BAYESIAN METHODS FOR NETWORK TRAFFIC ANALYSIS

by

Barış Kurt

B.S., Computer Engineering, Boğaziçi University, 2005

M.S., Computer Engineering, Boğaziçi University, 2009

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2019

To my wife Aylin and my son Poyraz ...

ACKNOWLEDGEMENTS

I would like to thank my beloved wife Aylin for her unconditional support and guidance in this long journey that ended up with this thesis. I cannot thank enough my father Şadi and mother Aynur for their support throughout my education. I thank my sister Pelin and sister-in-law Güneş for their precious motivational speeches.

I am grateful to my PhD supervisor Ali Taylan Cemgil not only for his academic guidance but also for being an overall mentor and introducing me to running. Bülent Sankur always inspired me with his wisdom, unlimited energy and sense of humor. I would like to thank Güneş Karabulut Kurt for her guidance and collaborations; Cem Ersoy for always being helpful and constructive, and Suzan Üsküdarlı for being a great listener and mentor.

I worked together with many smart and fun people during my PhD. I would like to thank Çağatay Yıldız and Yusuf Taha Ceritli for their great friendship and efforts in our joint works. I would like to thank Orhan Sönmez, Beyza Ermiş, Barış Evrim Demiröz, Murat Semerci, Umut Şimşekli, Alp Kındıroğlu, Serhan Daniş, Yunus Emre Kara Gaye Genç Kara and Ömer Deniz Akyıldız for their companionship in this journey. And, I will always remember İsmail Arı with his diligence, thoughtfulness and kindness.

ABSTRACT

BAYESIAN METHODS FOR NETWORK TRAFFIC ANALYSIS

Statistical information about traffic patterns help a service provider to characterize its network resource usage and user behavior, infer future traffic demands, detect traffic and usage anomalies, and possibly provide insights to improve the performance of the network. However, the increasingly high volume and speed of data over modern networks make collecting these statistics difficult. Moreover, smarter network attacks require sophisticated detection methods that are able to fuse many network and hardware signals. Fortunately, Bayesian statistical methods are powerful tools that can infer such information under the harsh network environments.

In this thesis we apply two Bayesian methods for two specific network problems. First, we use the Bayesian multiple change models to detect DDoS attacks in SIP networks by fusing the observations coming from the network traffic and the networking hardware. We show that our method is superior to classic DDoS detection methods and using hardware signals improve the detection rate. For this work, we developed a probabilistic SIP network simulator and a monitoring system, and published it as an open-source software.

In our second work, we estimated network statistics from a high speed network where we can only observe a fraction of the network traffic. For this problem we develop a generic novel method called ThinNTF, based on non-negative tensor factorization. This method can work with different network sampling schemes and recovers original network statistics by detecting the periodic network traffic patterns from the sampled network data and gives better estimates compared to the state of the art.

ÖZET

AĞ TRAFİĞİ ANALİZİ İÇİN BAYESÇİ METODLAR

Trafik örüntüleri hakkındaki istatistiksel veriler, ağ sağlayıcılarına ağlarındaki kaynak kullanımı ve kullanıcı davranışlarını nitelemek, gelecekteki ağ gereksinimlerini kestirmek, ağ olağandışılıklarını sezmek ve başarımı iyileştirmek konularında yardımcı olurlar. Bununla birlikte, günümüz ağlarında gittikçe artan yüksek veri hacmi ve hızı bu istatistiklerin elde edilmesini güçlendirmektedir. Dahası, akıllı ağ saldırıları ağ trafiği ve ağ donanımından gelen sinyalleri birleştirebilen, ileri tespit yöntemlerine ihtiyaç duymaktadır. Neyse ki, Bayesçi istatistiksel yöntemler zorlu ağ ortamlarında bu verileri elde edebilecek araçlardır.

