same router. Based on the association rules learned by the offline learning component and saved in a domain knowledge base, this method groups messages that have different templates, but happen close together in time. In our example, after the rule-based grouping, messages $m1, m3, m5, m9, m11, m13, m15$ are grouped together due to association rule $\{t1, t2\}$ with common location `r1,Serial1/0.10/10:0`. Messages $m2, m4, m6, m8, m10, m12, m14, m16$ are grouped together due to association rule $\{t3, t4\}$ and common location `r2,Serial1/0.20/20:0`. Finally, the **cross-router grouping** method will group together messages with locations that are on different routers yet closely connected (e.g., two ends of one link, two ends of one BGP session), determined by the location dictionary in the domain knowledge base. They will be applied to the Syslog+ messages in this order (justifications explained later). After these three grouping methods are applied in the order that they are described above to the online syslog message stream, we obtain groups of messages, each of which is considered a single *network event*, which are much smaller in number compared to

the raw syslog messages. In our example, $m1$ to $m16$ are eventually grouped together into a single network event.

The final step is to prioritize and present the network events based on their perceived importance to network operators. Various factors are considered together to determine the (relative) importance of an event, including the number of messages the event (group) contains, the frequency of this event type in the history and the perceived impact of this event on network health. Each event is presented as a well-formatted text line, with multiple fields summarizing the information contained in the raw syslog messages that are grouped into this event, including the start/end timestamps of this event, an location field that records where the event happens, an event type field that is more informative than its counterpart in individual raw syslog messages, and an index field that allows us to retrieve these raw syslog messages if necessary.

The presentation of Table 3 could be `2010-01-10 00:00:00|2010-01-10 00:00:31|r1 Interface Serial1/0.10/10:0 r2 Interface Serial1/0.20/20:0|link flap, line protocol flap`.

3.3 Syslog Mining Methodologies

In this section, we present the detailed methodologies used in both the offline learning and the online digesting systems.

3.3.1 Offline Learning Methodologies

There are several basic aspects that we need to learn from syslog messages: message templates, location information, temporal patterns of message templates and template relationship.

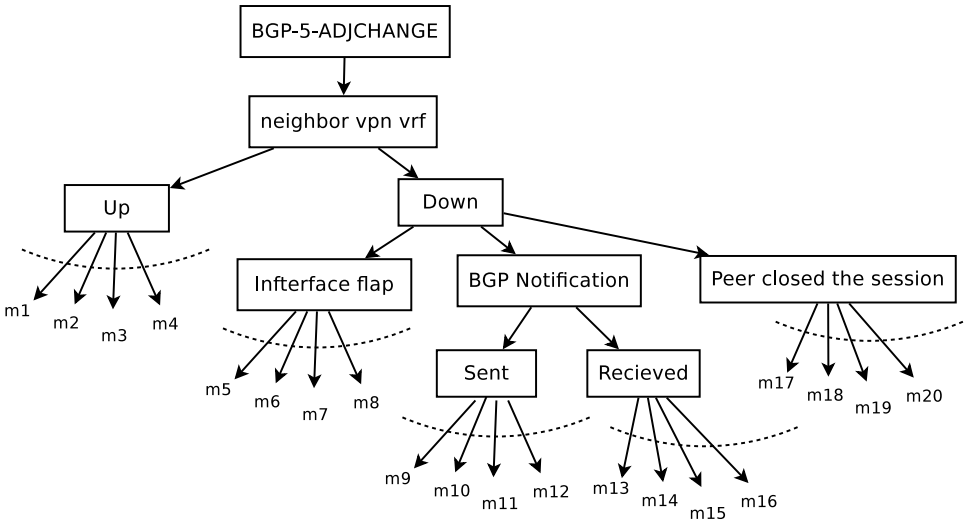


Figure 3: Sub type tree construction example.

3.3.1.1 Message Template Learning

As mentioned earlier, raw syslog messages have little structure. Although there is a message type field to describe the characteristics of messages, for each message type there can be multiple sub types. For example in Table 4, while all messages belong to the same type "BGP-5-ADJCHANGE" and correspond to BGP adjacency change in MPLS VPN, the details of these messages (and hence their sub types) can be different. At issue is how to automatically construct such sub types and combine them with the message type to form the template *without intervention from domain experts*. In our example, the neighboring IP addresses and the VRF[1] IDs (e.g., VRF 1000:1001 in m_1) differ from one message to another, but when these two fields are *masked* (i.e., replaced by the same symbol, say asterisk, as shown in Table 5), there are only five distinct "structure" types, or sub types as we call them. In practice, however, it is not easy to manually find all masked parts without domain knowledge, because not all needed-masked parts have obvious pattern like IP address or VRF ID.

[1]VRF stands for Virtual Routing and Forwarding. It is a technology that allows multiple instances of a routing table to co-exist within the same router at the same time. VRF is a common technique used in VPN environment. The VRF ID XXX:XXXX is a simple conceptional name.

Table 4: The messages belong to the same message type (BGP-5-ADJCHANGE)

m_1	neighbor 192.168.32.42 vpn vrf 1000:1001 Up
m_2	neighbor 192.168.100.194 vpn vrf 1000:1002 Up
m_3	neighbor 192.168.15.78 vpn vrf 1000:1003 Up
m_4	neighbor 192.168.108.38 vpn vrf 1000:1004 Up
m_5	neighbor 192.168.0.26 vpn vrf 1000:1004 Down Interface flap
m_6	neighbor 192.168.7.6 vpn vrf 1000:1001 Down Interface flap
m_7	neighbor 192.168.0.238 vpn vrf 1000:1003 Down Interface flap
m_8	neighbor 192.168.2.114 vpn vrf 1000:1002 Down Interface flap
m_9	neighbor 192.168.183.250 vpn vrf 1000:1002 Down BGP Notification sent
m_{10}	neighbor 192.168.114.178 vpn vrf 1000:1003 Down BGP Notification sent
m_{11}	neighbor 192.168.131.218 vpn vrf 1000:1001 Down BGP Notification sent
m_{12}	neighbor 192.168.55.138 vpn vrf 1000:1000 Down BGP Notification sent
m_{13}	neighbor 192.168.1.13 vpn vrf 1000:1000 Down BGP Notification received
m_{14}	neighbor 192.168.12.241 vpn vrf 1000:1002 Down BGP Notification received
m_{15}	neighbor 192.168.155.66 vpn vrf 1000:1003 Down BGP Notification received
m_{16}	neighbor 192.168.254.29 vpn vrf 1000:1004 Down BGP Notification received
m_{17}	neighbor 192.168.35.230 vpn vrf 1000:1004 Down Peer closed the session
m_{19}	neighbor 192.168.171.166 vpn vrf 1000:1001 Down Peer closed the session
m_{19}	neighbor 192.168.2.237 vpn vrf 1000:1002 Down Peer closed the session
m_{20}	neighbor 192.168.0.154 vpn vrf 1000:1003 Down Peer closed the session

Table 5: Sub message types of BGP-5-ADJCHANGE

M_1	neighbor * vpn vrf * Up
M_2	neighbor * vpn vrf * Down Interface flap
M_3	neighbor * vpn vrf * Down BGP Notification sent
M_4	neighbor * vpn vrf * Down BGP Notification received
M_5	neighbor * vpn vrf * Down Peer closed the session

Our template learning approach is inspired by the signature abstraction used in spam detection [52]. The high level idea is that a signature, which corresponds to an aforementioned sub type node, is a combination of words with high frequency. We decompose messages into words separated by whitespace. For each type of message, we construct a tree structure to express the template (sub type) hierarchy based on the input messages (e.g., $m_1, m_2, \ldots, m_{20}$), shown in Figure 3. We say that a word *associates with* a message when the word appears in the message. The detailed construction algorithm follows breath-first search tree traversal. We first use the message type (e.g., BGP-5-ADJCHANGE) as the root of the tree. All messages are associated with this message type. Then given the parent node, we look for the most frequent combination of words which can associate with most messages that the parent node can associate with, and make this combination as a child node. We repeat this process to create child nodes based on remaining messages, until all messages have been associated. We then recursively proceed to the child nodes and repeat the process. Finally we prune the tree until it has the desired degree properties as follows. If a parent node has more than k children, we will discard all children to make the parent a leaf itself. Now each path from root to the each leaf become one template (type + sub type). The intuition of this pruning is that on the one hand, there are only a few sub types for each message type, on the other hand, usually there would be many more messages associated with each sub type. For example, there should be many IPs and VRF addresses associated with each sub type given enough data. In practice, We choose $k = 10$ based on our experience that no message type has more than 10 sub types.

Our template inference approach is quite generic because it is based on the words frequency as opposed to text semantic. However, we use an implicit assumption that the variable part of the syslog messages would appear as many distinct values given enough historical data. This assumption is surely not always true. For example, if

certain protocol are enabled only on one type of network interface, say GigabitEthernet, then the "GigabitEthernet" part of the message may be falsely included in the syslog template of the protocol messages. However, this would have negligible impact on the final outcome of the grouping result, since the "GigabitEthernet" in this case contains as much information as the syslog sub type and hence there is no need to exclude it.

3.3.1.2 Location Information Learning

In a typical syslog message, we only have a router id field as the basic location information, but this is clearly not enough. For example, some events occur on a particular physical port while some other events occur on multiple logical links (e.g., IP links). Such detailed location information is essential for understanding what is going on in the network.

Figure 4 shows the generic location hierarchical structure. We classify the basic components here into physical ones and logical ones. The physical ones have a clear hierarchical structure from top to bottom. The arrow here illustrates a "one-to-many" relationship. For example, one router have multiple slots, each slot can have multiple ports and etc. Besides physical hierarchy, there are some logical configurations, but they will eventually map to some physical component. For example, one multilink/bundlelink can be mapped to multiple physical interfaces. Based on router configuration data, we can extract offline the hierarchy in Figure 4 specific to each router, cross-router location relationships such as neighbor links, etc.

The key question here is how to automatically exploit such kind of location information from each message. First, we know the particular format of these location information embedded into the messages. For example, the IP address has the format XXX.XXX.XXX.XXX, the port has the format X/X/X , etc. Compared with various parts of message we need to mask during template learning, the number of location

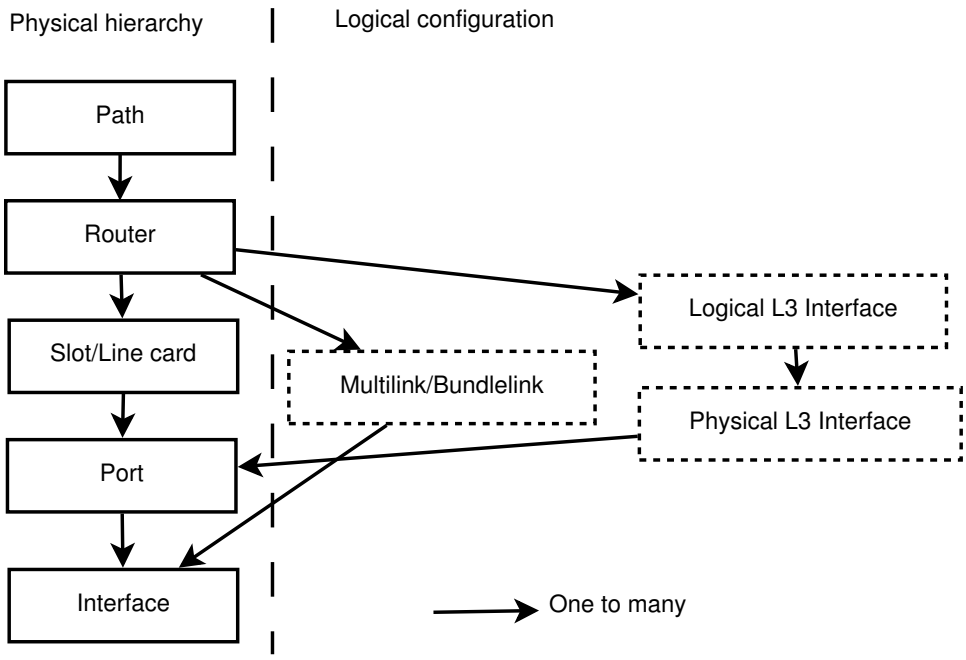


Figure 4: Location hierarchy

format patterns are limited, which is manageable for extraction using predefined patterns. However, the naive pattern matching is not sufficient to extract needed location addresses, mainly because more than one location pattern (no matter whether they are all needed) can be found in each message. For example, multiple IP addresses can be found in one message. One could belong to the router itself, one could belong the neighbor router, and it is also possible there are some remote (e.g., remote session connection) or invalid IPs (e.g., scanning attacks). To understand the exact meaning of multiple location patterns, especially the belongings, we correlate these locations with router configuration data. For example we can verify if this IP belongs to the router or its neighbors. Note that the acquired location information and location hierarchy will be used during the offline rule mining and online grouping.

3.3.1.3 Learning Temporal Patterns of Templates

Once we identified the message templates, we learn the temporal patterns of the message templates, i.e, the interarrival patterns. This learned knowledge will be used

in the online temporal grouping component.

We observe that if a particular template of message occurs periodically, the corresponding messages naturally form a number of clusters in the time series. For example, Figure 5 shows that one controller goes up down many times within a short interval because controller is unstable during the interval. Another example in Figure 6 shows that TCP bad authentication message has periodic occurrences, likely due to the underling timer configuration, or outside impact, e.g., scanning patterns.

In order to learn such temporal patterns, i.e. , the interarrival time within each cluster, we make a basic assumption, that is the impact of current message on the further message with the same template will exponentially decay. Such assumption is widely used for time series analysis and system measurement purpose [10, 59]. Our learning method is based on the interarrival sequence $S_1, S_2, \ldots$ for each message template. We compute the (predicted) exponential weighted moving average (EWMA) of interarrival time t, $\hat{S}_t$.

$$\hat{S}_t = \alpha \cdot S_{t-1} + (1 - \alpha) \cdot \hat{S}_{t-1}$$

where $\alpha \in (0, 1)$ is the weighting factor. A higher α discounts older observations faster. Intuitively, if the messages belong to the same cluster, in other words, there is a periodic pattern within the group, then the predicted value $\hat{S}_t$ should not be far away from the real one S_t. Consequently, we assume that if $S_t \leq \beta \cdot \hat{S}_t$ where ($\beta >= 1$), which means the real interarrival time is no much larger than predicted one, we view that the message belongs to the same group. Otherwise, there is another new group. Here parameter β defines a threshold for grouping. Larger β means tolerating larger intervals in the group.

The offline learning component uses long-term historical data to infer the proper value of parameters α and β and can be updated periodically. The actually values of these parameters will be discussed in Section 3.4. These parameters are used in the online temporal grouping component.

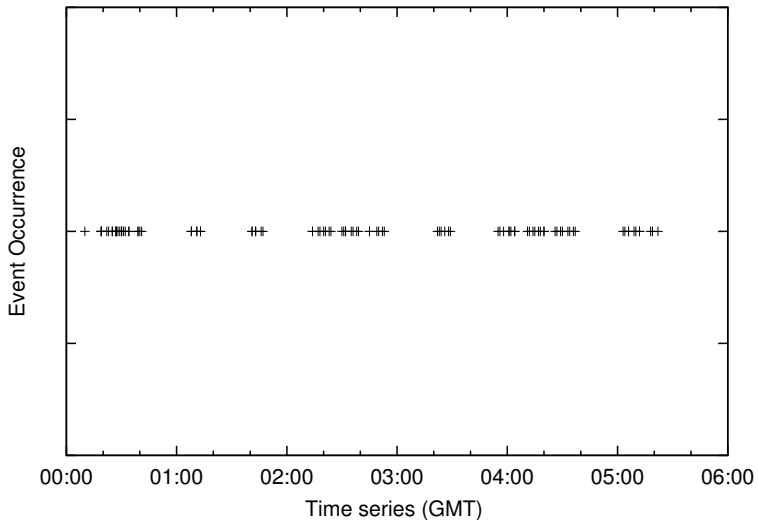


Figure 5: Controller up/down example.

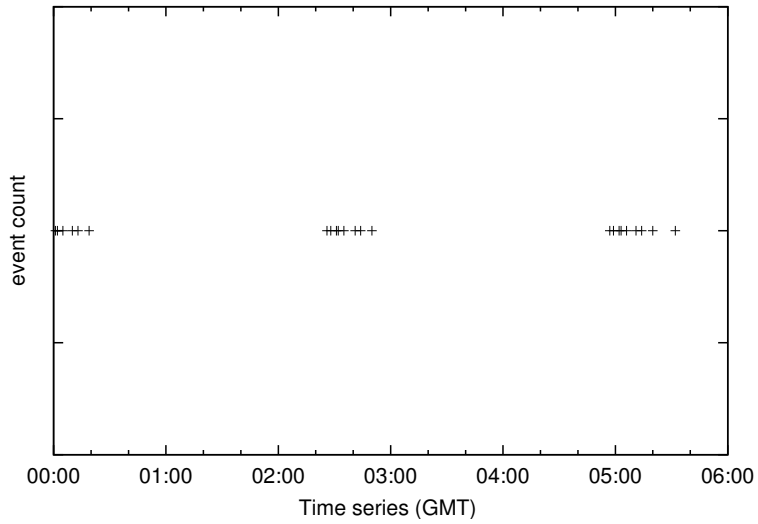


Figure 6: TCP bad authentication example

3.3.1.4 Template Relationship (Rule) Learning

In order to group different messages together to extract network events, one natural thinking is to discover some implicit rules among different templates. Some rules are very intuitive. For example, layer-1 link failures (LINK-3-UPDOWN) often trigger layer-2 failures (LINEPROTO-5-UPDOWN). Some others are much more subtle. As we explained before, we cannot rely on domain experts to compile and update a

complete rule set given the large number of templates. We need a systematic way to identify such rules. This turns out to be a typical association rule mining problem. Association rules describe items that occur frequently together in a dataset and are widely-used for market basket analysis. Following the original definition by Agrawal et al. [5] the problem of association rule mining is defined as: Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n binary attributes called *items*. Let $D = \{t_1, t_2, \ldots, t_m\}$ be a set of transactions called the *database*. Each transaction in D has a unique transaction ID and contains a subset of the items in I. A *rule* is defined as an implication of the form $X \Rightarrow Y$ where $X, Y \in I$ and $X \cap Y = \emptyset$. To select interesting rules from the set of all possible rules, constraints on various measures of significance and interest can be used. The best-known constraints are minimum thresholds on support and confidence. Support supp(X) of an itemset X is defined as the proportion of transactions in the data set which contain the itemset. The confidence of a rule is defined as

$$conf(X \Rightarrow Y) = \frac{supp(X \cap Y)}{supp(X)}$$

In our problem setting, each message template is one item. In order to construct the transactions, we use a sliding window W. It starts with the first message, and slides message by message (sorted messages on the time series). Each time there is one transaction. In one such transaction, the message templates in the window W are considered as the items showing up.

Note that we only consider pair wise association, or $|X| = |Y| = 1$. In other words, each rule only contains two templates. The reason is, first, the computation complexity is low, and second, it is relatively easy to verify the generated rule sets. Domain experts only need to verify the relationship of two templates per rule. The disadvantage of pair wise is that based on these rules we cannot group more than two templates each time, but since we assume the transition property during rule-based grouping discuss later in Section 4.2.2), the final digest will combine multiple templates together.

Until now, we assume that the rules are generated based on static dataset. But ideally we want to learn the rules continuously. In order to adaptively adjust the rules, we use the following conservative way. First, we training the a period of data to generate the basic rule sets. Then we keep change the rules periodically (e.g., each week). The new rules $X \Rightarrow Y$ should be added when $supp(X)$ and $conf(X \Rightarrow Y)$ are above the threshold. Old rules $X \Rightarrow Y$ should be deleted when updated $conf(X \Rightarrow Y)$ is below the threshold, no matter what $supp(X)$ is. Such conservative deletion approach ensures that we do not delete the rules because X are not common in this updating period (it is quite possible X become common again soon).

3.3.2 Online System Methodologies

The online system takes the real-time Syslog+ data (with message template and location information) and the offline-learned domain knowledge as input, groups related messages together and construct prioritized event. Roughly speaking, if two messages occur close in time and locations, they are related with high probability. While the temporal closeness between two messages can simply the characterized by the closeness of their timestamps, the characterization of spatial closeness is more subtle. We model various location types in the location hierarchy shown in Figure 4. We say the two locations are *spatially matched* when they can be mapped to same location in the hierarchy. For example, suppose one message happens on slot 2 and another one message on the same router happens on interface series 2/0/0:1. They are considered spatially matched because the later message's location (2/0/0:1) can be mapped upwards in Figure 4 to slot 2 (the first digit before the backslash interface series of 2/0/0:1).

3.3.2.1 Temporal Grouping

Online temporal grouping uses the same methodology as offline temporal patterns learning, presented in Section 3.3.1.3. Similar to Section 3.3.1.3 , if the real inter-arrival time $S_t \leq \beta \cdot \hat{S}_t$, then the messages belong to the same group, otherwise there is separate group.

We also introduce two thresholds S_{min} and S_{max}. S_{min} is the minimal interarrival time, and S_{max} is the maximum. The reason of introducing S_{max} is that the our algorithm cannot guarantee convergence. Each time we only guarantee that the S_t is not too large. But when $\hat{S}_t$ increase, S_t can grow exponentially. If the real interarrival time is smaller than S_{min}, then we consider the messages belong to the same group. If the real interarrival time is larger than the S_{max}, then we believe there is a separate group. We set S_{min} to be 1 second (this is the finest time granularity available in the syslog data we used) and S_{max} to be 3 hours (this is based on domain knowledge).

3.3.2.2 Rule based Grouping

In the temporal grouping part, we only consider grouping messages with the same template together. Now we try to discover the connections among messages with different message templates. The offline-learned rules using association mining contain pair-wise message templates that occur frequently together. The rule based grouping component groups the messages which happen on the spatially matched locations and happen close enough in time (within the window W discussed in rule-learning part). Note that our rule based grouping does not consider the direction of rule since our system is not a troubleshooting system thus does not rely on causality inference. It is possible that we have $A \rightarrow B$ and $A \rightarrow C$ in the rule set, but we ignore the direction and can group A, B, C together, assuming temporal and spatial constraints are satisfied. This is because it is very likely they are triggered by the same network condition thus should be considered as one event, even though we ignore the detailed

causal relationship among the messages.

3.3.2.3 Cross Router Grouping

The first two grouping methods all focus on a single router. A network event, however, can affect multiple routers. For instance, a link down event should involve two adjacent routers' links. To group such messages, our solution is a conservative one. Our offline location learning component already provides a dictionary for cross-router location relationship such as links, sessions, tunnels (a path) between different routers. Assuming that the propagation along the connects are fast enough, we group messages with the same template which happened on the same link, session, or path at almost the same time (e.g no larger than 1 second difference).

We perform three grouping methods in the order they are described. If any two messages in two different groups have been grouped together, then these two groups will be merged. Thus the changes of orders of these three parts have no impact on the final grouping results. We use this order because it is more natural to describe: from one signature to multiple ones, from single router to multiples ones.