Bu tezde iki farklı Bayesçi yöntemi farklı problemlere uyguladık. İlkinde hem ağ trafiği hem de ağ donanım verisi kullanan bir Bayesçi çoklu değişim noktası modeli kullanarak SIP ağlarındaki DDoS saldırılarını yakaladık. Yöntemimizin diğer DDoS tespit metodlarından daha iyi olduğunu ve ağ donanım bilgisini kullanmanın başarımı arttırdığını gösterdik. Bu çalışma için olasılıksal bir SIP ağı benzetime ve gözetleme sistemi geliştirdik ve açık kaynak kodlu olarak yayınladık.

İkinci çalışmamızda sadece bir kesmini gözlemleyebildiğimiz hızlı bir ağ üzerindeki ağ istatistiklerini kestirdik. Bu problem için ThinNTF adını verdiğimiz, negatif olmayan tensor ayrışımı tabanlı genel bir yöntem geliştirdik. Bu yöntem farklı ağ trafiği örnekleme şemalarıyla birlikte kullanılabilmekte ve örneklenmiş veriden dönemsel ağ istatistiklerini çıkartarak asıl istatistikler elde etmekte ve bu ilave bilgiyi kullanmatan yöntemlerden daha başarılı sonuç vermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a	Shape parameter of Gamma distribution.
b	Scale parameter of Gamma distribution.
$\mathbf{D}$	Day-of-week matrix in $\mathbb{R}$.
$\mathbf{F}$	Flow length matrix in $\mathbb{R}$.
$\mathbf{H}$	Hour-of-day matrix in $\mathbb{R}$.
$\mathcal{M}$	Binary mask tensor.
$\mathbf{r}$	Reset switches.
$\mathbf{S}$	Sampling matrix in $\mathbb{R}$.
$\mathcal{W}$	Latent variable tensor $\mathbb{R}$.
$\mathcal{X}$	Original flow length tensor in $\mathbb{Z}$.
$\mathcal{Y}$	Sampled flow length tensor in $\mathbb{Z}$.
α	Parameter of Dirichlet distribution.
δ	Dirac delta function.
π	Reset probability.
$\langle . \rangle_Q$	Expectation under Q distribution.

LIST OF ACRONYMS/ABBREVIATIONS

3G	Third Generation Mobile Communication Technology
4G	Fourth Generation Mobile Communication Technology
DDOS	Distributed Denial of Service
EM	Expectation-Maximization
MAP	Maximum a Posteriori
ML	Maximum Likelihood
NMF	Nonnegative Matrix Factorization
NTF	Nonnegative Tensor Factorization
ThinNTF	Thin Nonnegative Tensor Factorization
LTE	Long Term Evaluation
IP	Internet Protocol

1. INTRODUCTION

A computer network is a set of computers linked together. Although this simple definition includes just two entities, computers and their connections, networking is a complicated branch of computer science and engineering. As this thesis is being written, there are more than 2 billion personal computers and 2.6 billion smart phones in the world, all of which are interconnected thanks to the developments on mobile networking and it is estimated that 3.6 billion people are using the internet. This sheer volume and complexity introduces many challenges in network management. The network service providers have to spend tremendous efforts to maintain high quality of service, improve their infrastructure and provide network security. These services require real time monitoring of high-speed and high-volume network links, and fast processing of large amount of data. In this thesis we propose Bayesian methods for several network traffic analysis problems, which address some of issues.

New generation wireless technologies enable operators to provide broadband coverage. Especially with the introduction of LTE and smart phones, network management for data traffic is becoming a harder problem everyday. Data traffic is increasing rapidly and network operators cannot respond fast enough to demands for the capacity increase. The expectation is that demands on the infrastructure will be comparable or even exceed the current utilization of conventional fixed broadband connections. The trend is already clear, data transmitted in networks for mobile users is increasing fast and the operators need to find ways to reduce their investment per traffic. It is therefore more important than ever before to observe the network utilization and take necessary actions in terms of maintaining QoS per application, hence optimize the network usage for improved customer satisfaction and still remain profitable.

1.1. Description of the Network Data

In order to better understand the scope of our thesis, we first need to make a description of the network data that we are dealing with. Since networking is a

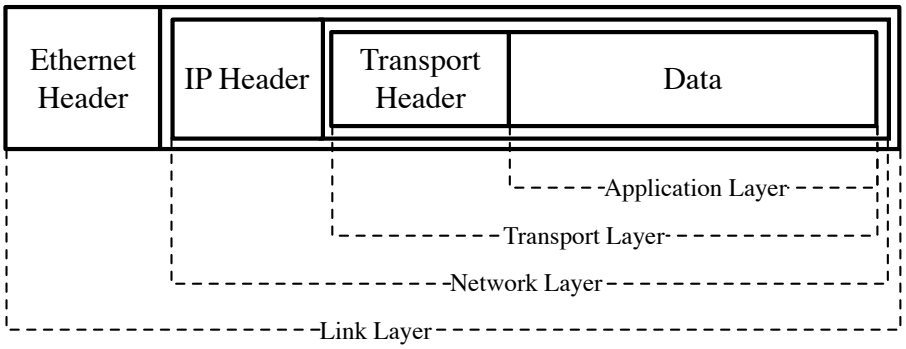


Figure 1.1. Structure of an ethernet frame.

broad term and network data can have different meaning from different perspectives, we need to narrow down its definition. We simply define the *network traffic* as data flowing between communicating devices on the network at a given time. In this thesis, we are focused on the most famous networking system, the *Internet*. The Internet is implemented by the Internet Protocol Stack, where the user data is transferred in small chunks which are called *packets*. The connections inside the network is handled by the *packet switching* technique, and the packets find their ways between the devices via *routing*. In this thesis we are not interesting in the underlying topology of the network or the routing information of the packets. Instead, we monitor the incoming and outgoing packets from an observation point, that is mainly a server.

In the internet, the user data is encapsulated by a stack of protocols. Figure 1.1 shows the stacked structure of a IP packet, or more formally, an *Ethernet frame*. In this stack structure, the user data is carried inside a transport segment, which is routed by an IP datagram, inside an Ethernet frame.

As we monitor the network, we extract several information fields from Ethernet packets, and ignore the rest. We are mainly interested in the information such as the IP addresses of the sender (source) and receiver (destination) hosts found in the IP header, and their corresponding port numbers found in the transport header. An IP address and a port number defines a *socket address* in the related host. Additionally. the transport protocol is also of special interest. As transport layer protocols, we will be dealing only with transport control protocol (TCP) and user datagram protocol

Table 1.1. Abstraction of a network packet.

Key	Source IP
	Source Port
	Destination IP
	Destination Port
	Protocol
Attributes	Arrival Time: a
	Length : l
	Application Features

(UDP). Other main packet features include the length and the arrival time of the packet. Depending on the problem we may also extract application level features. For example in Chapter 3, we will be mainly dealing with Session Initiation Protocol (SIP) headers. Table 1.1 lists the primary information we extract from a network packet.

While a packet is a first level network entity, we can define a network flow, which is a collection of packets, as the second level network entity. Network flows are basically a flow of data between two sockets in the application layer. If we consider a packet as a message, a network flow can be considered as a communication between two applications. More formally, we can define a network flow as an ordered set of packets exchanged between two sockets. The packets in this set share a unique key which is composed a source IP, source port, destination IP, destination port and the transfer level protocol. This 5-tuple distinctly identify a network flow.