3.3.2.4 Prioritization and Presentation

We now have a number of messages in each group. We first prioritize the messages so that the most important events will appear at the top of the digest. Recall from Section 2.2 that the severity level of a message provided by syslog shall not be trusted/used. Instead, we use a combination of the following three metrics. The first metric is the occurrence frequency of message signature on each router, say f_m for message m. The intuition is that we care more about rare events. We also consider the impact of the events. The event happened on the higher level of the hierarchy is more important. For example, an event happened on the router is more important than the one happened on the interface. Let l_m denote the location metric of message m, and we can assume that the value of l_m higher level is several (e.g., 10) times of

lower level. Finally, we consider the number of messages in the grouped event, which in some sense reflects severity of the event. The group with more messages should be relatively important. Based on these three metrics, there is a score we assign for each event:

$$Score = \sum_{m=1}^{M} l_m / log(f_m)$$

where the event contains M messages. The reason for taking logarithm here is to prevent rare events with tiny f_m values from dominating the top of the ranked list. Note that our scoring method provides a baseline for ranking. The network operators can adjust the weights for each type of messages, based on their experience.

We rank all events based on the score in a decreasing order. After ranking, we (actually the SyslogDigest system) are ready to present the final result. There are many ways to display the event, and we choose the most concise way. First, we present the beginning and ending time of the event, which map to the time range of all messages in the group. Second, we present the location information of the event. For each router, we present the most common highest level location on hierarchy. For example, if the event contains two messages, one on the router level while the other on the interface level, we only show the router. Third, we present the type of event. One direct way is to present the combinations of message signatures within the group. Domain experts can certainly assign a name for each type of event. For example, we can assign a name "link flap" to a event which contains "LINK-DOWN" and "LINK-UP" messages.

3.4 Evaluation

We evaluate SyslogDigest using real syslog data from two large operational networks. We first validate several design choices we made in the offline domain knowledge learning component. Then we report the results of the entire SyslogDigest system.

3.4.1 Evaluation Methodology

Although our dissertation focuses on the IPTV network, our SyslogDigest can apply to any operational IP networks, since our methodology is router-vendor independent and network independent. For evaluation, we use syslog data collected from two large operational IP networks in North America: a tier-1 ISP backbone network and a nation-wide commercial IPTV backbone network. Each of these two networks has around a couple of thousands of routers and records millions of syslog messages per day. We refer to these two syslog data as *Dataset A* and *Dataset B*, respectively. Note that these two networks use routers from different vendors and have different network design and protocol configurations for supporting different network applications/services – the ISP backbone network is for general/traditional IP/Internet service and the IPTV backbone is for commercial TV service. Both the types of messages and their signatures are very different in these two dataset. In our evaluation, we use three months of data collected from September to November 2009 for offline domain knowledge learning and two weeks' of data collected December 1-14, 2009 for online processing and reporting event digests.

We evaluate the effectiveness of SyslogDigest in the following two aspects. First, we use the metrics *compression ratio* to measure the ability of SyslogDigest to reduce the amount of information that operators need to receive and examine in order to know what happened in the network for each incident. We define the the compression ratio to be the number of events (compressed information size) divided by the total number of raw messages (uncompressed information size). Second, we validate the event digests output by SyslogDigest to see whether any unrelevant messages are collected into one network event. This validation process done manually by people who have rich network experiences and domain knowledge on these two operational networks.

Table 6: Sensitivity of minimal support SP_{min} value

SP_{min}	Top % (A)	Coverage (A)	Top % (B)	Coverage (B)
0.001	13.4%	98.72%	14.2%	89.34%
0.0005	27.5%	99.92%	32.3%	99.95%
0.0001	42.5%	99.98%	54.3%	99.99%

3.4.2 Components Effectiveness

We first evaluate our design choices in message template identification, association rule mining and temporal mining components shown in Figure 2.

3.4.2.1 Message Template Identification

SyslogDigest automatically abstracts the template of each type of syslog messages. We validate our template abstraction method presented in Section 3.3.1 by comparing the syslog message templates identified by SyslogDigest with the "ground truth" templates obtained from hard-coding comprehensive domain knowledge on syslog. The domain knowledge we used here are very specific to certain router vendors. Hard-coding domain knowledge is clearly not scalable, and hence we use it only for validation purpose. Note that such kind of methods require the knowledge of all message format in advance, which is not practical specially when there are many messages types and facing different syslog data sources. We observe that 94% of message templates matches. It indicates our learning approach can extract the template fairly well.

3.4.2.2 Association Rule Mining

In order to generated rules, we use three months (Sep 2009 to Nov 2009) for mining for both datasets.

There are three parameters used by SyslogDigest in mining association rules between syslog messages: Window size W, the threshold of minimal support SP_{min}, and the minimal confidence $Conf_{min}$. We evaluate the sensitivity of these parameter

setting on learning associations between syslog message. In particular, we vary W from 5 to 300 seconds and $Conf_{min}$ from 0.5 to 0.9. We also set SP_{min} at values 0.001, 0.0005, 0.0001. The implication of these settings of SP_{min} in our context is shown in Table 6. For example, when $SP_{min} = 0.005$, the top 27.5% types of messages will be used in rule mining and these types of messages cover over 99.9% of all messages in dataset A.

Figure 7 shows the number of association rules learned from dataset with fixing the value of W to be 1 minute and varying the values of SP_{min} and $Conf_{min}$. As we expected, the number of rules decreases as the value of $Conf_{min}$ increases. In addition, the higher the value of SP_{min} is, the fewer rules learned from the dataset. Similar observations hold different values of W. In our experiments, we set $Conf_{min} = 0.8$ and $SP_{min} = 0.0005$. With this setting, Figure 8 shows the number of generated rules by varying the value of W. We observe that the number of rules increases as W increases. However, the increase in the number rules diminishes at $W = 120$ seconds for dataset A and $W = 40$ second for dataset B. That is, the number of rules learned by SyslogDigest is less sensitive to the value of W when W is large. A detailed analysis on the rules reveals that the newly added rules by increasing W often captures implicit timing relationship between two types of messages. For example, in dataset A, we observe that when W changes from 10 to 30 seconds, there are several new rules added to the knowledge base. These rules associate the *controller flap*, *link flap* and*line protocol flap* messages, indicating that these types of messages usually occur together between 10-30 apart. Similarly, in dataset B we find that *ftp login failure* and *ssh login failure* messages are associated together when W is set to 30 - 40 seconds. Next, we present results on association rule mining with using $W = 120$ seconds for dataset A and $W = 40$ seconds for dataset B.

The association rule mining is performed weekly by SyslogDigest to (i) add new rules to the knowledge base, and (ii) identify invalid rules in the knowledge base and

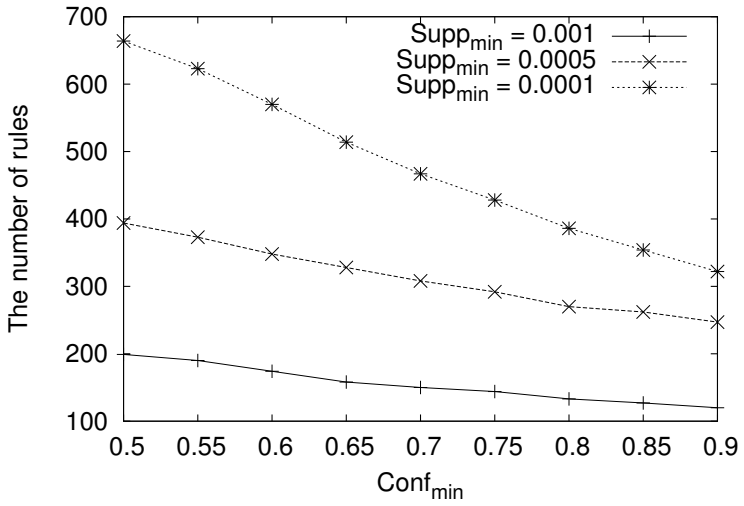


Figure 7: The impact of parameter SP_{min} and $Conf_{min}$, in dataset A.

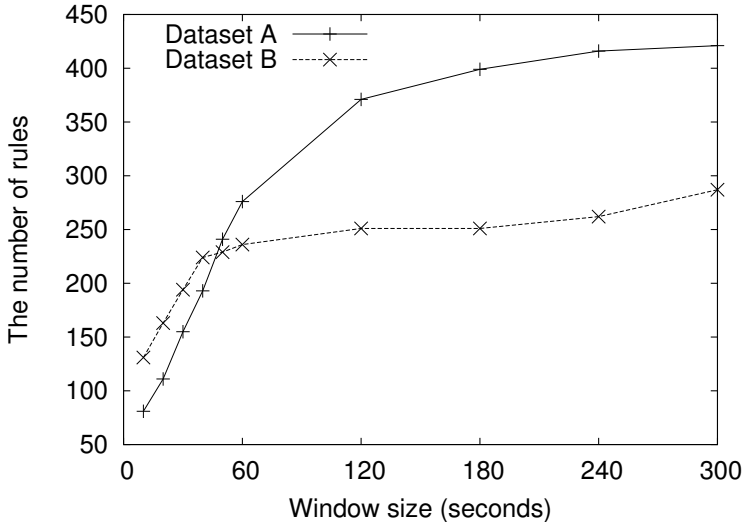


Figure 8: The impact of parameter W, when $Conf_{min} = 0.8$ and $SP_{min} = 0.0005$.

remove them using the method presented in Section 3.3.1.4. Figure 9 and Figure 10 show the total number of rules in the knowledge base as well as added/deleted rules for each week from week 2 to week 12. The number of rules in the knowledge base becomes stable after week 6 for dataset A and after week 8 for dataset B. This is because the number of added and deleted rules are close to zero after few weeks for both datasets.

We further validated the rule sets obtained at the end of week 12 with expert

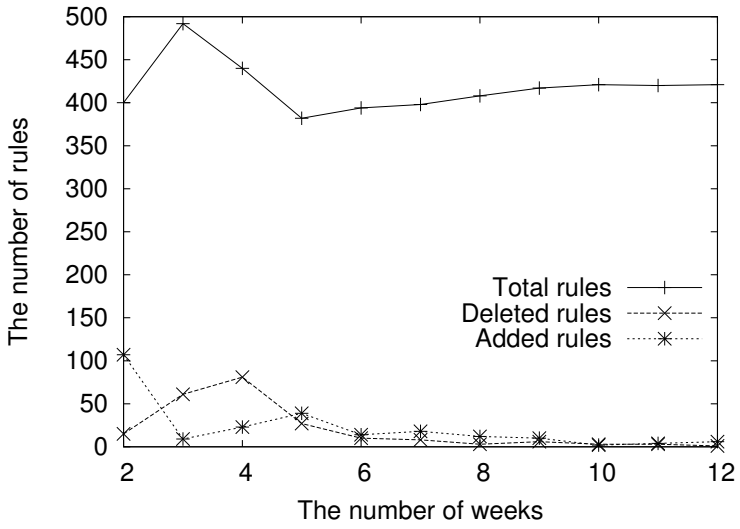


Figure 9: The number of rules over 12 weeks, dataset A.

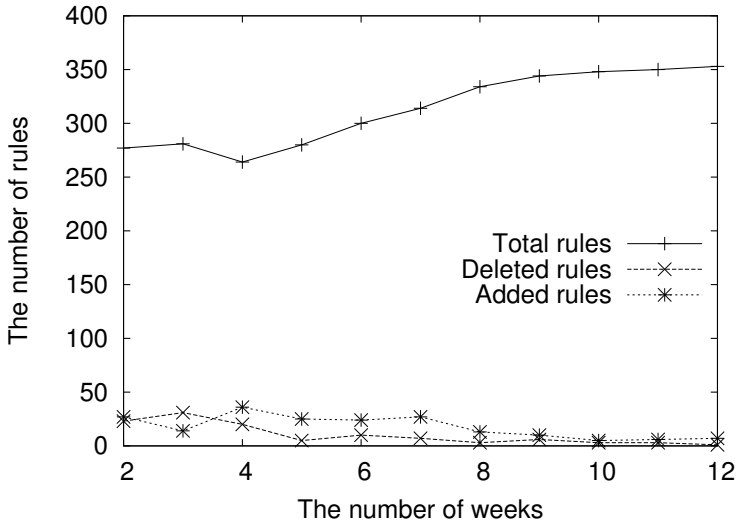


Figure 10: The number of rules over 12 weeks, dataset B.

domain knowledge and vendor documentations. We found that almost all the rules are consistent with either the domain knowledge or the expected behaviors specified in vendor documentations. Thus, we believe that SyslogDigest successfully captures network behaviors using automatically learned rules. However, we did report a few "unexpected" rules (3 rules for dataset A and 16 rules for dataset B), which means the potential false positive rate is less than 0.05 during rule mining. These unexpected rules are currently under investigation.

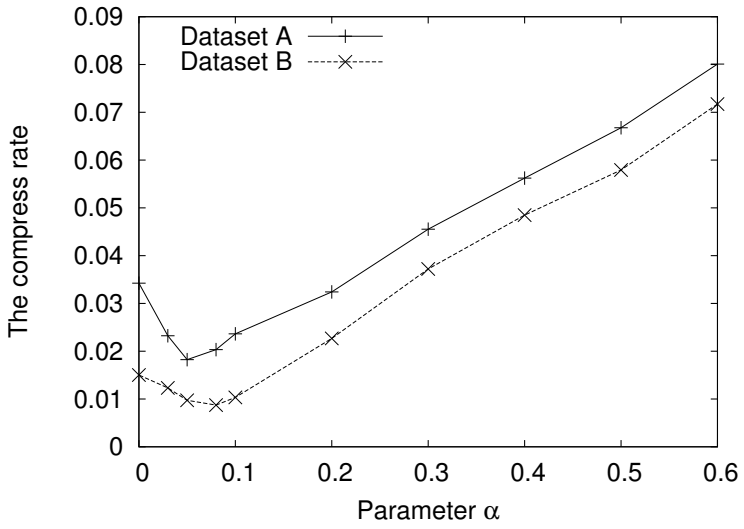


Figure 11: The impact of varying value of α on the compression ratio ($\beta = 2$).

3.4.2.3 Temporal Pattern Mining

The goal of temporal pattern mining is essentially to find the proper parameter α and β, such that the underlying interarrival model can present the temporal patterns very well. In other words, we want to find α and β such that we can group messages appropriately (i.e. compression ratio would be optimal). Figure 11 shows the compression ratio of temporal grouping with α varying from 0 to 0.6 and $\beta = 2$ (i.e., if a new message arrives at an interval that is at least twice of the predicted interval, the message is put in a separate group). We observe that in both datasets, the larger the value of α is the higher the compression ratio is, except for very small value of α (e.g., $\alpha < 0.05$). The lowest (i.e., best) compression ratio is achieved when $\alpha = 0.05$ for dataset A and $\alpha = 0.075$ for dataset B. They will be used as the default value for α in the remaining experiments.

Figure 12 shows the impact of varying value of β (from 2 to 7) on the compression ratio with α being set at the default values. We observe that the compression ratio first decreases as we increase the value of β. This is consist to our intuition because a larger β value means larger intervals are used in temporal grouping of messages and hence fewer number of groups are output. We also observe that the improvement of

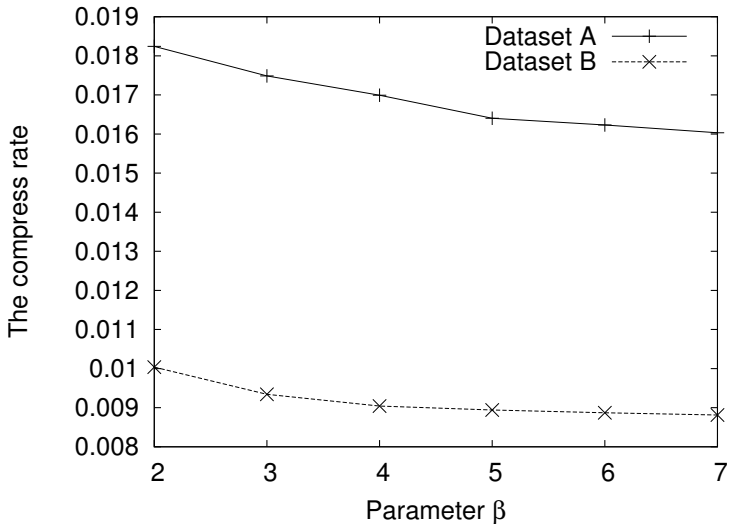


Figure 12: The impact of varying value of β on the compression ratio ($\alpha = 0.05$ for dataset A and $\alpha = 0.075$ for dataset B).

Table 7: Parameter setting in SyslogDigest

Dataset	α	β	W (Dataset A/B)	SP_{min}	$Conf_{min}$
A	0.05	5	120	0.0005	0.8
B	0.075	5	40	0.0005	0.8

compression diminishes when β increases. Thus, we set $\beta = 5$ for both datasets.

In summary, Table 7 shows the parameter settings in SyslogDigest. The rules of thumb are of choosing parameters are (1) to ensure the stability of rule sets and (2) to ensure the stability of the compress ratio, as we discussed through Section 5.2. These values in Table 7 will be used in the rest of our experiments presented in this paper.

3.4.3 Performance of SyslogDigest

Using the domain knowledge base built by applying offline learning on three months of syslog data, we run SyslogDigest in online mode and generate event digests for 2 weeks of syslog data to evaluate the effectiveness of the entire system. Note that it generally takes less than one hour to digest one day's syslog. Table 8 shows compression ratios of different message grouping methodologies for both datasets. We found that the

Table 8: Effectiveness (compression ratio) of three digest methodologies. T: temporal based, R: rule based, C: cross router

Methodology	Ratio (Dataset A)	Ratio (Dataset B)
T	1.63×10^{-2}	9.08×10^{-3}
T+R	5.15×10^{-3}	2.26×10^{-3}
T+R+C	3.27×10^{-3}	0.91×10^{-3}

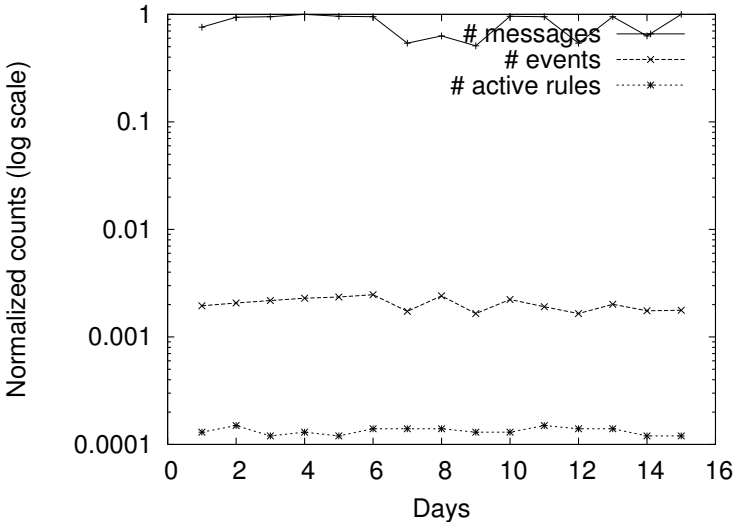


Figure 13: Number of event digests and active rules per day for dataset A.

compression ratio varies by grouping method by dataset. Overall, the number of event digests is over three orders of magnitude smaller than the number of raw syslog messages. This is fairly substantial information compression/reduction.

Figure 13 shows the number of events and number of messages per day during these two weeks for dataset A (the numbers are normalized by a fixed factor due to proprietary information). Again, we obverse the three orders of magnitude compression. In addition, the number of events per day is relatively stable across days for both datasets. In addition to the events digest, SyslogDigest also tracks the association rules that are used in message grouping (we call them "active rules"). The number of active rules is also stable: $100 \sim 200$ per day. The observations on dataset B is similar and omitted due to space limit.

Figure 14 show normalized number of raw messages and number of events on each

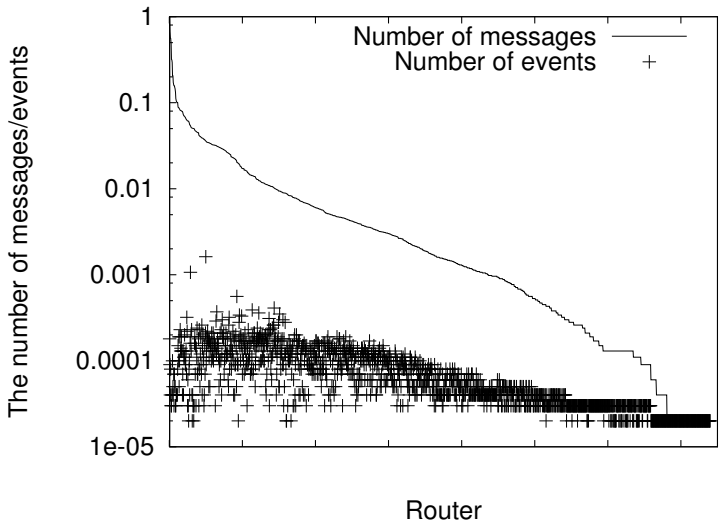


Figure 14: The digest result per router of dataset A.

individual router for datasets A[2]. We observe that the distribution of events across routers are less skew than that of raw syslog messages. In addition, routers that have more syslog messages usually have a better compression ratio. The best compression ratio is achieved on the router which has the largest number of raw message.

In order to verify that SyslogDigest system does not miss important network events during extraction, we compare the event digests output by SyslogDigest with known network events obtained from the trouble tickets. We obtained trouble tickets for both dataset A and dataset B, each of which is associated with a unique case identifier, timestamp of which the ticket is created and/or updated, and type and location of the event. While an extensive and systematic evaluation is undergoing, we show our preliminary results in this paper. In our preliminary evaluation, we rank the tickets based on the number of times a ticket is investigated and the corresponding record is updated. The intuition is that the more times that a ticket is investigated, the more likely the corresponding event is more important (and/or complicated). Hence, we use the number of times a ticket is investigated and updated as an approximation of the important of an event. We select the top 30 tickets regarding dataset B and

[2]Due to page limitation, very similar result for dataset B is not shown here

correlate them with event digests output by SyslogDigest. We say there is *match* between a trouble ticket and an event digest if (i) the duration of the event digest covers the creation time of the trouble ticket and (ii) the event location of specified in the event digest is consistent with that described in the trouble ticket (at the state level, e.g.,TX, GA, etc.). We found that all 30 tickets match with event digests that are ranked as top 5% or even higher by SyslogDigest. This initial evaluation shows that SyslogDigest does not miss important incidents.

3.5 Applications

In this section, we demonstrate that SyslogDigest can be used an essential building block for many critical applications in network operations, such as troubleshooting and network health monitoring and visualization.

3.5.1 Complex network troubleshooting

Router syslog is one of the most important data source for network troubleshooting, and SyslogDigest provides network operators the gist of the syslogs – high-level network events. This is very important especially for diagnosing complex events that involve protocol interactions across multiple network layers and locations.

We next examine a real-world example on PIM neighbor loss event in the IPTV network – an event that was identified by SyslogDigest and is intriguingly complex.

In the commercial IPTV network, live TV streams are delivered using native IP multicast (i.e., PIM in this example). A change or loss of PIM neighbor session (e.g., caused by link failures) can disrupt delivery of IPTV data streams. Hence, there are several mechanisms implemented in the layer 2 and layer 3 network to enhance the service reliability. Particularly, two static layer 2 paths are configured between each pair of routers on the multicast tree – the primary path is the single-hop one directly connecting these two routers and the secondary path is a multi-hop path through routers in different VHOs. When there is a physical link failure on the primary

path, the secondary path will be used to deliver IPTV data streams through the MPLS tunnels. The fast re-route (FRR) is done in layer 2 so that layer 3 routing (i.e., OSPF) is oblivious of this fail-over event, avoiding lengthy route-reconvergence impacting the PIM neighbor session. In this design, PIM neighbor session should only be impacted when there are dual failures on both primary path and secondary path. Such dual failures are extremely rare in operational networks.

In a troubleshooting task, operators investigated a PIM neighbor session flap event between a pair of nodes in two VHOs. The event was somewhat unexpected because the PIM neighbor session loss appeared to be triggered by a single link failure on the primary path between two routers. In theory, the PIM neighbor session should not be impacted. The event signature output by SyslogDigest revealed that the secondary path had not been set up successfully and was undergoing connection retries every five minutes. Consequently, the PIM neighbor session was immediately interrupted when the primary path failed. In the event signature, hundreds of syslog messages recorded on a dozen of routers in multiple VHOs are associated to this SyslogDigest event. These syslog messages are of 15 distinct error codes involving 6 network protocols across three network layers.

If without SyslogDigest, it would easily take operators hours to manually identify these messages among tens of thousands of syslogs recorded at these routers and close in time to the PIM loss event. As a matter of fact, it is not clear what time duration should the network operators focus on in searching for related syslogs, simply because different protocols operate (or react to network changes) at different time granularities (ranging from sub-seconds to minutes or longer). In this particular event, the syslog messages that indicates a failure in setting up the secondary path, i.e., the connection retries, are several minutes apart from the link failure. Without knowing the exact protocols and timer involved, it is difficult to find the right time window to focus on – a short window (e.g., $\pm$60 seconds) would risk missing the failure information of the

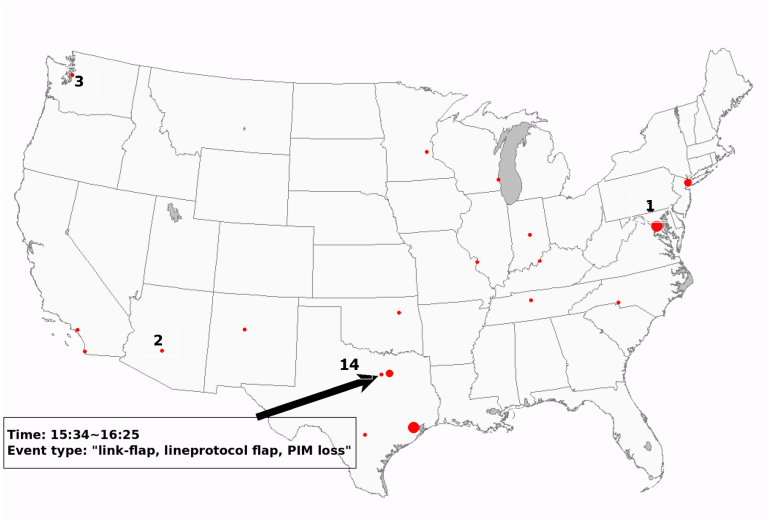


Figure 15: Visualization based on SyslogDigest output.

secondary path, while a long window (e.g., ±3600 seconds) would certainly increase the amount of syslogs to be analyzed, slowing down operators' investigation.

By contrast, SyslogDigest was able to uncover the complete stream of this complex event. This is because SyslogDigest "learns" both the types and the co-occurrence time patterns of related syslog messages, and consequently associate such syslogs together when they do co-occur.

Furthermore, even with other automated troubleshooting systems (e.g., [27]) in place, working with pre-processed high-level events can greatly improve the efficiency compared to working with large numbers of raw syslog events.

3.5.2 Network health monitoring and visualization

It is imperative for network operators to keep track of "what happened in my network?". Visualization is often an effective way to achieve this as operators are able to "see" what happened in the network and how things evolve over time. The digest events from SyslogDigest can greatly improve network health visualization.

Figures 15 and 16 show the snapshots of network status map at 2009/12/5 16:00:00 (in 10 minute updating window) using SyslogDigest and using raw syslog

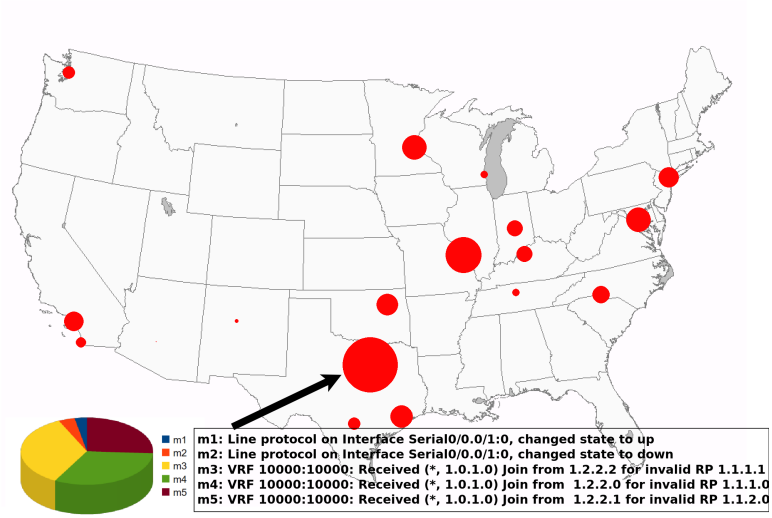


Figure 16: Visualization based on raw syslog data.

messages, respectively. Network topology and link load status are removed from the graph to protect proprietary information. The circles in the map indicate events (or messages) observed at these routers with larger circles indicating more events (or messages) observed. We observe that only a small number of events took place in the network then, while the corresponding syslog messages range from dozens to a couple of hundreds on each of these routers. Making sense of the raw syslogs visualization requires decoding plenty of supplement information (e.g., the pie chart shown in Figure 16 reporting the mixture of syslog types and counts for all events on the router). It is worth noting that high syslog message counts do not necessarily imply more network events or "bigger trouble" – the big circle in Figure 16 was one moderate level event compare to others in Figure 15. Visualization the raw syslog messages can potentially mislead operators to focus on routers with more messages and delay their investigation on more sever issues.

3.6 Related Work

Commercial softwares, like NetCool [2] Lonix [1], that are capable of parsing and making log data. These tools, however, require intensive domain knowledge to describe the format of logs. Xu *et al.* propose a general methodology to mine the console logs and to automatically detect system running time problem [55]. But they assume that they have the access to source code which generate the logs. It is not a practical assumption in the environment of router syslogs.

Troubleshooting network problems is one of the most important management tasks. Many approaches have been proposed recent years [31, 30, 46, 25, 36, 35, 29]. The general idea is to apply advanced statistical methodologies to multiple raw data sources. Our system is not specifically designed for troubleshooting, but as illustrated in Section 3.5, it can benefit complex troubleshooting task significantly.

Rule learning has been widely applied in acquiring insight on different network problems. Kandula *et al.* [28] mine the rules in edge networks based on the network traffic data. Brauckhoff *et al.* [9] use association rule mining techniques to extract anomaly in backbone network. Their data sources, detailed mining methodologies and utilities of final mining results are different from our system. Yamanishi *et al.* [56] provides a dynamic syslog mining technique on server syslogs. They focus on generating predicative alarm for system failures. Our goal is more broad – to represent the network events. Moreover, their syslog data are essentially logs on the end host devices, rather than router syslogs.

The idea of extracting high level information from raw data has been used in network traffic analysis. A few tools aggregate traffic volumes and visualize the resulting aggregates. For example, FlowScan [41] analyzes and reports on Internet Protocol (IP) flow data exported by routers, based on the application, the IP prefix, or the AS identifier. eXpose [28] identifies temporally correlated clusters of flows. NICE [36] is a correlation tool focusing on the chronic network conditions. In comparison, our

system is the first one used for extracting network events from logs.

3.7 Summary and Future Work

In this chapter, we develop a system called SyslogDigest that groups massive volume of syslog messages into small number of meaningful network events using data mining techniques. SyslogDigest systematically identifies signatures of syslog messages, learns association rules that capture network behaviors over time, groups related raw syslog messages across multiple routers into network events, and labels and prioritizes network events appropriately. We evaluated SyslogDigest using real syslog data collected from two large operational networks and demonstrated how SyslogDigest can be applied on complex network troubleshooting and network health monitoring and visualization. Though we focused on syslog data, our techniques can also be applied on other passive measurement data. We believe SyslogDigest will be an essential building block for many network management tools. Applying SyslogDigest on other data and integrating it into various network management tools are among our future work.

CHAPTER IV

MODELING IPTV CHANNEL POPULARITY

4.1 Introduction

Understanding the channel popularity or content popularity is an important step in the workload characterization for modern information distribution systems such as World Wide Web, P2P file-sharing systems, IPTV networks, video-on-demand (VOD) systems, content distribution networks, publish/subscribe systems, and RSS feeds distribution systems. The proper modeling of the distribution of user's interest in various contents and media in the system is a key building block for system design and performance analysis. For example, it has been well known that web site popularity is highly skewed and can be characterized by a Zipf-like distribution [38], a factor that carries important implication in evaluating different DNS caching policies. Similar popularity skewness has also been observed in other systems including P2P file-sharing [15], VOD [60], web servers [6], and IPTV [12].

Another important aspect of the content or channel popularity is its temporal dynamics, which captures the popularity changes over time. Examples of such dynamics are the shift of users' search and download interest among files in a P2P file-sharing system, the change of subscriber numbers among different topics in a publish/subscribe system, and the growth/shrink of community groups in a social network. The popularity dynamics can be either attributed to the stochastic nature of users' interest at the time, or attributed to the change of active users' population at the time, or a combination of both. Understanding the process of popularity dynamics can provide important insight into service design and optimization. For example, properties on TV channel popularity dynamics are an essential piece of information

for evaluating the proposal of using peer-assisted TV stream distribution (e.g., [24]) in an IPTV system.

In this chapter, we focus on analyzing the channel popularity in the context of Internet Protocol Television (IPTV). Our goal is to construct mathematical models to capture the distribution and the time dynamics of channel popularity. This is motivated by recent booming growth among telecommunication companies around the world in the rapid deployment of the IPTV infrastructure and service expansion, and hence the increasing demand in the workload characterization and performance evaluation of the IPTV system. However, we believe that the basic principle and methodology used herein are applicable to other domains (e.g., RSS feeds, news groups).

Our analysis is based on a large collection of user channel access data from a nation-wide commercial IPTV network[1]. We conduct an in-depth analysis of the user channel switch activities and study the channel popularity for different channels, at different time and different aggregation scales (ranging from minutes to days). Then, we identify a stochastic model that matches well in all attributes of interest with respect to the channel popularity. We also explore subsets of user population and investigate whether they intrinsically have different channel preferences from others. We then construct multi-class population models that capture the non-stationary behavior of channel popularity exhibited by its diurnal patterns, which has been reported in previous measurement study [12].

Our contributions can be summarized as follows:

- We observe that channel popularity is highly skewed and can be well captured by a Zipf-like distribution. This holds true at different times of day and at various aggregation scales. We find that the popularity of each individual channel

[1]To protect the identity of the IPTV network subscribers, individual set top boxes were assigned a non-identifiable ID number for purposes of this research. The authors did not have access to subscriber's identity or address of individual set top boxes.

has an exponentially decaying autocorrelation function, a common behavior across different channels. We also examine the change of two channel popularity vectors at adjacent time bins while varying the aggregation step. We find that the cosine similarity between channel popularity vectors exhibits an interesting multi-scale behavior, forming a *V*-shape when the aggregation scale increases from minutes to days.

- We model channel popularity dynamics as an Ornstein-Uhlenbeck process and find that it matches remarkably well with respect to the above properties. The success in capturing the underlying channel popularity dynamics enables our model to produce a satisfying result for channel popularity prediction.

- We develop a method to identify subsets of user population with inherently different channel interests. We apply the K-means clustering algorithm on various features of users, and use a symmetric uncertainty measure and hypothesis test to evaluate the significance of channel popularity difference. By tracking the change of population mixtures among different user classes, we extend our model to a multi-class population model, which enables us to capture the moderate diurnal popularity patterns exhibited in some channels.

4.2 Measurement Results of Channel Popularity

Channel popularity needs to be precisely defined before we can present our analysis result. There are two common used metrics to measure the channel popularity. The first is based on *channel access frequency* which is defined as the number of channel switching requests to the channel. The other is based on *channel dwell time* defined as the amount of time STBs stay tuned in the channel. They measure two different aspects of channel popularity: weighted by visit frequency vs. weighted by watching time.

Figure 17 and Figure 18 show the time series of the number of online STBs and

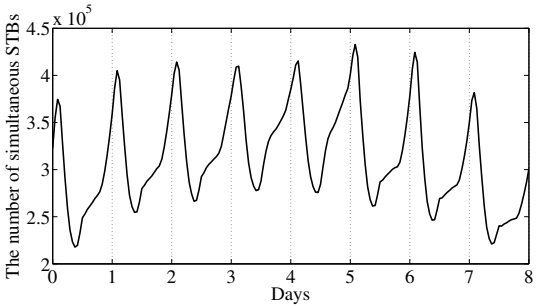


Figure 17: The number of online STBs for each hour during a week.

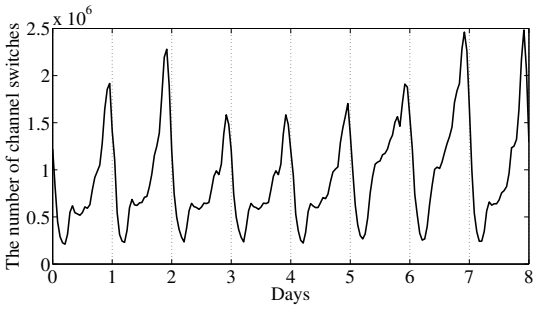


Figure 18: The number of channel switches for each hour during a week.

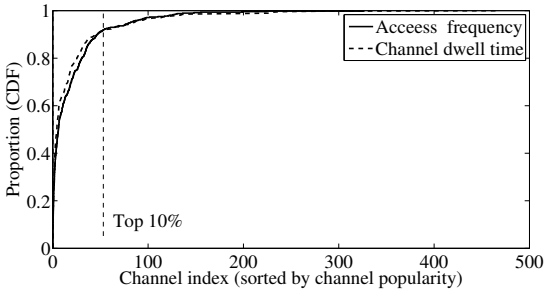
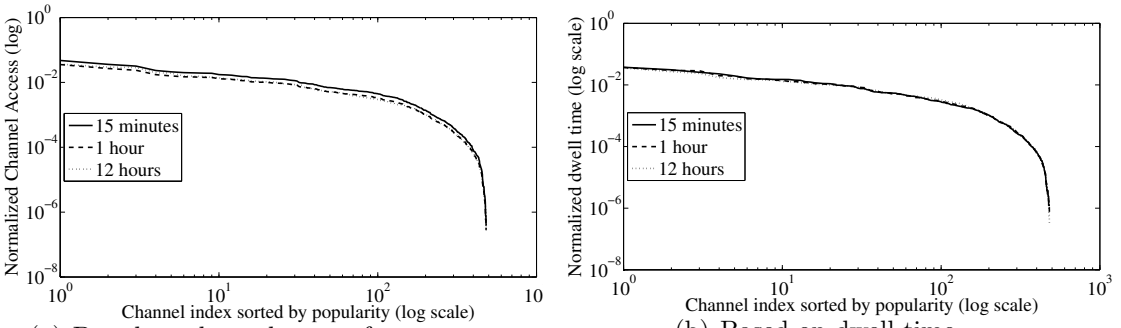


Figure 19: CDF of channel popularity.

the total number of channel switches respectively. As expected, we find both of them highly variable, exhibiting strong diurnal patterns. To account for the variation due to the change in the number of active users, we focus on the probability distribution (i.e., normalized among all channels) instead of the absolute value of the channel popularity measure.

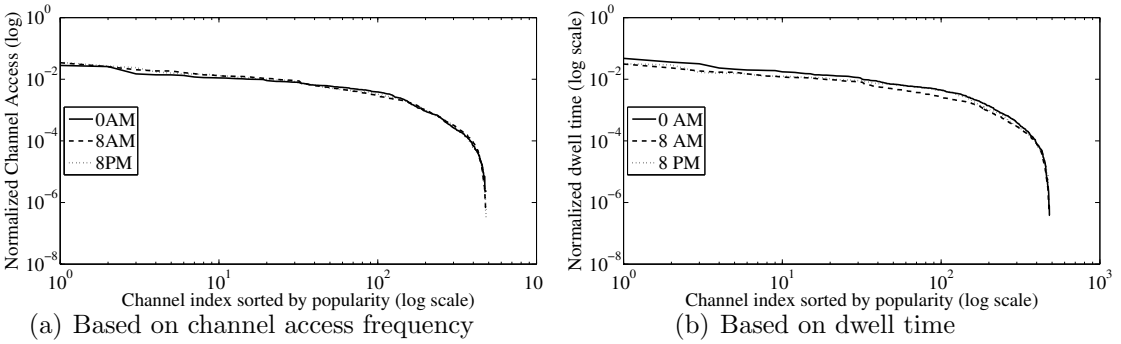
4.2.1 Distribution of Channel Popularity

We first examine the long term distribution of channel popularity of all channels using both metrics. Figure 19 shows the cumulative distribution function (CDF) of channel popularity ranked by access frequency and dwell time. We observe a close match between the CDF curves of the two different popularity metrics. Both distribution functions exhibit high skewness – the top 10% of channels account for more than 90% of channel access.



(a) Based on channel access frequency (b) Based on dwell time

Figure 20: Channel popularity distribution (varying time period).



(a) Based on channel access frequency (b) Based on dwell time

Figure 21: Channel popularity distribution (varying start time).

We next focus on the short term distribution of channel popularity with respect to the two metrics. We examine this property at different time scales and at different points in time. Interestingly, we find the channel popularity distribution is nearly *invariant* across a large range of measurement time scales or over different time periods.

For example, in Figure 20, we show the average popularity probabilities (i.e., normalized channel access ratio and normalized channel dwell time ratio) measured during 15-minutes, 1-hour, and 12-hour periods starting from 0 AM. We sort the channels in a decreasing order of popularity and plot their rank in x-axis. We find that the three curves almost overlap on top of each other. In Figure 21, we also show the channel popularity probabilities of the same day using 4-hour aggregation granularity (0 AM to 4 AM, 8 AM to 12 PM, and 8 PM to 12 AM). Again, we find the curves very close to each other. We emphasize that the nearly invariant distribution function does not necessarily imply the channel popularity itself being invariant — the rank order of the channels is actually different from time to time and from one scale to another. We will turn to the temporal dynamics of channel popularity in Section 4.2.3.

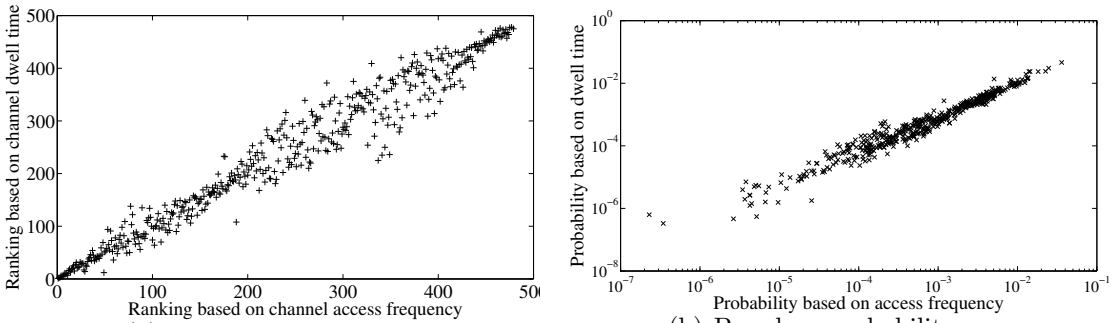
The log-log scale curves in Figure 20 and Figure 21 also suggest that channel popularity is highly skewed in all cases. To simplify computation, in the rest of the chapter, we focus only on the top 150 channels which account for over 98% of the channel accesses. We acknowledge that modeling the tail part may be important for some applications. However, this simplification should have little impact on the analysis of overall time dynamics of channel popularity, which is the main focus of our study.

4.2.2 Correlation between Channel Accesses and Channel Dwell Time

We have observed in the previous subsection that channel popularity based on access frequency and based on dwell time produces a very similar result. This may be an indication for a strong correlation between these two popularity measures, which turns out to be true as illustrated in Figure 22, based on the entire period of trace. Figure 22(a) shows the scatter plot of the ranks of the channels in which the popularity rank according to channel access frequency is shown on the x-axis and the rank according to channel dwell time on the y-axis. Figure 22(b) shows the similar scatter

plot of the actual probability value by the two metrics instead of the corresponding ranks. In both figures, we find that the points are spread well along the diagonal line — their Spearman rank correlation value equals to 0.98 and their Peterson correlation coefficient equals to 0.97 – demonstrating the strong correlation between the two popularity metrics. We believe that the relatively long delay during channel switches and the convenient TV guide and favorite-channel-list features are both contributing factors to this high correlation, as people are more likely to switch directly to the channel that they intend to watch.

In the rest of this chapter, we use the channel access frequency as the metric for channel popularity when illustrating our findings.



(a) Based on ranking (b) Based on probability

Figure 22: The correlation between channel access frequency and dwell time.

4.2.3 Temporal Dynamics of Channel Popularity

We now turn our attention to the time dynamics of popularity for individual channels. We start by looking at the time series of the channel popularity. (We refer to channel popularity measured by channel access frequency simply as channel popularity). Figures 23 and 24 show the time series of 9 days for a kids channel K and a news channel N respectively. In contrast to the time-invariant behavior reported in Section 4.2.1, both time series exhibit strong fluctuations over time. We next follow classic time series analysis processes to analyze these channel popularity series. In particular, we examine their stationarity, their first-order and second-order statistics,

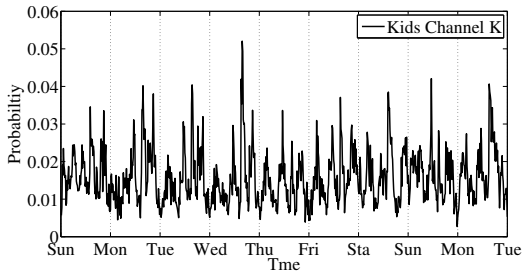


Figure 23: The dynamics of channel popularity of kids channel K, 1 point every 15 minutes

and their autocorrelation structure.

To test the stationarity of the channel popularity series, we apply the nonparametric *runs test* [8]. Given a time series $X(t)$, the runs test works as follows: (i) divide the series into equal-length time intervals and compute a mean value $\bar{X}_i$ for each bin, (ii) compute the median value of $\bar{X}_i$ over all bins and mark the ones below the median as "−" and the rest as "+", (iii) consider a consecutive sequence of "+" or a consecutive sequence of "−" as a *run* and count the total number of runs, and (iv) compare the number of runs against known run-count-distribution for stationary random data.

At the 95-th percentile confidence interval, we find that 92% of the channels pass the stationarity test when aggregated at 15-minute intervals. A small number of channels that fail the runs test exhibit non-trivial daily pattern, to which we will offer explanation in Section 4.4. .

We also calculate the coefficient of variation (CoV) for the channel popularity series. Figure 25 shows the distribution of CoV's of the channels. Despite the wide difference in their mean value (shown in Figure 19), we find that the CoV's of channel popularity series are narrowly centered around 0.6. For example, the CoV for the series of the kids channel K (in Figure 23) is 0.57 and that for the news channel N (in Figure 24) is 0.68. We will see how the empirical CoV help in our model in Section 4.3.

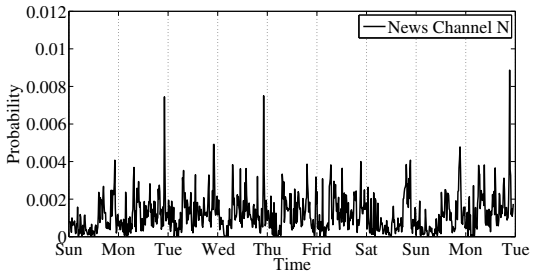


Figure 24: The dynamics of channel popularity of news channel N, 1 point every 15 minutes

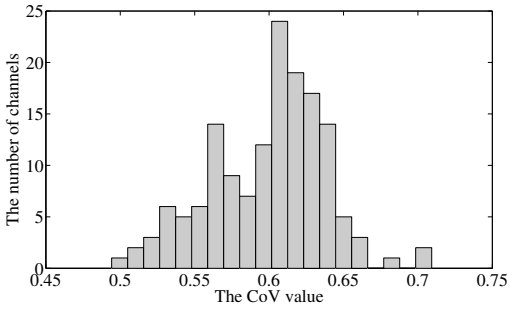


Figure 25: The distribution of CoV

We further study the autocorrelation structure of the channel popularity series, defined by their autocorrelation function (ACF):

$$R(\tau) = \frac{\mathrm{E}[(X_t - \mu)(X_{t+\tau} - \mu)]}{\sigma^2}$$

We compute the ACF for each channel, when the lag ranges from 15 minutes to 8 days. The autocorrelation value decays exponentially as the time lag increases (although some increases at the daily boundary). It is a typical behavior often observed in auto-regression processes. We further observe that the slope of the decreasing curves, which is the exponent of the exponentially decreasing ACF, are close among all channels. Using least square fitting, we obtain the best estimate of the exponent for each channel and plot their distribution in Figure 26. We find that their values concentrate at around -0.12.

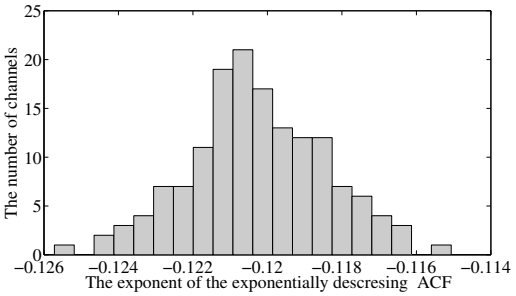


Figure 26: The distribution of the slope in ACF

4.2.4 Multi-scale Property of Channel Popularity Similarity

We have examined the autocorrelation structure of the popularity measure of each individual channel. To quantify the similarity (or dissimilarity) of the channel popularity collectively among all channels, we adopt the metric named *cosine similarity*. Cosine similarity measures the similarity between two vectors by finding the cosine value of the angle between them. For a pair of vectors $\mathbf{A}$ and $\mathbf{B}$, the cosine similarity is given by:

$$similarity(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}||\ ||\mathbf{B}||}$$

Its value ranges from -1 to 1, with value closer to 1 indicating higher similarity between $\mathbf{A}$ and $\mathbf{B}$. Cosine similarity has been widely used in high-dimensional data analysis such as applications in text mining [58].

In the context of IPTV, we expect the channel popularity to be relatively stable over time. This is indeed true—the average cosine similarity between adjacent 15-minute time bins is around 0.97, indicating the distribution of the channel popularity is quite stable in a short time frame. We further investigate whether the similarity becomes more pronounced with different time scales and perform the following multi-scale analysis.

We discretize our data traces by fixed-interval time bins, with interval lengths ranging from 15 minutes to 3 days. At each interval, we calculate the channel access probability of different channels for each time bin. Then, for each pair of adjacent

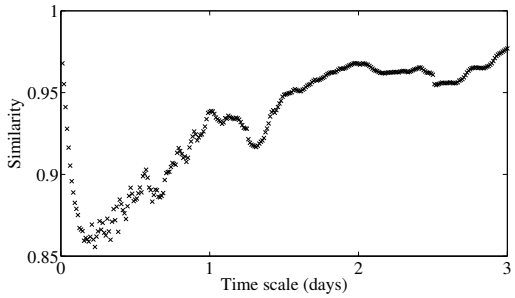


Figure 27: The average cosine similarity for different aggregation time scales

time bins, we compute the cosine similarity of channel popularity vectors. Based on these values, we calculate the average for each aggregation interval. Figure 27 shows the result where x-axis is the aggregation time scale (interval length) and y-axis is the average similarity.

In Figure 27, we observe that the curve forms a V-shape as we increase the aggregation level. Specifically, the similarity value first decreases as the aggregation times increases, reaching its minimum at around 3-4 hour aggregation scale. After that, we observe an increasing trend as we increase the aggregation time scale. This is because when the time scale is short, the similarity/dissimilarity of the channel popularity is determined by the TV program (shows) of the time. On the other hand, when the time scale is long, the similarity/dissimilarity is determined by the overall type of TV program on the channels. Both the viewer base of individual TV shows and the long term user affinity to the type of program are relatively more stable, which makes the time scale in between the weakest in term of channel popularity stability. We can also gain some intuition from the perspective of process analysis. Specifically, exponentially decaying autocorrelation function of channel popularity (note the log-scale on y-axis) causes the fast decreasing stability in short time scales. As the aggregation level becomes sufficiently large, the short term disturbances are smoothed out, converging to long term average, and hence improving the stability. We next present our model and demonstrate that our proposed model can closely match this behavior.

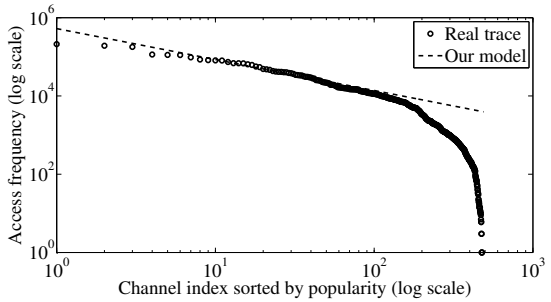


Figure 28: Channel popularity distribution.

4.3 Modeling Channel Popularity

We now present our model for channel popularity. We will use Zipf-like model to capture the long term channel popularity distribution among different channels and mean reversion model to capture the stochastic process of popularity fluctuation for individual channels.

4.3.1 Zipf-like Model

The Zipf-like distribution has been proved successful in capturing the skewness in content popularity such as Web [6] and VOD [45]. In the Zipf-like distribution, an object of the rank i has the access probability of $C/i^{1-\alpha}$, where C is a normalization constant and α is the distribution skew parameter. In Section 4.2, we have observed that the channel popularity is also highly skewed. We naturally model it using Zipf-like distribution.

Figure 45 shows the access frequencies of all channels in the order of decreasing popularity from the real trace and the fitted Zipf-like distribution (the dashed line with $\alpha = 0.55$). We observe a very good match up to around 150 channels, which account for over 98% of the channel-switches (see Figure 19).

4.3.2 Mean Reversion Model

Modeling the temporal dynamics of channel popularity is more challenging. Based on our analysis in Section 4.2, we choose a class of stocastic models, namely *mean*

reversion model for this purpose. Mean reversion model has been widely used in financial data analysis. The basic idea is that the price of a stock or a commodity may fluctuate but will revert to its long-term equilibrium level. Ornstein-Uhlenbeck (OU) process $\{X_t : t > 0\}$ is the most widely used mean reverting stochastic process in financial modeling [47]. It is stationary, Gaussian, Markovian, and continuous in probability. The Ornstein-Uhlenbeck process is characterized by the following linear stochastic differential equation (SDE) [17]:

$$\mathrm{d}X_t = \lambda(\mu - X_t)\,\mathrm{d}t + \sigma\,\mathrm{d}W_t, \tag{1}$$

where $\lambda > 0$ is the mean reversion rate, μ the long-term mean, and σ the volatility. W_t denotes a Wiener process (also known as Brownian motion), which is characterized by: (i) $W_0 = 0$, (ii) W_t is almost surely (i.e., with probability one) continuous, and (iii) W_t has independent increments with distribution $W_t - W_s \sim \mathcal{N}(0, t - s)$ for $0 \leq s < t$.

To understand the OU process, we can view the RHS of Eq (1) as summation of a deterministic term (the first term in RHS) and a stochastic term (the second term in RHS). When $X_t > \mu$, the deterministic term $\lambda(\mu - X_t)$ is negative, resulting in pulling back down toward the equilibrium level (i.e., μ); if $X_t < \mu$, the deterministic term is positive, pushing X_t back up to the equilibrium level. As a result, every time the stochastic term makes X_t deviate from the equilibrium, the deterministic term will act in such a way that X_t will head back to the equilibrium μ.

For an OU process, we have the moments:

$$\mathrm{E}(X) = \mu \tag{2}$$

$$Cov(X_s, X_t) = \frac{\sigma^2}{2\lambda} e^{-\lambda|s-t|} \tag{3}$$

This implies that the autocorrelation function of an OU process decays exponentially as the lag $|s-t|$ increases, which would match well with the empirical ACF of channel popularity series.

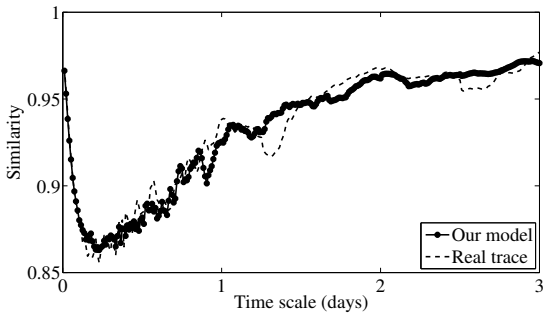


Figure 29: Cosine similarity using a simulated trace based on the mean reversion process.

We now determine the model parameters from the analysis result in the previous section. It is straightforward to see that the long term equilibrium μ can be derived from Eq (2), which we further model by the Zipf-like distribution. From Eq (3), we find that the autocorrelation decreases with lag at the rate $e^{-\lambda}$. Use the value extracted from in Figure 26, we set $\lambda = 0.12$. Finally, to determine σ, we use the coefficient of variation (CoV). Using Eq (3), we can derive σ as follows:

$$\sigma = \mu \times \sqrt{2\lambda} \times \text{CoV}$$

Using fixed time steps of 1, we can obtain a discrete version of the OU process and derive a first-order autoregressive sequence of X_t:

$$X_{i+1} = X_i e^{-\lambda} + \mu(1 - e^{-\lambda}) + \sigma\sqrt{\frac{1 - e^{-2\lambda}}{2\lambda}}\mathcal{N}_{0,1} \tag{4}$$

where $\mathcal{N}_{0,1}$ is a standard Gaussian random variable. This can be used to drive simulation of IPTV channel popularity.

We validate our model against measurement data. Recall that we observed multi-scale property as shown in Figure 27. Figure 29 shows that our model faithfully reproduce the V-shape behavior in the cosine similarity of channel popularity vectors.

4.3.3 Forecasting Channel Popularity

We have shown that with properly chosen parameters, the OU process can nicely capture various properties on the channel popularity. We now explore whether we

can use it as the underlying process to perform forecasting. More specifically, given the historical states from X_0 to X_i for a channel, how accurately can we predict X_{i+1}?

This can be viewed as a linear regression problem due to the AR(1) model of the sequence of X_i in Eq (4). To facilitate the regression analysis, we rewrite Eq (4) as:

$$X_{i+1} = aX_i + b + \epsilon \tag{5}$$

The first objective of linear regression analysis is to best-fit the data by estimating the parameters of the model. Of the different criteria that can be used to define what constitutes a best fit, the least squares criterion is a very powerful one. Using the least squares criterion, we obtain the model parameters as follows.

$$a = \frac{nX_{xy} - X_x X_y}{nX_{xx} - X_x^2}$$
$$b = \frac{X_y - aX_x}{n}$$
$$sd(\epsilon) = \sqrt{\frac{nX_{yy} - X_y^2 - a\left(nX_{xy} - X_x X_y\right)}{n(n-2)}}$$

where

$$X_x = \sum_{i=1}^{n} X_{i-1}, \qquad X_y = \sum_{i=1}^{n} X_i,$$
$$X_{xx} = \sum_{i=1}^{n} X_{i-1}^2, \qquad X_{xy} = \sum_{i=1}^{n} X_{i-1}X_i, \qquad X_{yy} = \sum_{i=1}^{n} X_i^2$$

We take the trace of a news channel to evaluate the performance of our forecasting model. We find that a small resulting mean squared error (MSE) ($= 8 \times 10^{-8}$) is obtained compared to its mean value 0.0014 and variance 9.3×10^{-7}. This means our forecasting model predicts the dynamics of channel popularity reasonably well. We have performed the prediction on various channels and observed the similar results.

4.4 Multi-class Popularity Modeling

So far we have presented our analysis on the dynamics of channel popularity and developed a mean reversion process to model them.

In this section, to enhance our model, we investigate whether we can identify sub-groups of STBs that have distinct channel preference, compared to the overall pattern and dynamics of channel popularity. We first explore various features that we can use to group STBs (Section 4.4.1), and then analyze properties of different groupings to identify a desirable grouping that we can use for our multi-class modeling (Section 4.4.2 and 4.4.3). This grouping method actually provides an interesting insight behind the dynamics of channel popularity (Section 4.4.4), and we employ this finding to develop a multi-class popularity model that better captures the channel popularity dynamics (Section 4.4.5).

4.4.1 Grouping STBs

Given the data set we have, we have a number of different ways to group the appearing STBs. Here we choose the following attributes which can best characterize a STB for grouping:

TV watching time: For each STB, we consider various aspects of TV watching time, such as daily average, hourly average, and average nightly watching time.

Channel change frequency: We consider the daily average and hourly average of channel changes to group STBs.

Dwell time per channel change: For each channel change, we determine how long a STB stays on the channel. This dwell time can be reported long when a user does not watch the channel, but leaves the STB on. To minimize such effect in our analysis, we investigate both the median value and the average value of dwell time per channel change.

Location: We use the network location of a STB to group STBs.

We use the first 15 days of the logs to calculate the attributes for each STB. In other words, we use the data in these days as the training set. As described later in

this section, we use the remaining data to evaluate the properties of grouping. We next describe different grouping strategies that we use to identify various groupings of STBs, which can be classified into two categories. One is threshold-based grouping. The other is clustering algorithm-based group.

4.4.1.1 Threshold-based

In this grouping, we select a grouping attribute and a set of corresponding thresholds to group STBs. These thresholds are chosen by the common sense of viewers instead of some specific computer algorithm.

Daily watching time (WT-D): We consider the daily average TV watching time for each STB. Specifically, we call a STB a *heavy-watcher* if the STB spends more than 12 hours on average, and a *light-watcher* if it spends less than 1 hour. We call the remaining STBs *medium-watchers*. In our data, 28% of STBs are heavy-watchers, and 36% of them are light-watchers. In the rest of this section, we call this grouping outcome WT-D.

Daytime vs. Nighttime (DN-D): We define a STB as a *daytime-watcher* if the average TV watching time during the day (from 6am to 6pm) is more than twice the time during the night (from 6pm to 6am). We define a *nighttime-watcher* similarly. We call the remaining STBs *all-time-watchers*. We observe 31% of STBs are daytime-watchers, and 39% of them are nighttime-watchers.

Daily channel change count (CHG-D): We use the average channel change count per day. We define the STBs that switch channels more than 200 times on average as *frequent-switchers* (24% of STBs), and the STBs that switch the channel less than 10 times as *infrequent-switchers* (12% of all). We call the remaining 64% of STBs *moderate-switchers*.

Median dwell time (DWL): For each STB, we use the log of 15 days and find the median value for the dwell time per channel change. Then we use 10, 20, and 30

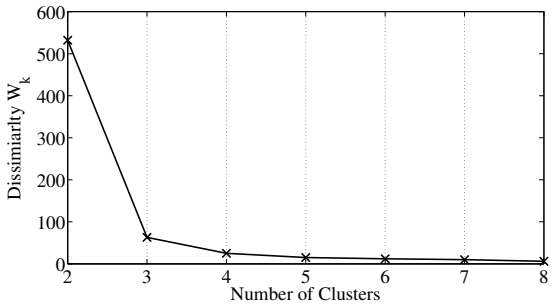


Figure 30: Dissimilarity vs. K when we use TV watching time as grouping feature.

minutes as thresholds to divide them into four groups.

Location (LC): We use the metropolitan area as the granularity of grouping STBs based on the location.

4.4.1.2 Clustering Algorithm-based

In this category, we employ unsupervised clustering algorithms to group STBs. While we have explored multiple clustering algorithms, we focus on the results based on the well-known *K-mean* algorithm [34], which is effective for large data sets. In this algorithm, we need to provide the number groups K as input parameter. While there are several ways to find the optimal K, we use the intra-cluster dissimilarity W_K as the measure:

$$W_K = \sum_{k=1}^{K} \sum_{C(i)=k} ||x_i - \hat{x}_k||^2,$$

where x_i is the data item, and $\hat{x}_k$ is the center of items in k-th cluster. We vary $K \in \{1, 2, \ldots, K_{max}\}$ and obtain a separate grouping result and the corresponding W_K for each K. Then, we consider the trade-off between dissimilarity and the number of clusters when we choose the optimal K. We present a concrete example below when we use the following set of attributes as input feature to the clustering algorithm. Different from the threshold-based category, they are all feature vectors.

Hourly TV watching time (WT-H): We assign a 24-element tuple to each STB, where each value corresponds to the average TV watching time per hour in a day.

Then we perform the K-mean algorithm to cluster the STBs. In a sense, WT-H simultaneously considers the two features used in WT-D and DN-D. In Figure 30, we plot W_K as a function of K. We can observe that as small a value as $K = 3$ provides a good grouping result. These three clusters cover 60%, 27%, and 13% of STBs, respectively.

Hourly channel change (CHG-H): Similar to WT-H, we collect the number of channel changes for each hour in a day and assign a 24-element vector to each STB. In this grouping, $K = 4$ leads to the optimal grouping result, where the clusters have 47%, 25%, 21%, and 7% of STBs, respectively.

Hourly dwell time (DWL-H): For each hour, we calculate the average dwell time per channel change. Then, we assign 24-element vector to each STB, to obtain four groups with 37%, 31%, 24%, and 8% of STBs from the K-mean algorithm, respectively.

Hourly median dwell time (MDWL-H): Unlike DWL-H, we use 1-hour intervals and calculate the median dwell time value for each 1-hour bin and input them into the K-mean algorithm. From this grouping, we obtain four groups with 41%, 20%, 10%, and 29% of STBs, respectively.

Channel preference (PREF): For each STB, we calculate the access probability to each of top 150 channels (which covers 98% of channel popularity as shown in Figure 19). Then, using Table 9[2], we classify these channels based on their program contents and obtain aggregate access probabilities for 8 types, which we use as the grouping attribute. We use $K = 8$ after the number of types.

In the rest of this section, we investigate whether we observe any correlation among the features and corresponding groupings and we can explain underlying channel popularity dynamics based on the identified sub-groups.

[2]In this classification, both educational channels and documentary channels belong to "science." The category "others" includes channels that offer diverse programs (e.g., news, TV series, shows, etc.) as well as less known channels that are not easy to classify.

Table 9: Classification of top 150 IPTV channels

Type	Examples	# channels
News	CNN, NBC News	13
Kids	Disney, Cartoon Network	15
Sports	ESPN, Star games, NBA TV	20
Movies	HBO, Cinemax	15
Science	Discovery channel, Animal planet	20
Music	MCM, MTV	21
Foreign	TF1, BFM, Al jazeera, CCTV	18
Others	TBN, EWTN	28

Table 10: Channel preferences of STB groups based on PREF.

	News	Kids	Sports	Movies	Science	Music	Foreign	Others	Group size (%)
All STBs (%)	52.3	14.4	5.2	3.1	1.8	0.3	0.4	22.4	100
Group1 (%)	**67.8**	9.7	3.5	2.1	1.2	0.2	0.3	15.2	45
Group2 (%)	49.5	**19.0**	4.9	2.9	1.7	0.3	0.4	21.2	12
Group3 (%)	50.2	13.8	**9.0**	3.0	1.7	0.3	0.4	21.5	5
Group4 (%)	50.7	14.0	5.1	**6.0**	1.8	0.3	0.4	21.8	6
Group5 (%)	50.6	13.9	5.0	3.9	**5.1**	0.3	0.4	21.7	3
Group6 (%)	51.0	14.8	5.1	3.0	1.8	**3.0**	0.4	21.8	2
Group7 (%)	51.6	14.2	5.1	3.0	1.9	0.3	**1.8**	22.1	2
Group8 (%)	48.9	13.5	4.9	2.9	1.7	0.3	0.4	**27.5**	27

4.4.2 Measuring Difference in Channel Preferences of STB Groups

In this part, we examine whether STBs in different groups exhibit different channel preferences. We use *mutual information* in measuring the *difference* of channel preferences of STBs belonging to different groups.

In probability theory and information theory, the mutual information of two random variables quantitatively measures their mutual dependence. Formally, the mutual information of two discrete random variables X and Y can be defined as:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p_1(x)\, p_2(y)} \right) \tag{6}$$

where $p(x,y)$ is the joint probability distribution function of X and Y, and $p_1(x)$ and $p_2(y)$ are the marginal probability distribution functions of X and Y respectively. The smaller the mutual information value is, the larger the difference between X and Y is.

We conduct *significance testing* to determine whether the channel preference of a given STB group G is significantly different from that of all STBs S. For this, we

first compute the mutual information I_G between channel preference vector of G and that of S using Equation (6). Here, X and Y are two variables describing channel preferences. In particular, $p_1(x = X)$ is the probability to choose a type X channel for group G. Similarly, $p_2(y = Y)$ is a probability to choose a type Y channel for S. $p(x = X, y = Y)$ is the probability of choosing type X channel in G and choosing type Y channel in S.

Then we randomly select a subset S_i of S, which has the same size as group G. Similarly, we compute the mutual information I_{S_i}. After taking a large number of random selections of S_i, we can get the empirical distribution of I_{S_i}. According to the Central Limit Theorem, $\bar{I}_{S_i}$ is approximately normally distributed with mean $\hat{\mu}$ and deviation $\hat{\delta}$. Here, our null hypothesis H_0 is: group G is not significantly different from S in terms of channel preferences. For the sampled distribution, we compute the *p-value* $\Pr[\bar{X} \leq I_G|(\hat{\mu}, \hat{\delta})]$. If the p-value is very small, e.g., < 0.005, we shall reject H_0. Using this method, we can verify if a group G has a significant difference in the channel preference compared with all STBs S. We can also apply the same method on a given type of channels to determine if G has a significant difference in preference for that type of channels.

Table 10 shows channel preferences of all STBs as well as STB groups based on PREF. There are eight STB groups, each of which corresponds to one type of channels. The size of STB groups varies from 45% of all STBs to 2% of all STBs. The STB group preferring news channels is the largest, and STB groups preferring music and foreign channels are the smallest. We highlight the identified significant difference in channel preference in bold. Compared to all STBs, we observe that each group clearly exhibits distinct preference for the corresponding type of channels. For example, group1 shows significant preference for news channels. These results indicate the potential benefit of modeling different groups separately, which we focus on later in Section 4.4.5.

4.4.3 Identifying Best Grouping Methods

Now we propose a generic method for selecting the best grouping methods. In our case, a "good" grouping should achieve the following two goals. First, the method should yield STB groups that well represent the channel preferences. Second, the resulting STB groups should be stable over time.

To identify grouping methods that yield good representation of channel preferences of STBs, we compute mutual information between STB groups based on PREF and those based on each of other grouping methods (denoted as M) using Equation (6). Here, we consider each STB group as a random variable. $p_1(x = X)$ is the probability that a STB belongs to group X according to PREF. $p_2(y = Y)$ is the probability that a STB belongs to group Y according to a given grouping method M. The joint distribution $p(x = X, y = Y)$ is the probability that a STB belongs to group X based on PREF and belongs to group Y based on M.

It is likely that different grouping methods yield different number of groupings. For example, the location based grouping will yield over 150 clusters while other grouping methods usually yield a handful of groups. In such a case, the mutual information $I(X;Y)$ defined in Equation (6) can be misleading. In order to perform a fair comparison on different grouping methods, we adopt a normalized metric called *symmetric uncertainty*, which is defined as:

$$U(X,Y) = 2\frac{I(X;Y)}{H(X) + H(Y)} \tag{7}$$

where $I(X;Y)$ is the mutual information defined in Equation (6) and H is the entropy:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i), \tag{8}$$

When X and Y are independent, $U(X,Y) = 0$. When X is a function of Y, $U(X,Y) = 1$.

Table 11 shows the symmetric uncertainty between the channel preferences (i.e.,

Table 11: Symmetric uncertainty between PREF and different grouping methods

	WT-D	DN-D	CHG-D	DWL	LC	WT-H	CHG-H	DWL-H	MDWL-H
PREF	0.314	0.305	0.254	0.309	0.179	0.123	0.206	0.430	0.513

Table 12: Stability of different grouping methods

PREF	WT-D	DN-D	CHG-D	DWL	LC	WT-H	CHG-H	DWL-H	MDWL-H
67.1%	83.5 %	79.4%	77.6%	74.3%	100%	70.4%	72.3%	66.5%	69.4 %

PREF) and different grouping methods described in Section 4.4.1. We find that clustering algorithm-based on hourly median dwell time (MDWL-H) and on hourly TV watching time (WT-H) yield the highest and lowest symmetric uncertainty values (0.513 and 0.123) among all the grouping methods. Intuitively, this can be explained as follows. Users who watch TV at the same time during a day does not necessarily watch the same set of channels (i.e., they do not necessarily have a clear mutual interest in channels). However, users who switch channels at the same time during a day may have a strong preference for the type of channels they watch. This is because most of the channel change behaviors are impacted by the start/end times and commercial breaks of the TV program. The symmetric uncertainty values for threshold-based grouping methods range from 0.179 to 0.314, with the grouping based on the daily watching time WT-D having the highest value and grouping based on the location LC having the lowest value.

We also prefer a grouping method that yields STB groups that are stable over time. We perform a stability test on our grouping results in the following way. We use the percentage of STBs that stay in the same group over a certain time period (e.g., 15 days) as the metrics to measure the stability of STB groups. In our analysis, we divided our data traces into two parts, where each part lasts 15 days. We compute STB groups on each 15-day trace separately and examine the stability of STB groups. Note that for clustering algorithm based grouping methods, because the group centers are determined non-deterministically, we group the second 15-day trace by using the

Table 13: Channel preferences of STB groups based on WT-D.

	News	Kids	Sports	Movies	Science	Music	Foreign	Others	Group size (%)
all watchers (%)	52.3	14.4	5.2	3.1	1.8	0.3	0.4	22.4	100
heavy-watchers (%)	**62.6**	9.7	4.9	2.3	2.0	0.4	0.3	19.6	28
light-watchers (%)	47.4	**17.5**	5.4	2.3	1.7	0.4	0.4	25.3	36
medium watchers (%)	53.3	13.9	4.7	3.0	1.9	0.3	0.3	22.5	36

same group centers as those that are identified in the first 15-day trace. Then, for each STB, we compute the distance between attribute vector obtained from the second 15-day trace and each group center identified in the first 15-day trace. The STB is assigned to the group of which the center is closest.

Table 12 shows the stability of different grouping methods. We have four key observations. First, the grouping based on channel preference PREF is not stable over time. This indicates that PREF may not be a good grouping method to be used in our model even though Table 10 shows PREF clearly represents distinct channel preferences in each STB group. Second, we find that all the grouping methods based on hourly features (i.e., WT-H, CHG-H, DWL-H, and MDWL-H) have low stability over time. Hence, they are not considered good grouping methods to be used in the model. Third, we observe that grouping based on location LC yields perfect stability of STB groups. This is expected because STBs location is less likely to change over time. However, since LC has a low symmetric certainty value as shown in Table 11, it is not considered a good choice either. Finally, we observe that the grouping based on daily TV watching time WT-D has the highest stability among all grouping methods other than LC. In addition, WT-D also has a relative high value in symmetric uncertainty as shown in Table 11 (it is the highest among the threshold based grouping methods). Thus, we identify WT-D to be the best grouping method based on our data trace.

4.4.4 Explaining Channel Popularity Dynamics

In Section 4.2, we have observed diurnal patterns in channel access popularity (Figures 23 and 24). In this subsection, we examine whether some of these groups exhibit different channel access preference. Based on the result in the previous subsection,

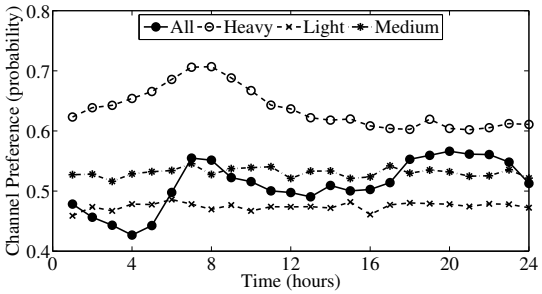


Figure 31: Time-of-day dynamics for news channels, comparing WT-D with all watchers

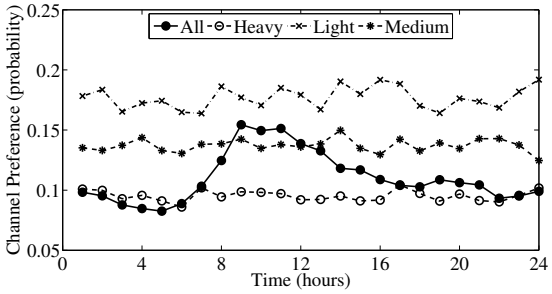


Figure 32: Time-of-day dynamics for kids channels, comparing WT-D with all watchers

we focus on the grouping result by WT-D, because it has the highest stability (except for LC) as well as a reasonably high symmetric uncertainty measure against PREF. Table 13 compares the channel preference of each group based on WT-D with that of all STBs. Based on our significance testing, we find that heavy-watchers group and light-watchers group have distinct preferences to news and kids channels, which are marked in bold. In this subsection, we focus on the preference for these two channel types. Although we do not present here, we also investigated other groupings and observed a similar result in many cases.

In Figures 31 and 32, we show the access probability of news channels and kids channels (as defined in Table 9), respectively. We display one line for each group in WT-D as well as an additional line for the all-STBs case (denoted by "all-watchers"). We observe that while the line for all-watchers shows a clear diurnal access pattern (e.g., with peaks around 7am and 8pm for news channels), the channel preference

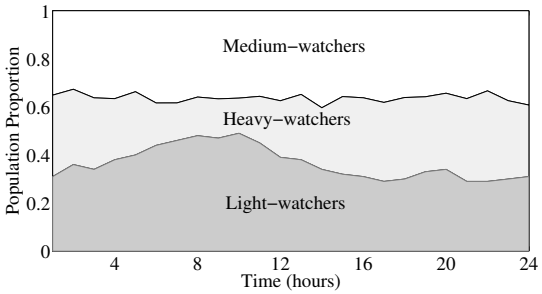


Figure 33: Population mix for each group based on WT-D

within a group is more stable throughout the day, except for heavy-watchers' increased preference during the morning time. Specifically, in Figure 31, at least 60% of channel changes by heavy-watchers are for news channels throughout the day, which is significantly higher than the overall average 52.3%. While the average preference of medium-watchers for news channels (53.3%) is similar to the daily average, this group also exhibits a more stable access pattern, compared to all-watchers. In Figure 32, while the group-level access probabilities for kids channels fluctuate more than for news channels, the values are still more stable than that of all-watchers. These results illustrate that we can identify sub-groups that have distinct channel preference, and although the channel popularity within the groups may vary over time, some groups often have fairly constant channel preference for some channels.

In Figures 31 and 32, we observed that group-level channel preference stays reasonably stable all day, but the overall channel popularity shows a diurnal pattern. How can we explain two seemingly conflicting results? In Figure 33, we show the proportion of active STBs for each group by WT-D. We determine that a STB is active if there is a channel change during the 1-hour window. This figure shows that medium-watchers constitutes around 35% of active STBs throughout the day, heavy-watchers between 15% and 35%, and light-watchers between 30% and 50%. We observe that there is strong correlation between channel popularity change and population mix change. For instance, in Figure 32, the overall channel access probability for kids channels peaks between 8am and noon. This coincides with the population gain by

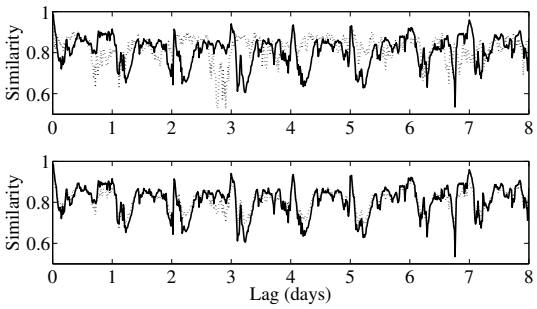


Figure 34: The cosine similarity function when varying lag. The solid line represents the real trace, the dash line represents the model. Top: single-class, bottom: multi-class.

light-watchers (Figure 33), which has significantly higher preference for kids channels. In Figure 31, while the overall channel popularity peaks at the morning time (7am) due to the change in preference by heavy-watchers, the increase of heavy-watchers in the population mix obviously explains the other peak at later time (around 8pm).

In sum, our results show that some sub-groups have different channel preference, and their population mix change has a strong correlation with overall popularity change. In the rest of this section, we further investigate these findings and demonstrate that we can better model channel popularity dynamics by employing this grouping methodology.

4.4.5 Modeling and Simulation

Now we can use our classification results to further improve the modeling results shown in Section 4.3. We use the grouping result from the feature WT-D since it has the highest stability over time and a reasonably large symmetric uncertainty value. Assume that all channels still follow the mean reversion model in each group. But the parameters need to be revisited. Let X_{ij}^t, μ_{ij}, λ_{ij} and σ_{ij} denote the popularity measure, the long-term mean, the mean reversion rate and the volatility of the group j on the channel i, respectively. The estimation procedure described in Section 4.3 can be easily adapted to derived the parameters for every $(channel, group)$ combination.

Then to simulate the temporal popularity dynamics for a channel i, we mix all

$(i, j), j = 1, 2, ...,$ using the empirical population proportion for each group (see Figure 33) as the mixture weight. In other words,

$$X_i^t = \sum_j W_j^t \times X_{ij}^t$$

where X_i^t denotes the popularity of channel i at time t and W_j^t denotes the proportion of STBs in group j at time t.

To evaluate the above multi-class model, we compute cosine similarity (defined in Section 4.2) on the trace generated by our models. We compare the cosine similarity of both single- and multi-class models with that from the real trace. In Figure 34, given a fixed lag, we compute the cosine similarity between the channel popularity vectors of two adjacent 15 minute-time-bins and take average on the length of 9 days. We repeat this by gradually varying the length of lag. This resulting curve from the real trace reflects the degree of similarity of channel popularity across the time domain. The top and bottom subfigures in Figure 34 compare the single-class model and multi-class model to the real trace in terms of cosine similarity, respectively. It is clear that the multi-class model can capture the high daily similarity, but single class model fails to do so. As the result, the MSE of multi-class model is 10^{-3} which is around one order of magnitude smaller than that ($= 9 \times 10^{-3}$) from single-class model. In summary, taking advantage of a good grouping feature with high stability and symmetric uncertainty scores, our multi-class model can generate a more accurate temporal dynamics process to simulate the real scenario than the previous single-class model.

4.5 Related Work

The channel popularity or content/media popularity, in general, has been widely studied in different applications. Costa et al. [16] analyzed user activities and media distribution in media streaming applications. Cherkasova et al. [13], Chesire et al. [14], and Tang et al. [45] modeled workload of media streaming service. Yu et al. [61]

studied the user activities to access a Video-on-Demand (VoD) system. Cha et al. [11] explored how users access videos in the YouTube system. Guo et al. [21] compared access patterns of different types of media content on the Internet including Web, P2P, VoD, and live streaming. However, the findings in these studies may not be applicable to IPTV systems as the user behavior can be inherently different from those in other applications.

More recently, there are a number of studies on IPTV system. Cha et al. [12] report various findings about user watching behavior by analyzing control messages in an IPTV system. While some of our findings are consistent with those reported in their study, we focus developing a multi-class population model of channel popularity based on key observations in our analysis. Smith [44] models bandwidth demand to support both multicast and unicast for fast channel change, where channel switching is modeled as a renewal process. However, the work does not consider the temporal dynamics within a day. Hei et al. [23] and Silverston et al. [43] report their measurement studies on P2P-based IPTV systems, while our work focuses on analyzing and modeling a large commercial IPTV system.

4.6 Summary and Future Work

In this chapter, we analyze and model channel popularity based on user channel access data in a nation-wide commercial IPTV system. We find that the channel popularity is highly skewed and can be well captured by a Zipf-like distribution. We also observe a fair amount of channel access popularity change during a short time window, although we find that channel popularity during moderately long time windows stays relatively stable. We demonstrate that we can model such popularity dynamics using a mean reversion process. Further, we develop a method for identifying groups of users which show intrinsic difference in their channel preference. We demonstrate that we can combine this grouping and the change of population mix to obtain a multi-class

population model, which enables us to capture diurnal patterns in channel popularity dynamics.

Although the focus in this paper is on analyzing and modeling channel popularity in an IPTV system, our methodology can be applicable to other systems, which we plan to investigate in our future work.

CHAPTER V

MODELING IPTV USER BEHAVIOR

5.1 Introduction

In the past several years, there has been a global trend among telecommunication companies on the rapid deployment of IPTV (Internet Protocol Television) infrastructure and service, in which live TV streams are encoded in a series of IP packets and delivered to users through the residential broadband access network. This fast growth is motivated in part by commercial reasons – strengthening their competitiveness with so-called triple-play package that combines digital voice, TV and data service together. More importantly, this new technology provides the users with great interactive capability and functional flexibility, and creates tremendous opportunities for a broad range of new applications (e.g., CollaboraTV [22]), which may very well define a next generation of TV entertainment.

While the industry rushes into the IPTV era, what lags behind is a comprehensive understanding of the user activities, which directly dictate the flow of video streams and other bi-directional data (e.g., for user interactive sessions). Although commercial TV distribution networks (e.g., cable, satellite) have prevailed for decades, to the best of our knowledge, no detailed study on modeling individual users' TV watching activities is available in the literature. This might be partially because there has not been a strong need, as data flows in conventional TV networks are typically limited to the downstream direction from servers to set-top boxes (STBs), and user channel switching (with the exception of pay per view) has very little system-wide impact. In fact, even tracking the viewership of TV program – a statistic that bears significant commercial value – is typically done through a third party [3]. In IPTV

systems, by contrast, an understanding of user activities is essential to many system design and engineering tasks such as evaluation of various design options, optimal system parameter tuning, improving customer care, and defining effective system care procedures to minimize service impact.

Without a realistic user activity model, the research community often has had to rely on some hypothetical user models when analyzing system performance [44, 48, 4, 50]. Unfortunately, such models are sometimes quite different from the reality and can potentially lead to incorrect estimation of the system performance. For example, while a constant-rate Poisson process is widely used as a workload model in other systems, it is incapable of capturing the high bursts of channel switches at around hour boundaries observed in our IPTV data. An alternative is to directly use actual IPTV trace data for the evaluation of system performance. However, such data, even when anonymized, can be highly sensitive, containing too much commercial and user private information to be publicly distributed. This creates a barrier for research community to perform system evaluation against real data traces. In this work, we bridge this gap by developing realistic models for user activities in a large IPTV system.

Our work in this chapter is based on a large collection of data obtained from a nation-wide operational IPTV network, which includes the system logs from all of its subscribers' STBs, control plane signaling messages, network topology and configuration data, and TV channel information. Our approach starts with an in-depth investigation of the user activities, analyzing many intrinsic characteristics on attributes such as user viewing sessions, per-channel dwell time, and channel popularity. While some of our findings overlap with a previous study [12], we further abstract and generalize the chosen characteristics to enable *realistic workload generation*, which can be used for various stages of IPTV system design. Specifically, we develop a workload generator that faithfully mimics the user activities in real IPTV systems—this

workload generator can turn a limited number of input parameters (published in the paper) into synthetic traces having similar statistical properties to realistic data traces.

We also consider this work a snapshot of user activity workload for the current IPTV system, which provides a feature set highly similar to that of conventional TV services such as cable and satellite. We envision that user viewing pattern evolves with more advanced IPTV features fundamentally changing the way users watch TV, and this work is used as a baseline to understand and to quantify such changes.

We make three major contributions. First, we present in-depth analysis results based on data traces from a nation-wide operational IPTV system (Section 5.2). In addition to the largest scale of such study (using more than a million STBs in four different time zones), we identify many interesting characteristics. For example, we find that user activities (such as channel switching) are often correlated, hence the aggregate activities are much more bursty than the outcome of a fixed-rate Poisson model that many previous studies assume [4, 50].

Second, we develop a series of models that capture these intrinsic characteristics on each of the attributes (Section 5.3). We use the mixture exponential distribution to model various session duration distributions. To characterize the time-varying nature of user activities, we apply Fourier Transform and model the periodically correlated events. We distinguish sequential-channel-scans and targeted-channel-switches and use Zipf-like and exponential distribution to characterize channel access popularity. We also adopt a mixture population model to capture the channel popularity dynamics observed at the finer time granularity.

Third, we combine these models and construct a workload generation tool, namely SimulWatch (Section 5.4), which takes a small number of parameters as input and outputs a series of synthetic user traces that mimic a set of real users watching IPTV. We also validate SimulWatch prototype by comparing the synthetic trace with a

real data trace and show that they closely match even for some properties that we do not explicitly model. Specifically, we show that for a given number of STBs, we can accurately estimate the unicast and multicast traffic bandwidth based on the synthetic workload, which also illustrates how to use SimulWatch to drive the performance study in an IPTV system.

We also review related work in Section 5.5 and conclude our work in Section 5.6. In the following section, we first overview a typical IPTV system architecture and describe the data set we use in this study.

5.2 Analyzing User Activities

Recall that our objective is to define a mathematical process that mimics the activities of IPTV users and thus can produce realistic event series for tasks such as system performance evaluation. To accomplish this, we first need to understand how real users act in an operational IPTV system. We do so by studying various characteristics of our data traces. In particular, we focus on the aggregate properties regarding to users' turning STBs on and off, channel switches, and channel popularity.

5.2.1 Turning STBs On and Off

We first focus on the length of STB on- and off-sessions. An *on-session* is defined as the duration from a STB being switched on till it gets switched off. Similarly, an *off-session* is the duration from the last time a STB was switched off till it gets switched on. We first examine the distribution function of the length of on- and off-sessions respectively.

Figure 35 shows the complementary cumulative distribution function (CCDF) of the length of on- and off-sessions. Using CCDF, we can better illustrate the tail property of the distributions. We first observe that both on-sessions and off-sessions exhibit a very long tail in their distributions – around 5% of the on-sessions and off-sessions are over 1 day in length. In fact, we believe the fast drop in both tails

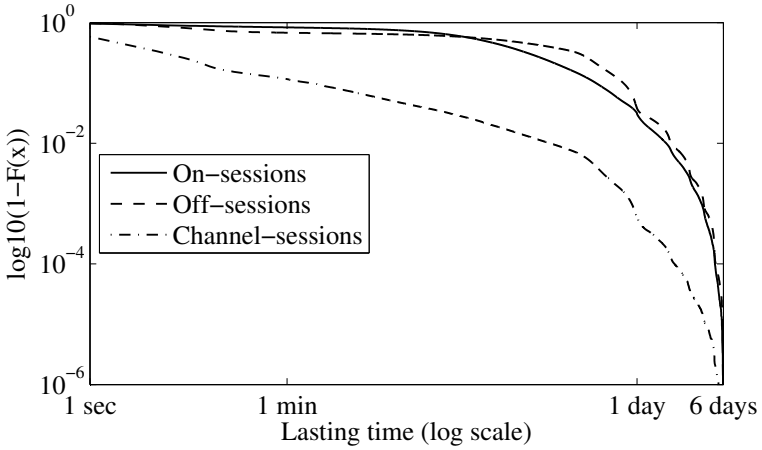


Figure 35: CCDF of the length for on-, off-, and channel-sessions

approaching the right end of the x axis is due to the limit of our dataset, which is 6 days in total. Comparing off-session and on-session, we find that the off-session has a heavier tail than the on-session. This matches our intuition since it is more likely that an IPTV user leaves the TV off for a long time (several days) than leaving the TV on. We also notice that the curve of off-session is below that of on-session for low session length. This is likely due to users' mistake in operating the remote controller – a user accidentally turning the STB off while watching a TV program may quickly switch the STB back on, producing a short off-session of a few seconds.

In Figure 36, we show the time series of the number of on-line STBs in one-minute precision (normalized by the average number of on-line STBs). We observe a very strong diurnal pattern, with daily peak at around 9PM, followed by a quick decrease in number, reaching daily minimum at around 4AM, and then steadily ramping up during the course of day. Note that there are a significant number of STBs left on over night.

As both on- and off-sessions are bounded by users' action in switching on and off the STBs, it makes sense to observe these event processes directly. Figures 37(a) and 37(b) show a one-day time series of the event rate for the *switching-on* and *switching-off* events respectively. Both plots are shown in one-minute precision. Here the event

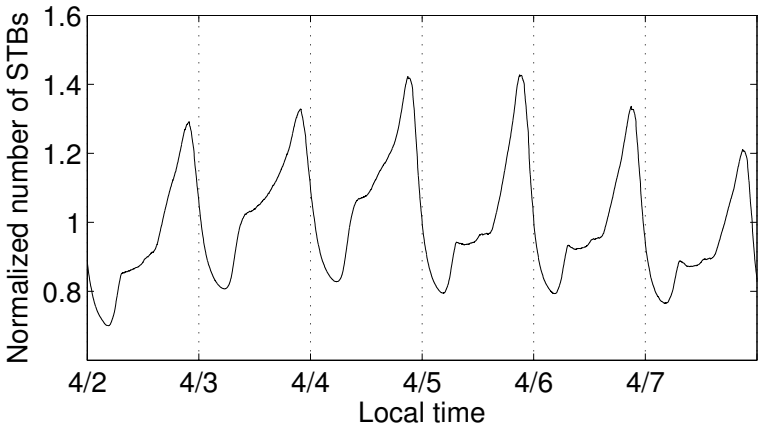
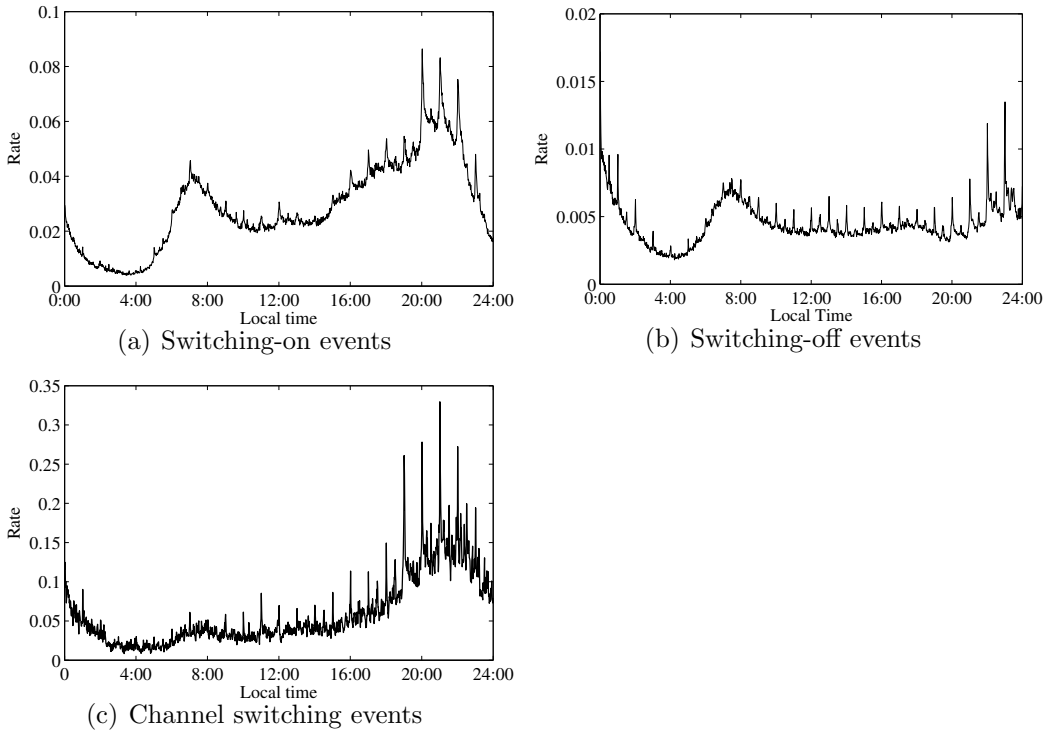


Figure 36: Number of on-line STBs



(a) Switching-on events

(b) Switching-off events

(c) Channel switching events

Figure 37: The normalized switching-on, switching-off, and channel switching events (one-minute granularity)

rate is the number of switching-on/-off events during the interval normalized by the total number of off-line/on-line STBs at the beginning of the interval. We make two observations. First, there is a strong time-of-day effect in both figures.The switching-on event rate has local peaks at around 7AM and around 9PM and the switching-off

event rate has local peaks at around 7:30AM and around 12AM, both matching well with our intuition relating to the daily living schedules of most people. More interestingly, we observe that both event rate series are very bursty, with significant spikes aligning closely with hour or half-hour boundaries (it is more pronounced in Figure 37(b)). This is due to the fact that most TV programs are aligned to hour boundaries. Many users may turn on TV in anticipating for a TV program or turn off TV after watching a TV program. This introduces significant correlations among users' activities, causing very strong bursts in the aggregate event rates.

5.2.2 Switching Channels

We now turn to channel switches. Figure 35 shows the distribution function of the length of channel-sessions, which we define as the duration from the time of a user's last channel switch (or turning on STB) till the next channel switch (or turning off STB). We find that this distribution also has a long tail, although not as heavy as those of on- and off-sessions.

Similar to those for switching-on and switching-off events, we also examine the aggregate event process for channel switches. Figure 37(c) shows the time-series of such event rate, which is defined as the total number of channel switches normalized by the number of on-line STBs. We note that the diurnal pattern in Figure 37(c) is quite representative of different days. Compared to switching-on/-off event rates, the channel switching events demonstrate even stronger spikes with the period of 30 minutes. This is again due to correlated user activities related to TV program alignment – many users switch channels together when a TV program ends, which may create temporarily high workload on the IPTV servers.

We next try to gain insight on how IPTV users switch channels. We classify channel switching events into two categories: *sequential-scanning* and *target-switching*. Sequential scanning represents the user in a channel-browsing mode by going through

the available channels using the *Up/Down* button on the remote controller, while target switching represents the user intentionally switching to a specific channel of choice. We assume channel switches between adjacent channels being the sequential-scanning and the rest target-switching. To define the channel adjacency, we need to infer the list of available channels, which can be quite different from one user to another (e.g., due to different subscription plans). We keep track of all channels that a STB requests over an extended period (e.g., one month) and regard these channels as the complete list.

From the data we collected, we observe that 56% of channel switching events are sequential-scanning. This is a little bit lower than our expectation. We find that the high ratio of target switching can be attributed to many advanced features that the IPTV provider supports, including a user-customized favorite-channel list, a program menu where users can browse and switch channels by name, and an easy access to DVR. All of these help users find the TV program of interest easily and directly. To understand this effect better, we construct a user's favorite channel list using a heuristic (top ranked channels by watching time and frequency, e.g., watched in at least 4 days of a week) and find a large portion (46%) of the target-switching is toward such "favorite channels". Among sequential-scanning, we observe an unbalanced up- and down-channel-switches – 72% of them are up-channel-switches. It implies that more people prefer increasing channel number. Our analysis finds the ratio moderately stable over time, although we do not have a good intuition on why this is the case.

5.2.3 Channel Popularity

We now focus our attention on the properties of different channels. We first rank nearly 700 different channels that appeared in our data using two metrics: (1) the request count, which we call *channel access frequency*, and (2) total time STBs stay

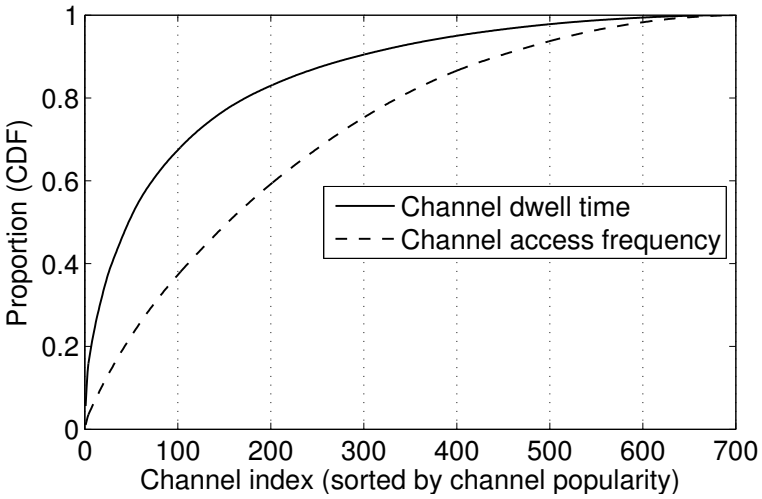


Figure 38: CDF of channels popularity

tuned in the channel, which we call *channel dwell time*. Figure 38 shows the cumulative distribution function of channel popularity ranked by the two metrics. We find that the distribution of channel dwell time is highly skewed – the top 100 channels account for around 63% of the total channel dwell time. As a comparison, the channel access frequency curve is less skewed. This is likely due to the large number of sequencial-scanning channel switch events. We observe similar level of skewness in the distribution of channel popularity when we examine different subsets of our data (such as by different time zones or by different date), although the ranking of the channels varies from one subset to another.

Figure 39 shows how the top 10/50/100 popular channels change in the two adjacent hours during a day. The change percentages are averaged over 6-day data. We observe that the channel popularity is relatively stable over time of day. For example, in Figure 39, among top 100 channels at 12pm, less than 20% of them did not belong to top 100 channels at 1pm, while more than 80% of them were among top 100 channels at both time periods. We find that the relative channel popularity changes the most during morning hours, but remain moderately stable for most part of the day. Figure 40 illustrates an example on the dynamics of channel popularity

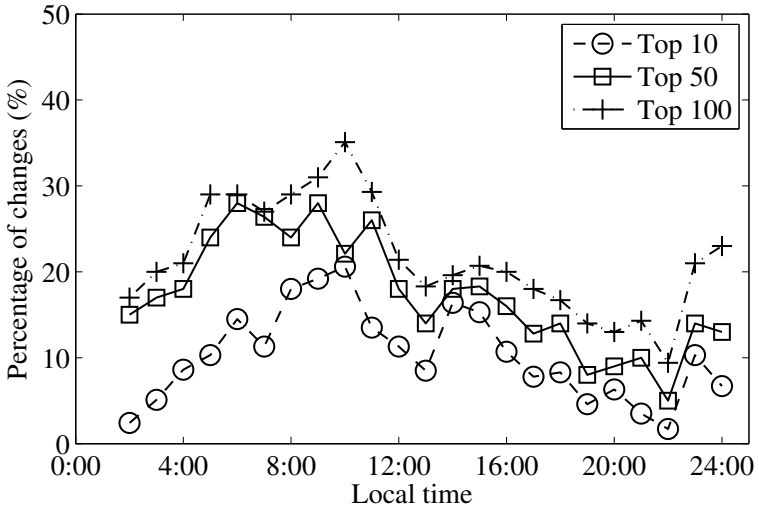


Figure 39: Ratio of change in popular channels, as seen over hours in a single day.

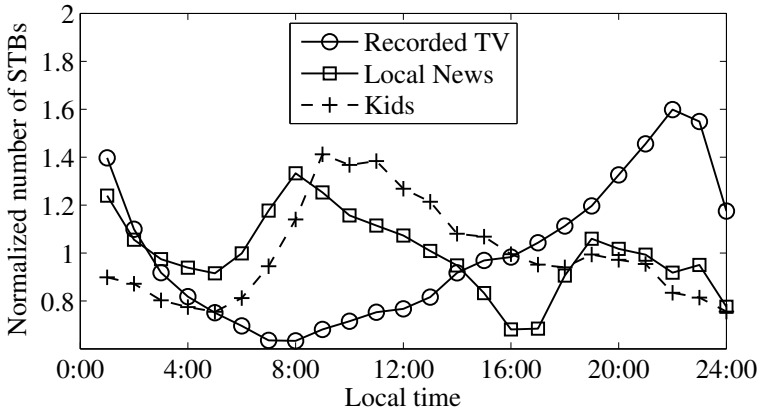
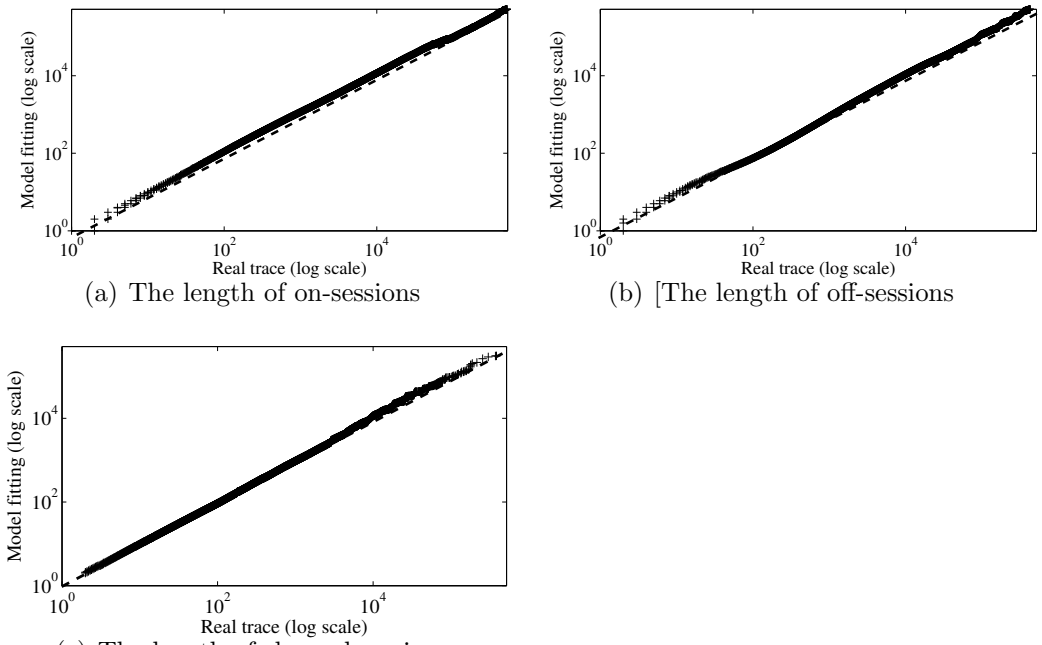


Figure 40: Channel popularity distribution change (hourly)

within a day, in which we compare normalized numbers of STBs of a top-ranked kids channel and a top-ranked local news channel against that of the "recorded TV" that people use to watch recorded contents (called DVR channel). We observe some interesting time-of-day trends – for example, the local channel peaks in the morning when people catch early news and weather forecast before going to work; the kids channel sharply loses popularity after 8PM when most kids go to bed. In comparison, the DVR channel has the most dramatic change in scale, which finds its peak late into night. The recent work [42] uncovered the reasons behind this behavior by grouping the whole user population into subgroups according to their preference. We will also



(a) The length of on-sessions

(b) [The length of off-sessions

(c) The length of channel-sessions

Figure 41: QQ plots comparing models and real traces

integrate this into our workload generator in Section 5.3.4.

5.3 Modeling User Activities

In this section, we construct mathematical models to capture the observed characteristics of IPTV user activities. We need to model three different user activities – *switching-on*, *switching-off* and *channel-switch*. For each of them, we match their timing properties in both the session length distribution and the dynamics of the aggregate rate. For channel-switch, we also model channel popularity properties including popularity distribution and its temporal dynamics. We first present our models and then describe our methods to deriving the parameters of our model from the data traces.

5.3.1 Modeling Session Length

In order to capture the long tails exhibited in the empirical session length distributions (Figure 35), we adopt the mixture-exponential model [26] for on-, off-, and

Table 14: Model parameters for session length distributions

	λ_1	a_1	λ_2	a_2	λ_3	a_3
On-session	1.3e-2	0.3	3.3e-3	0.66	2.3e-4	0.04
Off-session	3.2e-2	0.19	2.5e-3	0.75	2.4e-4	0.06
Channel-session	2.1	0.23	2.6e-2	0.64	3.2e-3	0.13

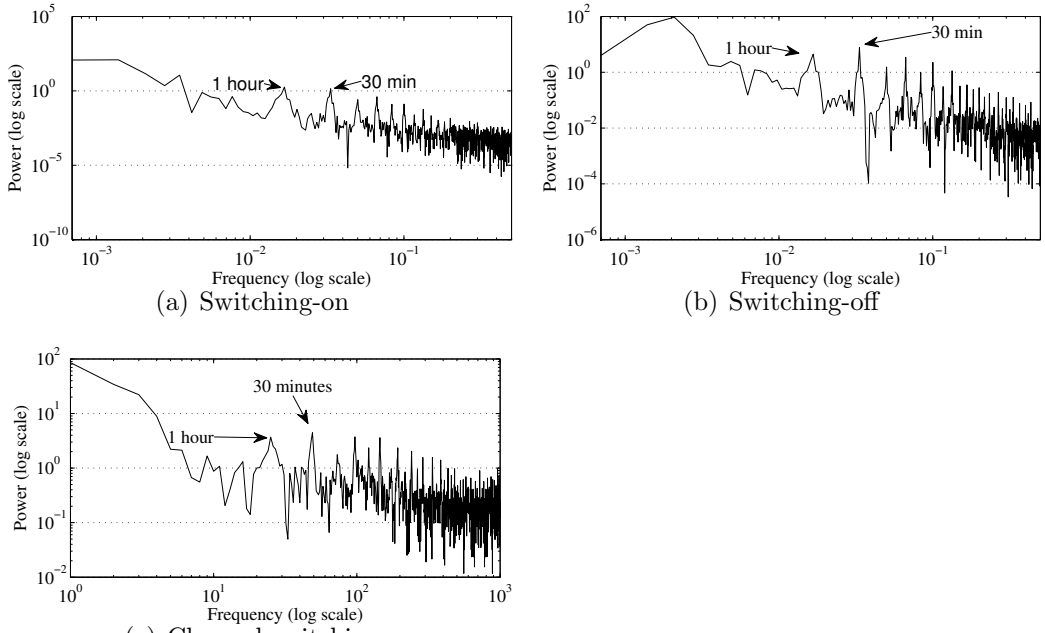
channel-sessions. The probability density function (PDF) of a mixture-exponential distribution is

$$f(x) = \sum_{i=1}^{n} a_i \lambda_i e^{-\lambda_i x} \tag{9}$$

where $1/\lambda_i$ is the mean of the i-th exponential distribution in the mixture and $\sum_{i=1}^{n} a_i = 1$. This model has been widely applied due to its simple form and its capability in approximating heavy-tailed distributions in a wide range [26].

To determine the model parameters that best describe the data trace we collected, we apply data fitting for on-, off-, and channel-sessions respectively. In the following, we use channel-sessions as an example while the procedure for fitting on- and off-sessions is essentially the same. We iteratively explore different values for the number of exponential distributions, n, in the mixture model. For a given n, we apply the Expectation Maximization (EM) algorithm [7] to find the maximum likelihood estimate (MLE) for the parameters λ_i and a_i. For the length distribution of channel-sessions, we identify the best tradeoff at $n = 3$, as it achieves a close match to the data while using a small number (i.e., 6) of model parameters. In Table 14, we report the parameter values that fit our trace. The QQ (quantile-quantile) plots in Figure 41 demonstrate good matches between our models and real traces collected.

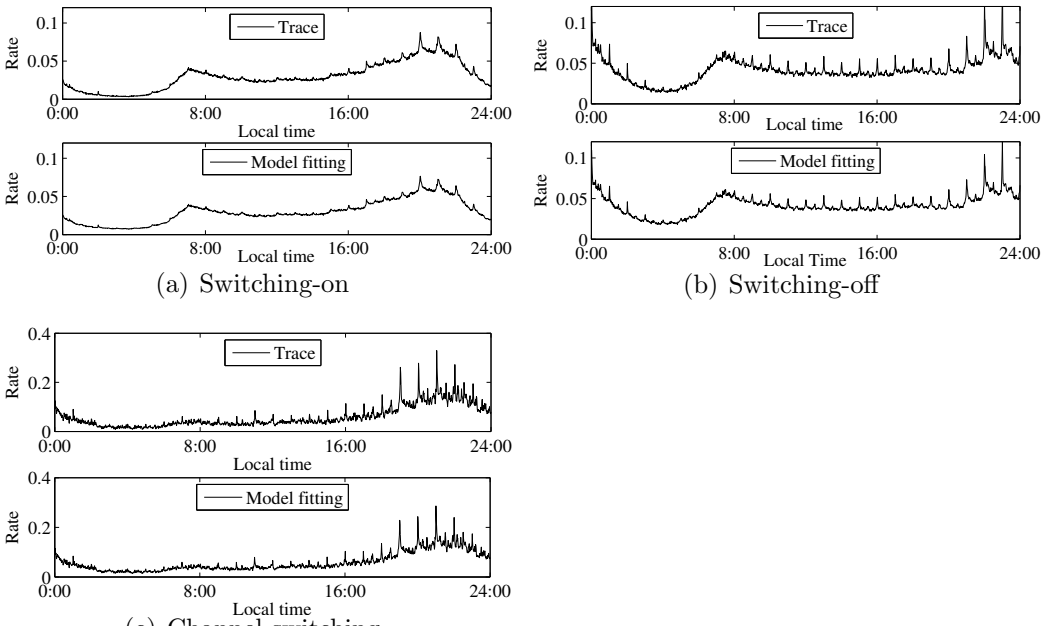
Looking into the parameters, we gain tremendous insight on the process. For channel-sessions, the different λ_i corresponds to Poisson processes with average inter-arrival time of around 30 seconds, 40 minutes and 5 hours, representing an IPTV user in the state of channel-browsing, TV-program-watching, and being away-from-TV respectively. The likelihood of a user entering these modes is quantified by the



(a) Switching-on (b) Switching-off

(c) Channel-switching

Figure 42: The time-varying rates in frequency domain



(a) Switching-on (b) Switching-off

(c) Channel-switching

Figure 43: Modeling aggregate event rate

a_i values. Similar observation can be made for on-sessions and off-sessions too.

Table 15: Modeling parameters for event rates

	k	μ	$p_{1\text{-hour}}$	$p_{30\text{-min}}$	$p_{15\text{-min}}$
Switching-on	0.0036	278	1.76	1.41	x
Switching-off	0.0316	233	4.43	7.85	x
Channel-switch	0.03840	293	4.23	5.34	4.53

5.3.2 Modeling Time-Varying Rates

The mixture-exponential models in the previous subsection imply a constant-rate stochastic process with the mean event rate equal to $1/(\sum_i a_i/\lambda_i)$. However, we have observed in Figure 37 that the aggregate event rate for switching-on, switching-off, and channel-switches are all highly variant, highlighted by many apparently-periodic spikes. The problem lies in a subtle underlying independence assumption (which has been commonly used in similar study without careful validation). The reality is that each individual user's activities are influenced by a common external process – the TV program schedules, and as a result, they become highly correlated to each other, breaking the independence assumption. In this subsection, we incorporate this impact from the external process through modeling the aggregate event rates in Figure 37.

As the aggregate event rates appear very complicated in the time domain – requiring an overwhelming number of parameters to characterize it, we decide to try a different angle and approach the problem from the frequency domain. We apply fast Fourier transform (FFT) to the event time series and present the result in Figure 42. The structure, in all cases, suddenly becomes very clear – there are a few of distinct spikes at frequencies that correspond to 1 hour, 30 minutes, 15 minutes etc., and an ambient gradual decrease in the power level (y-axis) from low to high frequencies. We next approximate the ambient power level by using the Weibull distribution. Its probability density function is:

$$f(x;k,\mu) = \frac{k}{\mu}\left(\frac{x}{\mu}\right)^{k-1} e^{-(x/\mu)^k}$$

where k and μ are model parameters. We choose the Weibull distribution since it

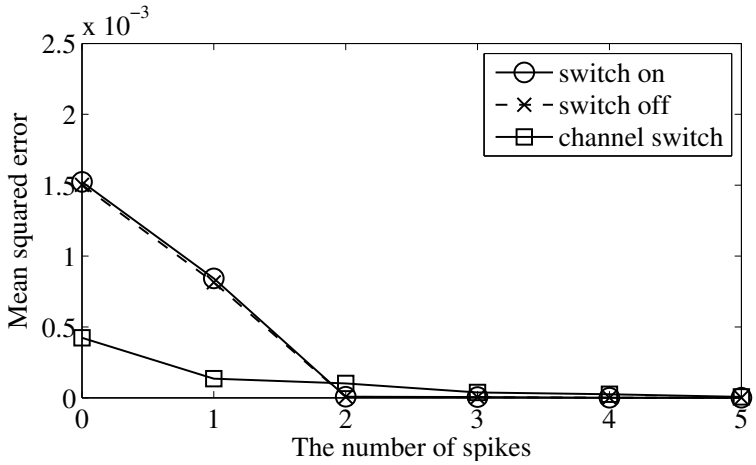


Figure 44: Find the optimal number of spikes

can very well approximate a wide range of classes of functions including exponential, normal and lognormal only with two parameters. The model parameters that best match our data traces are reported in Table 15.

From only a small number of parameters (k, μ, and the values for the spikes in the frequency domain), we can now generate the frequency domain function and apply inverse FFT to reconstruct the time series that initially seemed highly complex. To determine the best trade-off between the number of spikes to explicitly include in the model and the quality of the match between the model and the empirical trace, we show in Figure 44 the discrepancy metric (we use mean squared error between the empirical trace and our modeling output) as a function of the number of spikes in the model. It is clear that the discrepancy becomes negligible when we choose 2 spikes for switch on/off and 3 spikes for channel switch. So they are the values we use in the rest of the paper. Figure 43 compares the result from the real trace (top) and the result from our model (bottom). We find that they match very well, even when we use only 13 parameters here (k, μ and the value for the 2-3 spikes in Table 15).

Finally, we define the time series function obtained from the aforementioned process as our *rate moderating function* $g(t)$, which models the impact of external TV program schedule to individual users' activity. Since $g(t)$ is constructed from data

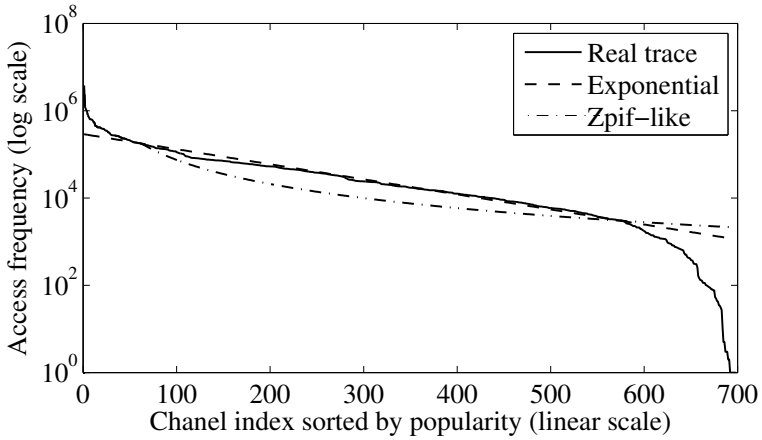


Figure 45: Fitting the channel popularity distribution

in a given window W, (in our example $W = 86,400$ seconds, or 1 day), we simply repeat $g(t)$ to make it a periodic function: $g(t + W) = g(t)$. Furthermore, we normalize $g(t)$ such that $\int_0^W g(t)dt = W$. Note that the periodic moderating function g will not impact the tail behavior of the session length distributions that we have modeled previously. However, it does change the shape of session length distribution at small durations. In particular, depending on the start time-of-day, the session length distribution varies.

5.3.3 Modeling Channel Popularity Distribution

We have observed in Section 5.2 that the channel popularity is highly skewed. Motivated by the success of Zipf-like distribution in modeling skewed access frequencies of Web [6] and VoD systems [45, 61], we also examine the Zipf-like distribution in modeling channel access frequencies – for a channel of popularity rank i, the access probability is a power function of its rank i. Figure 45 shows the channel access frequency as a function of the rank, along with the best-fit power law function and the best-fit exponential function. We find the Zipf-like distribution well captures the top 10% channels (consistent with our previous study in Section 4.3.1) while the exponential function achieves a better fit for the large "body" part of the distribution function. The parameters for the Zipf-like distribution, $f_1(i) = C_1 i^{-\alpha}$, are $\alpha = 0.513$,

$C_1 = 12.642$. The parameters for the exponential function, $f_2(i) = e^{-\beta+C_2}$, are $\beta = 0.006$, $C_2 = 2.392$. In the rest of the paper, we use a hybrid model – approximating the top 10% of the channel popularity distribution using the above Zipf-like power-law function and the remaining part using the exponential function). Particularly, the probability density function can be expressed as follows,

$$f_0(i) = \begin{cases} C_1 i^{-\alpha}/C_0 & i < 10\% \text{ of available channels,} \\ e^{-\beta+C_2}/C_0 & others, \end{cases}$$

where C_0 is the normalization factor such that $f_0(\cdot)$ is a well-defined density function.

The concatenated distribution function achieves a good match for the top 600 popular channels, which together account for over 97% of the channel-switches (as shown in Figure 38).

Channel popularity in terms of channel access frequencies is only applicable to target-switching. For sequential-scanning, the channel number simply increments or decrements. We define the probability of user entering target-switching mode as p_t, which is 0.44 in our data. The probability of user entering sequential-scanning mode is hence $1-p_t$. When in sequential-scanning state, a user switches to a higher number with the probability of p_u (0.72 in our data), and to a lower number with $1 - p_u$.

To align the channel ID to the channel popularity, we adopt a simple random permutation method – we randomly shuffle the ranks of the channel popularity and use them as the channel ID. This however does not capture the subtle clustering effect in the commercial channel listing, such as music channels being next to each others. Depending on the application, a detailed modeling of such effects can be of interest.

5.3.4 Modeling Channel Popularity Dynamics

The channel popularity model described in the previous section captures popularity skewness, which have been found relatively stable at large time scale (e.g., daily [42]).

However, we also observe from our data that channel popularity exhibits some temporal patterns over time-of-day (See Figure 40). While a stationary channel popularity model might be sufficient for many applications (for example network capacity planning analysis), we expect that some other applications (for example evaluating a P2P type content caching scheme for IPTV) may require a proper modeling of such channel popularity dynamics. One way of modeling such dynamics is to observe the differences across multiple smaller time intervals (e.g., hourly granularity) and model the channel popularity in each small interval separately. Alternately, we can try to understand the underlying structure producing such dynamics and model this underlying process. In fact, in the previous chapter, we demonstrate that the channel popularity dynamics can be well explained by groups of users that have intrinsically different channel preference and tend to watch TV at different time of day, as described next.

We divide STBs into multiple classes according to some feature. In the previous chapter, we have compared different choices for such feature. To model the daily dynamics of channel popularity, we choose *average daily watching time* as our classifier because (1) the resulting subgroups exhibit distinct and stable channel preference and (2) the STBs in each subgroup tend to affiliate with the same subgroup over time. Specifically, we classify STBs using two thresholds, and in our data, 28% of STBs are *heavy-watchers* (12 hours or longer average daily watching time), 36% of them *light-watchers* (1 hour or shorter average daily watching time), and the remaining 36% *medium-watchers*. We find that the channel preference among STBs in a particular subgroup stays stable (e.g., throughout a day), and the overall channel popularity dynamics is largely due to the change in the population mix of those groups.

We thus extend the model to a multiple-class population model. We first define the membership ratio for different subgroups using the numbers above. Next, we identify the channel preference within a subgroup (which is stable and follows Zipf-like distribution) and characterize the session lengths and moderating functions for

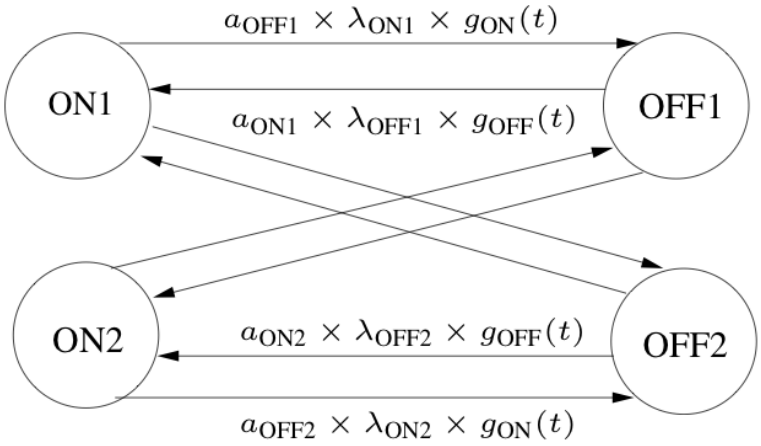


Figure 46: ON-OFF model

each subgroup separately. This would capture the change of population mix over time. We present more details and evaluations of this method in the next section.

5.4 SimulWatch*: A Workload Generator*

Thus far we have constructed several models to characterize various aspects of IPTV user activities. In this section, we present our design of SimulWatch – user TV watching activity generator. We validate our tool by comparing its output with the real data traces. We also demonstrate how to use this tool to drive the network performance study in an IPTV system (e.g., estimate unicast and multicast traffic rates given the number of subscribers).

5.4.1 SimulWatch Design

For simplicity, we first describe the design of SimulWatch based on the single-class population model. Then we present the extension using the multi-class population model if the dynamics of channel popularity is of interest.

In the single-class population model, we first focus on generating switching-on and switching-off events matching both on- and off-session length distributions and the aggregate event rates. We define a closed-population ON-OFF model where both ON and OFF states comprise several sub-states, each of which corresponds to one of the

mixture exponential distributions in Section 5.3.1. Figure 46 illustrates the structure of the ON-OFF model with 2 sub-states in each of the ON and OFF states. The transition rate between sub-states are constructed using the parameters in Sections 5.3.1 and 5.3.2. For example, the transition rate from ON_i to OFF_j state is

$$a_{\text{OFF},j} \times \lambda_{\text{ON},i} \times g_{\text{ON}}(t)$$

and similarly the reverse direction rate from OFF_j to ON_i is

$$a_{\text{ON},i} \times \lambda_{\text{OFF},j} \times g_{\text{OFF}}(t)$$

To drive the event simulation, assuming a STB arrives at state ON_i at time t, we can easily determine the edge of the next transition using the branching probabilities $a_{\text{OFF},j}$, and we can also determine the time of the next transition, $t + x$, using the following probability density function

$$\phi_{\text{ON},i}(x;t) = \lambda_{\text{ON},\,i} \times g_{\text{ON}}(t+x) \times e^{-\lambda_{\text{ON,i}} \int_t^{t+x} g_{\text{ON}}(y)dy}$$

We next focus on generating channel-switch events. It is not hard to see that the timing of channel-switch events can be determined in the same fashion as those of switching-on or switching-off events. There are two subtle details worth noting. First, we need to trigger the event generation for a next channel-switch event not only at the time of the previous channel-switch, but also when a new switching-on event takes place. Second, we need to cancel a pending channel-switch event if a switching-off event from the same STB takes place first.

In order to determine which channel to switch to, we keep track of, for each STB, the last channel watched. At the time of a scheduled channel-switch event, assuming the last channel watched is i with popularity rank r_i, we compute the probability that the next channel is j with rank r_j as follows.

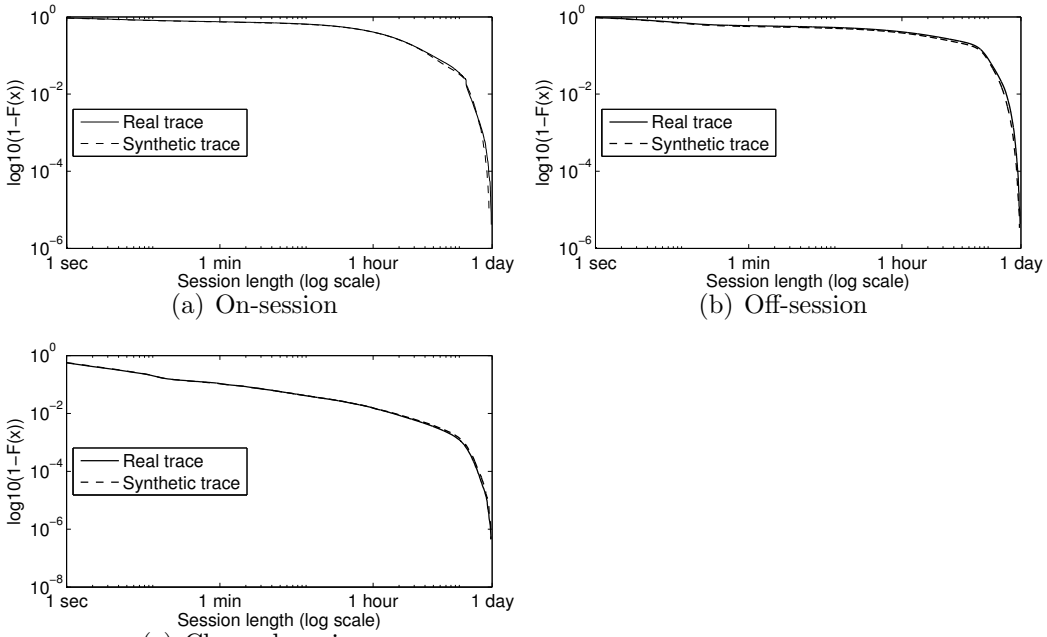
$$\text{Probability} = \begin{cases} (1-p_t)p_u + p_t f_0(r_j) & j = i+1, \\ (1-p_t)(1-p_u) + p_t f_0(r_j) & j = i-1, \\ p_t f_0(r_j) & |i-j| > 1. \end{cases}$$

The initial rank r_i is randomly assigned as described in Section 5.3.3. The definition of f_0 and all other parameters involved are defined in Section 5.3.3.

Now we have described the design of SimulWatch using the single-class population model. We will show that the above procedure simulating the channel switches cannot precisely generate the dynamics of channel popularity. To equip SimulWatch with this functionality, we add an extension of the multi-class population model as follows. Assume that we obtain N classes/groups, each of which consists of a fixed proportion, $p_i, i = 1, 2, ..., N$, of all STBs, where $\sum_{i=1}^{N} p_i = 1$. Then for each STB, we first determinate which group it belongs to based on the probabilities p_i's. Then in each group, the workload is generated using the same method as we described above using the single-class population model with the proper parameters. To generate the synthetic traffic to mimic the dynamics of channel popularity, we can determine the mapping between channel ID and its popularity rank within each group and then the channel popularity can be calculated by combining the results from all the groups.

5.4.2 Evaluation

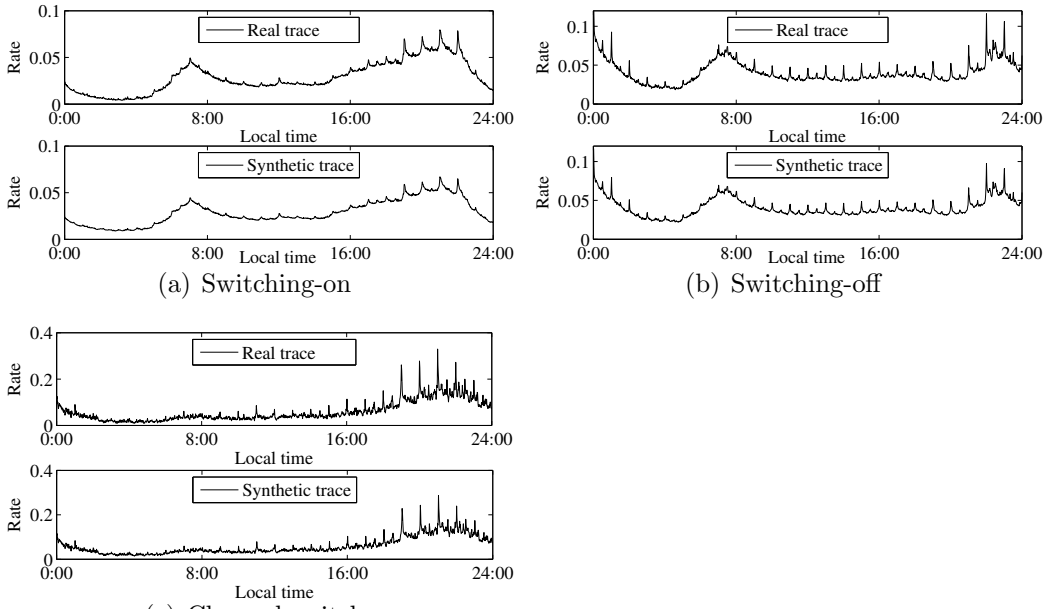
In this subsection, we will evaluate whether the synthetic traces generated by our SimulWatch mimic the real user activities very well. We do so by comparing the synthetic traces and real traces from three aspects – (i) properties that we explicitly model such as session length distribution, aggregate event rate, and channel popularity distribution, (ii) properties we do not explicitly model like channel popularity dynamics and numbers of on-line STBs, and (iii) a case study on estimating the bandwidth consumed by simultaneous unicast streams, and concurrent multicast channels



(a) On-session (b) Off-session

(c) Channel-session

Figure 47: Comparison of the session-length distribution. CCDFs for the real trace and generated workload closely match in all cases.



(a) Switching-on (b) Switching-off

(c) Channel-switch

Figure 48: Comparison of the aggregate event rate. The real-trace results are on the top, and the workload results are on the bottom.

at different time.

In our experiments, we generate synthetic user activities for two millions STBs

and 700 channels based on model parameters listed in Section 5.3. Each STB starts from a random state at time 0 and we discard the initial part of the output until the system reaches a steady state. On a PC with 2.4GHz CPU and 4GB memory, it takes about 5 hours for our implementation of SimulWatch to generate one-day worth of data. We compare the synthetic trace against the real trace collected on a different date (April 8, 2009) than the dates from which we derive the model parameters. Since the single-class population based workload generator works reasonably well for many properties, we use the single-class population model unless specified otherwise for the interest of simplicity. We also use the multi-class population model when illustrating its capability in capturing the dynamics of channel popularity.

5.4.2.1 Properties explicitly modeled

Session-length distribution: Figures 47 shows the session-length distribution of different types of sessions, where we observe an exceptionally good match between the real trace and the synthetic trace from SimulWatch by visual inspection. In order to qualitatively measure the closeness of two distributions, we further compute the *goodness-of-fit*. In the chi-square goodness-of-fit computation, we divide the data into m bins and test

$$\chi^2 = \sum_{i=1}^{m} (O_i - E_i)^2 / E_i$$

where O_i is the observed frequency for bin i (generated by model) and E_i is the expected frequency for bin i (collected from the real trace). The smaller the value is, the better the model and trace match. First, we want to test whether observation O can be considered as arising from the same distribution as E. We represent it through associated one-sided chi-square P-value $P(\chi^2)$, i.e., the proportion of the time that a value of χ^2 or greater would be obtained if O and E were drawn from the same distribution. For a hypothesis testing at significant level P_0, we reject the null hypothesis (O and E are from the same distribution) if $P(\chi^2) < P_0$.

Table 16: Goodness-of-fit scores for session length and channel popularity distributions

Model	Session length			Channel popularity
	ON	OFF	Channel	
Single-class	0.147	0.132	0.132	0.083
Multi-class	0.099	0.089	0.091	0.067

Table 17: RMSE when modeling the time-varying rate

Model	Switch-on	Switch-off	Channel-switch
Single-class	2.3e-3	2.4e-3	2.5e-3
Multi-class	1.8e-4	1.9e-4	2.4e-4

Table 16 shows the goodness-of-fit for session length distribution (the bin size is 1 minute). Using a common significant level $P_0 = 5\%$, we see that in all cases the two session length distributions (synthetic trace and real trace) are *statistically* the same. In addition, multi-class population model yields smaller goodness-of-fit score, indicating that it can fit the real trace better than the single-class population model.

Aggregate event rate: Figure 48 shows the aggregate event rate of different types of events. Again, we observe a reasonably good match between the model and real trace from visual inspection. Table 17 shows the root mean square error (RMSE) between the model and real trace when modeling the time-varying rate. We find that the RMSE of multi-class population model is an order of magnitude smaller that that of single-class population model. For example, the RMSE of modeling switch-on events by using single-class population model is 2.3e-3 where the corresponding figure by using multi-class population model is 1.8e-4. The reduction in RMSE is due to the fact that much more (triple) parameters are used in the multi-class population model (i.e., more details have been modeled). Depending on the applications of SimulWatch, this may or may not be a desirable property.

Channel popularity distribution: Figure 49 shows the channel popularity distributions for both the real trace and synthetic trace. As expected, the popularity of top 600 channels matches very well. Table 16 shows the goodness-of-fit for channel

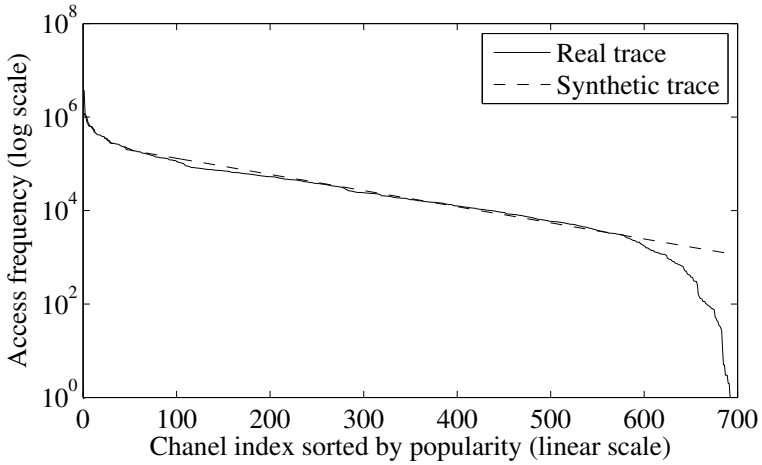


Figure 49: Channel popularity distributions for the real trace and the generated workload.

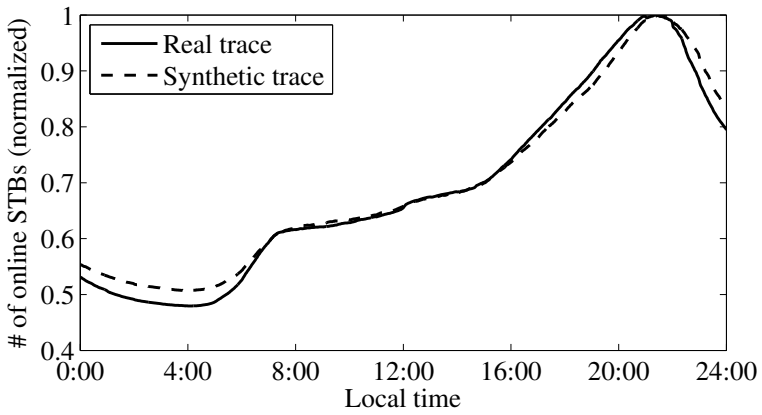


Figure 50: Number of on-line STBs over time. The results from real trace and workload closely match.

popularity distribution (the bin size is 1 channel). Again, we observe that the multi-class population model yields smaller goodness-of-fit score, indicating that it can fit the real trace better than the single-class population model.

5.4.2.2 Properties not explicitly modeled

Number of on-line STBs: Figure 50 shows the average number of on-line STBs as a function of the time-of-day. We normalize both synthetic trace and real trace such that the value at their peak time is 1. This is a property that we do not model directly, however, we still find a decent match in their shape.

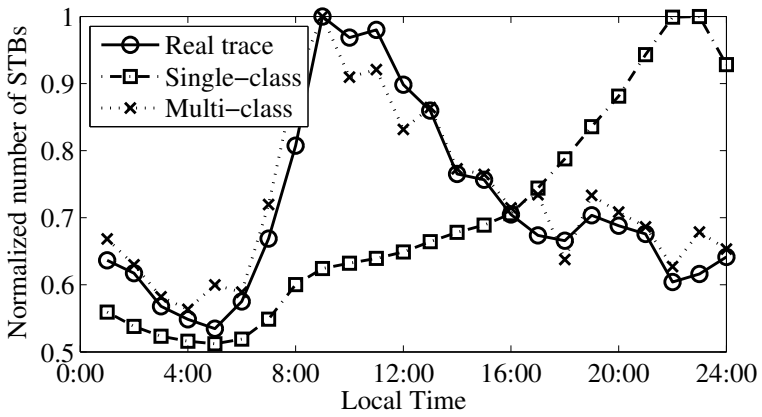


Figure 51: Multi-class population model captures the change of channel popularity over time (hourly)

Channel popularity dynamics: Recall that in Section 5.2 we showed that the channel popularity distribution changes over time within a single day. Taking a kids channel as an example, we show the change of channel popularity in Figure 51. The change is normalized such that the maximum value is 1. We observe that the strength of using multi-class population model is that this model can capture the dynamics of channel popularity very well, while single-class population model fails. Note that the curve for the single-class population model is actually similar to the curve of online STBs over time in Figure 50 because the single-class population model treats every channel-switch event uniformly. Therefore, the changing rate of channel popularity is proportional to the changing rate of on-line STB population.

To better understand the reasons that a multi-class population model can better capture the dynamics of channel popularity, we drill down the above example. There are two factors which shape the popularity dynamics of a particular channel: i) the channel access probability defined as the number of STBs in a particular population (for multi-class population model) watching that channel divided by the number of online STBs in that population at that time period (e.g. one hour). It is actually the transition probability of that channel; ii) the population mix which consists of the proportions of different subgroups in the whole user population. The channel

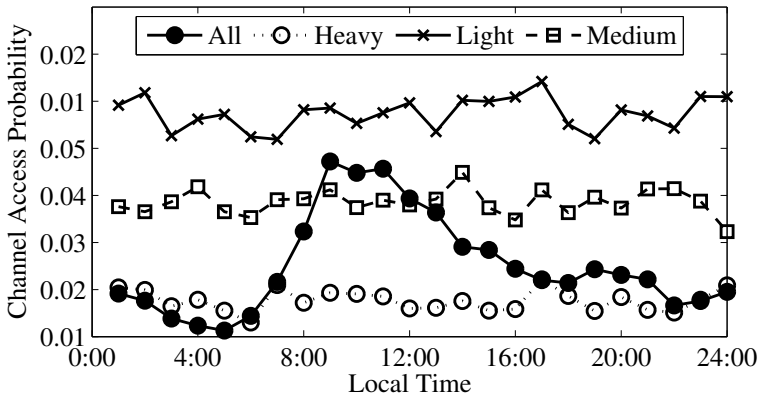


Figure 52: Time-of-day dynamics for a popular kids channel, based on multi-class synthetic trace

popularity dynamics is generated by the combinational effect (which can be viewed as the weighted sum of productions of subgroups) of these two factors.

Figure 52 shows the channel access probability of a popular kids channel from our multi-class synthetic trace. We display one curve for each group and an additional curve for all-STB cases (denoted by "All"). We observe that the curve for all-watchers shows a diurnal pattern, but all the other curves are quite stable. This means the change of the channel access probability in each group is very small and hence does not contribute to the dynamics of channel popularity in Figure 51. Figure 53 illustrates the population mix in our synthetic trace. By [42], the light-watcher group consists of the majority of watchers for kids channels. When we compare Figure 52 with Figure 53, it is clear that the increase bump (approximately 5:00 – 14:00) of light-watcher group in population mix contributes to the spike view of Figure 51.

5.4.2.3 Case Study

Since our ultimate goal is to use the synthetic trace from SimulWatch in evaluating the performance of different design of IPTV system, different system parameter settings, etc., we put SimulWatch to a final test by using it in a case study. In particular, we are interested in evaluating the bandwidth requirement to support

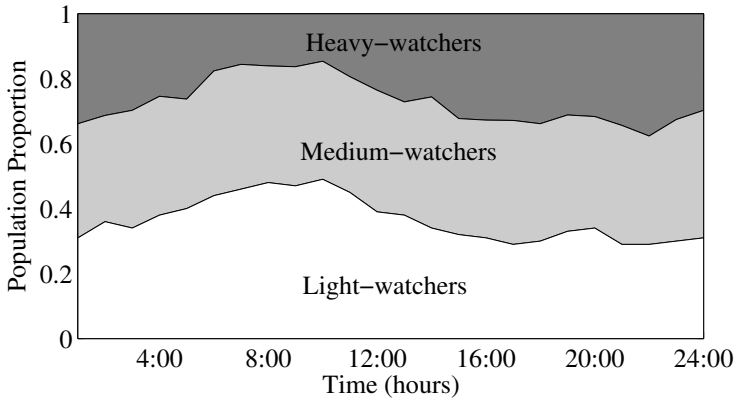
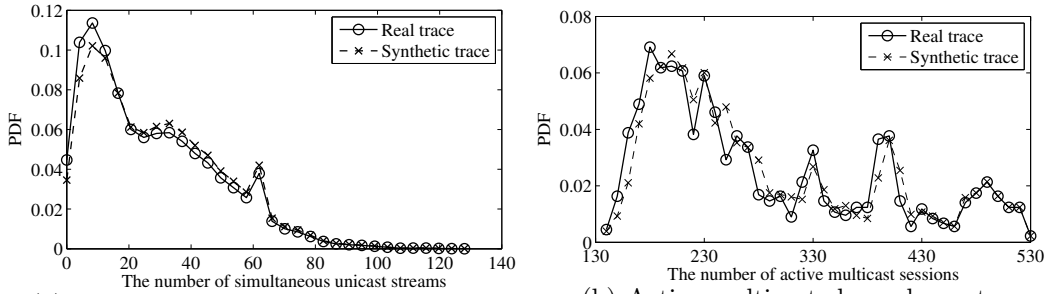


Figure 53: Population mix for each group, based on multi-class synthetic trace



(a) Simultaneous unicast stream count (b) Active multicast channel count

Figure 54: Case study. The results from real trace and workload closely match.

fast-channel-switch. Recall that using fast-channel-switch, a short (x seconds) unicast stream is transmitted to the STB (in addition to a new multicast stream) when a user switches to a new channel. We focus on a single router in one of the VHOs in the IPTV network, which connects to $2,137$ downstream STBs. We evaluate different value of x (4, 8, 16, 32). We only present result for $x = 32$ seconds, while other results are quantitatively very similar.

We study the number of simultaneous unicast streams flowing downstream from the router of interest under the above settings. Figure 54(a) shows the distribution density function on the number of concurrent unicast streams when using either real trace or synthetic trace in evaluation. We observe that the two curves closely match. Both curves show that for around 4% of time there is no unicast stream in the system; with a small probability, there can be demand for more than 80 concurrent unicast

streams, with the maximum being 128 in both cases; and interestingly there are two local peaks (at 10 and 60) in both distribution functions, the second of which may relate to the correlated channel switchings at hour boundaries. This result demonstrates that SimulWatch faithfully preserves the intrinsic characteristics of user activities that are essential to our evaluation.

We also examine the number of channels that these 2,137 STBs collectively request. This value translates to the amount of multicast traffic involved to support live TV viewing for the users. In Figure 54(b), we report the probability density function for the number of channels, in which we confirm that the result from synthetic trace closely matches that of real trace.

5.5 Related Work

Traditionally, understanding users' TV viewing activities in the conventional TV systems relied on phone surveys or specialized monitoring boxes (e.g., by Nielsen Media Research [3]). The challenge with that approach is the difficulty conducting a large-scale survey or deploying monitoring boxes for the majority of TV users. In this paper, we analyze user activity data from more than one million commercial IPTV subscribers and present models that can be used to generate realistic user activity workload.

Many researchers recently have looked into various aspects of IPTV systems. The closest work to our study is the recent measurement study conducted by Cha et al. [12]. While some of our findings overlap with their study, our focus is to *model the user activities* based on the measurement study and *design a workload generator*, which can be used to evaluate different aspects of IPTV system design and performance with respect to realistic user workload. In our earlier work [42], we extensively study one aspect of IPTV system: channel popularity. In this paper, we model a wider range of aspects of user activities, and design a workload generator. Smith [44]

analyzed bandwidth demand to support both multicast and unicast for fast channel change, where channel switching is modeled as a renewal process. However, the work is not based on actual traces, and such a study can benefit from our workload model and trace generator. Whereas our work focuses on IPTV services running on top of a provider backbone, there are a number of peer-to-peer (P2P) based IPTV systems [63, 32] and the measurement study focusing on P2P IPTV systems [23, 43].

Some researchers have investigated user activity workload in other context. For example, Costa et al. [16] analyzed user activities in media streaming applications. Cherkasova et al. [13], Chesire et al. [14], and Tang et al. [45] built models for the workload of media streaming service. Yu et al. [61] studied the user activities to access a Video-on-Demand (VoD) system. Cha et al. [11] explored how users access videos in the YouTube system. Guo et al. [21] compared access patterns of different types of media content on the Internet including Web, P2P, VoD, and live streaming. These studies are complimentary to our work in that as IPTV providers offer more interactive video streaming and VoD services, we also need to consider these aspects in the system design.

5.6 Summary and Future Work

In this chapter, we have performed an in-depth analysis on several intrinsic characteristics of user activities in large IPTV systems, including durations for on-, off- and channel-sessions, time-varying rates of switching-on, switching-off and channel switching events, and channel popularity. We have also developed a series of practical mathematical models to capture these characteristics. Furthermore, we construct the first IPTV user activity workload generation tool SimulWatch, which can generate synthetic yet realistic activity traces of a large number of IPTV users. All the derived models and the implementation of SimulWatch have been validated using real traces collected from a large nationwide IPTV provider in the United States.

In particular, we demonstrate that while not explicitly modeled, the estimation of unicast and multicast traffic demand based on SimulWatch trace closely matches the actual values from the real trace. We believe that SimulWatch will prove useful in many different aspects of IPTV system design and evaluation.

Our future work includes several extensions to our current model. We plan to include proper modeling for the use of advanced features in IPTV, such as PIP and DVR. We also expect that users' activities likely change over time as IPTV providers introduce more features. It would be also interesting to analyze such changes and evaluate the performance impact those new features impose on IPTV systems.

CHAPTER VI

CONCLUSION

Through this dissertation, we expect to demonstrate how to better utilize systems logs to manage a large-scale IPTV network. We study two types of system logs: router syslogs and STB logs. We present the methodologies to uncover the useful information buried inside the raw logs, and demonstrate their usefulness to a number of network management applications. We will conclude this dissertation by summarizing our contributions. Open problems in each of these topics can be found at the end of the corresponding chapters.

In Chapter 3, we design a SyslogDigest system that can automatically transform and compress such low-level minimally-structured syslog messages into meaningful and prioritized high-level network events, by exploring both temporal and spatial relationship among different logs. These events are three orders of magnitude fewer in number and have much better usability than raw syslog messages. We demonstrate that the output of SyslogDigest can be used in network troubleshooting, and network health monitoring and visualization of a IPTV network.

In Chapter 4, we focus on analyzing the channel popularity in the context of IPTV by analyzing the STB logs. In particular, we propose a series of models to capture two important aspects of channel popularity – the distribution and temporal dynamics of the channel popularity. Furthermore, we propose a method to identify subsets of user population with inherently different channel interest. We also validate our channel popularity model using real user channel access data from a commercial IPTV network.

In Chapter 5, we perform an in-depth study on several intrinsic characteristics of

IPTV user activities by analyzing STB logs collected from an operational nation-wide IPTV system. We further generalize the findings and developed a series of models for capturing both the probability distribution and time-dynamics of user activities. We then combine theses models to design an IPTV user activity workload generation tool, which takes a small number of input parameters to generate synthetic workload traces that mimic a set of real users watching IPTV. This tool can estimate the unicast and multicast traffic accurately, proving itself as a useful tool in driving the performance evaluation study in IPTV systems.

In this dissertation, we show that system logs can be better utilized in a large-scale IPTV network. We hope this work can inspire network operators to rethink the way of using system logs collected in a large-scale IP network. We also hope that the general methodologies we propose (e.g., temporal-spatial association rule mining) for system logs can be extended to other network measurement data sets.

REFERENCES

[1] "Emc lonix website." `http://www.emc.com/products/family/ionix-family.htm`, August 2011.

[2] "Ibm netcool website." `http://www-01.ibm.com/software/tivoli/welcome/netcool`, August 2011.

[3] "The Nielsen Company." `http://www.nielsenmedia.org`, August 2011.

[4] Agrawal, D., Beigi, M. S., Bisdikian, C., and Lee, K.-W., "Planning and Managing the IPTV Service Deployment," in *10th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 353–362, 2007.

[5] Agrawal, R., Imielinski, T., and Swami, A., "Mining association rules between sets of items in large databases," in *Proc. ACM SIGMOD*, 1993.

[6] Barford, P. and Crovella, M., "Generating representative web workloads for network and server performance evaluation," in *SIGMETRICS*, pp. 151–160, 1998.

[7] Barger, K. J.-A., "Mixtures of exponential distributions to describe the distribution of poisson means in estimating the number of unobserved classes," Master's thesis, Cornell University, 2006.

[8] Bradley, J., *Distribution-free statistical tests*. Prentice-Hall., 1968.

[9] Brauckhoff, D., Dimitropoulos, X., Wagner, A., and Salamatian, K., "Anomaly extraction in backbone networks using association rules," in *Proc. ACM IMC*, 2009.

[10] Brockwell, P. J. and Davis, R. A., *Introduction to Time Series and Forecasting*. Springer, 2002.

[11] Cha, M., Kwak, H., Rodriguez, P., Ahn, Y.-Y., and Moon, S., "I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System," in *Proceedings of ACM IMC*, 2007.

[12] Cha, M., Rodriguez, P., Crowcroft, J., Moon, S., and Amatrianin, X., "Watching Television Over an IP Network," in *Proceedings of ACM IMC*, 2008.

[13] Cherkasova, L. and Gupta, M., "Characterizing locality, evolution, and life span of accesses in enterprise media server workloads," in *NOSSDAV*, 2002.

[14] Chesire, M., Wolman, A., Voelker, G. M., and Levy, H. M., "Measurement and analysis of a streaming media workload," in *USITS*, pp. 1–12, 2001.

[15] Chu, J., Labonte, K., and Levine, B., "Availability and locality measurements of peer-to-peer file systems," 2002.

[16] Costa, C. P., Cunha, I. S., Vieira, A. B., Ramos, C. V., Rocha, M. M., Almeida, J. M., and Ribeiro-Neto, B. A., "Analyzing client interactivity in streaming media," in *WWW*, 2004.

[17] Doob, J. L., "The Brownian movement and stochastic equations," *Annals of Math*, vol. 40, no. 1, pp. 351–369, 1942.

[18] Eirinaki, M. and Vazirgiannis, M., "Web mining for web personalization," *ACM Transactions on Internet Technology (TOIT)*, vol. 3, no. 1, pp. 1–27, 2003.

[19] Facca, F. and Lanzi, P., "Mining interesting knowledge from weblogs: a survey," *Data & Knowledge Engineering*, vol. 53, no. 3, pp. 225–241, 2005.

[20] Gerhards, R. and GmbH, A., "The Syslog Protocol," in *IETF RFC*, 2009.

[21] Guo, L., Tan, E., Chen, S., Xiao, Z., and Zhang, X., "The stretched exponential distribution of internet media access patterns," in *PODC*, pp. 283–294, 2008.

[22] Harrison, C. and Amento, B., "CollaboraTV: Using Asynchronous Communication to Make TV Social Again," in *EuroITV*, 2007.

[23] Hei, X., Liang, C., Liang, J., Liu, Y., and Ross, K. W., "A measurement study of a large-scale p2p iptv system," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1672–1687, 2007.

[24] Huang, Y., Fu, T. Z. J., Chiu, D.-M., Lui, J. C. S., and Huang, C., "Challenges, Design and Analysis of a Large-scale P2P-VoD System," in *Proc. ACM SIGCOMM*, 2008.

[25] Huang, Y., Feamster, N., Lakhina, A., and Xu, J. J., "Diagnosing network disruptions with network-wide analysis," *SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 1, pp. 61–72, 2007.

[26] Jewell, N. P., "Mixtures of Exponential Distributions," in *Annuals of Statistics*, 1982.

[27] Kalmaneka, C. R., Ge, Z., Lee, S., Lund, C., Pei, D., Seidel, J., der Merwe, J. V., and Yates, J., "Darkstar: Using exploratory data mining to raise the bar on network reliability and performance," in *Proc. the 7th international workshop on Design of Reliable Communication Networks (DRCN)*, October 2009.

[28] Kandula, S., Chandra, R., and Katabi, D., "Whats going on? learning communication rules in edge networks," in *Proc. ACM SIGCOMM*, 2008.

[29] Kandula, S., Mahajan, R., Verkaik, P., Agarwal, S., Padhye, J., and Bahl, P., "Detailed diagnosis in enterprise networks," in *Proc. ACM SIGCOMM*, 2009.

[30] Kompella, R. R., Yates, J., Greenberg, A., and Snoeren., A. C., "Detection and localization of network blackholes.," in *Proc. INFOCOM*, 2007.

[31] Lakhina, A., Crovella, M., and Diot, C., "Mining anomalies using traffic feature distributions," in *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, (New York, NY, USA), pp. 217–228, ACM, 2005.

[32] Liao, X., Jin, H., Liu, Y., Ni, L. M., and Deng, D., "Anysee: Peer-to-peer live streaming," in *INFOCOM*, 2006.

[33] Lim, C., Singh, N., and Yajnik, S., "A log mining approach to failure analysis of enterprise telephony systems," in *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, pp. 398–403, IEEE.

[34] MacQueen, J. B., "Some methods for classification and analysis of multivariate observations,," in *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297, 1967.

[35] Mahimkar, A., Ge, Z., , Shaikh, A., Yates, J. W. J., Zhang, Y., , and Zhao, Q., "Towards automated performance diagnosis in a large iptv network," in *Proc. ACM SIGCOMM*, 2009.

[36] Mahimkar, A., Yates, J., Zhang, Y., Shaikh, A., Wang, J., Ge, Z., and Ee, C. T., "Troubleshooting chronic conditions in large ip networks.," in *Proc. ACM CoNEXT*, 2008.

[37] Mahimkar, A., Song, H. H., Ge, Z., Shaikh, A., Wang, J., Yates, J., Zhang, Y., and Emmons, J., "Detecting the performance impact of upgrades in large operational networks," in *Proc. ACM SIGCOMM*, 2010.

[38] Nielsen, J., "Zipf curves and website popularity, www.useit.com/alertbox/zipf.html," 1997.

[39] Oliner, A. and Stearley, J., "What supercomputers say: A study of five system logs," 2007.

[40] Perkowitz, M. and Etzioni, O., "Adaptive web sites," *Communications of the ACM*, vol. 43, no. 8, pp. 152–158, 2000.

[41] Plonka, D., "Flowscan: A network traffic flow reporting and visualization tool.," in *Proc. USENIX System Admin. Conf.,*, 2000.

[42] Qiu, T., Ge, Z., Lee, S., Wang, J., Zhao, Q., and Xu, J. J., "Modeling Channel Popularity Dynamics in a Large IPTV System," in *SIGMETRICS*, 2009.

[43] Silverston, T., Fourmaux, O., Salamatian, K., and Cho, K., "Measuring p2p iptv traffic on both sides of the world," in *CoNEXT*, p. 39, 2007.

[44] Smith, D. E., "IPTV Bandwidth Demand: Multicast and Channel Surfing," in *INFOCOM*, pp. 2546–2550, 2007.

[45] Tang, W., Fu, Y., Cherkasova, L., and Vahdat, A., "Medisyn: a synthetic streaming media service workload generator," in *NOSSDAV '03*, pp. 12–21, 2003.

[46] Tariq, M., Zeitoun, A., Valancius, V., Feamster, N., and Ammar., M., "Answering what-if deployment and configuration questions with wise.," in *Proc. SIGCOM*, 2008.

[47] Uhlenbeck, G. and Ornstein, L., "On the Theory of Brownian Motion," *Physical Review*, September 1930.

[48] Weber, J. and Gong, J., "Modeling switched video broadcast services," in *Cable Labs*, 2003.

[49] Willinger, W., Taqqu, M., Sherman, R., and Wilson, D., "Self-similarity through high-variability: statistical analysis of ethernet lan traffic at the source level," *IEEE/ACM Transactions on Networking (ToN)*, vol. 5, no. 1, pp. 71–86, 1997.

[50] Won, Y. J., Choi, M.-J., Park, B.-C., Lee, H.-W., Hwang, C.-K., and Yoo, J.-H., "End-user iptv traffic measurement of residential broadband access networks," in *NOMS Workshops 2008*, 2008.

[51] Xiao, Y., Du, X., Zhang, J., Hu, F., and Guizani, S., "Internet protocol television (iptv): The killer application for the next-generation internet," *Communications Magazine, IEEE*, vol. 45, no. 11, pp. 126–134, 2007.

[52] Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G., and Osipkov, I., "Spamming botnets: Signatures and characteristics," in *Proc. ACM SIGCOMM*, 2008.

[53] Xu, W., Huang, L., Fox, A., Patterson, D., and Jordan, M., "Mining console logs for large-scale system problem detection," in *Proceedings of the Third conference on Tackling computer systems problems with machine learning techniques*, pp. 4–4, USENIX Association, 2008.

[54] Xu, W., Huang, L., Fox, A., Patterson, D., and Jordan, M., "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pp. 117–132, ACM, 2009.

[55] Xu, W., Huang, L., Fox, A., Patterson, D., and Jordan, M. I., "Detecting large-scale system problems by mining console logs," in *SOSP '09: Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, (New York, NY, USA), pp. 117–132, ACM, 2009.

[56] Yamanishi, K. and Maruyama, Y., "Dynamic syslog mining for network failure monitoring," in *Proc. ACM KDD*, August 2005.

[57] Yang, Q., Zhang, H., and Li, T., "Mining web logs for prediction models in www caching and prefetching," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 473–478, ACM, 2001.

[58] Yang, Y., "Expert network: effective and efficient learning from human decisions in text categorization and retrieval," in *SIGIR 94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, NY, USA), pp. 13–22, Springer-Verlag New York, Inc., 1994.

[59] Ye, N., Vilbert, S., and Chen, Q., "Computer intrusion detection through EWMA for autocorrelated and uncorrelated data," in *IEEE transactions on reliability*, October 2003.

[60] Yu, H., Zheng, D., Zhao, B. Y., and Zheng, W., "Understanding user behavior in large-scale video-on-demand systems," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 4, pp. 333–344, 2006.

[61] Yu, H., Zheng, D., Zhao, B. Y., and Zheng, W., "Understanding user behavior in large-scale video-on-demand systems," in *EuroSys*, pp. 333–344, 2006.

[62] Za
"iane, O., Xin, M., and Han, J., "Discovering web access patterns and trends by applying olap and data mining technology on web logs," in *Research and Technology Advances in Digital Libraries, 1998. ADL 98. Proceedings. IEEE International Forum on*, pp. 19–29, IEEE, 1998.

[63] Zhang, X., Liu, J., Li, B., and Yum, T.-S. P., "Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming," in *INFOCOM*, pp. 2102–2111, 2005.

[64] Zhang, Z. and Nasraoui, O., "Mining search engine query logs for query recommendation," in *Proceedings of the 15th international conference on World Wide Web*, pp. 1039–1040, ACM, 2006.

[65] Zhao, X., Massey, D., Lad, M., and Zhang, L., "On/off model: a new tool to understand bgp update burst," tech. rep., USC/CS Technical Report 04-819, 2004.

VITA

Tongqing Qiu received his Bachelor of Science and Master of Engineering both in computer science and engineering from Nanjing University, Nanjing, China in 2000 and 2004 respectively. Tongqing joined the College of Computing at the Georgia Institute of Technology, as a Ph.D. student in August 2007. His thesis work was conducted under the able guidance of Dr. Jun (Jim) Xu.