Since a network flow defines a communication, the packets that belong to flow can travel in either direction, from source to destination or from destination to source. When used for network flows, source and destination keywords have slightly different meanings. The source of a network flow is actually the source that generated the first packet of the flow. In other words, the source IP and source port of a network flow is the source IP and source port the first packet. For packets traveling backwards, the source and destination IP and port pairs would be interchanged. Therefore, in order to find out whether a packet belongs to a flow, we need to check for both scenarios by

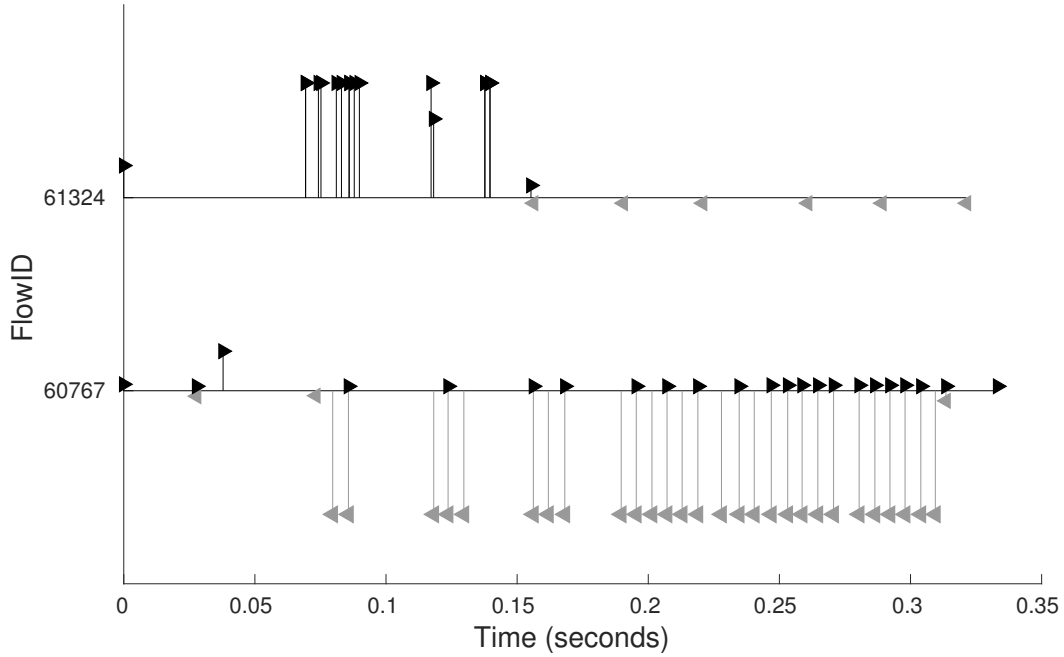


Figure 1.2. Example flows.

creating two different keys, one for forward direction, one for backward direction.

Finally, a flow is said to be terminated whenever a timeout period is passed without any packet generated by the communicating sockets. A typical timeout period is 1 minute. Additionally, we can detect the termination of a TCP flow by observing a TCP packet with its BYE flag set. Figure 1.2 shows a representation for 2 network flows captured from the live traffic. Horizontal axis denotes the arrival time of packets in seconds. Each vertical line length is proportional to a packet size and the direction denotes up (dark ▶) and down (light ◁) stream packets during the communication. In the figure, the top flow consists of large packets sent to the destination, and the bottom flow has mainly large packets received.

Mathematically, we represent a flow f_n as the set $f_n = \{t_n, c_n, \mathbf{a}_n, \mathbf{d}_n, \mathbf{l}_n\}$ where t_n is the number of packets in the flow, which we call the *flow size* and c_n is the category of the application generating the flow. The remaining $\mathbf{a}_n, \mathbf{d}_n$ and $\mathbf{l}_n$ are the vectors, each of length t_n. The $\mathbf{a}_n$ vector contains arrival times of packets to the destination socket, in epoch time format. The $\mathbf{d}_n$ vector contains the directions of each packet. We define the packet directions as $+1$, or *up*, if the packet send from source to destination socket,

Table 1.2. Abstraction of a network flow.

Key	Source IP
	Source Port
	Destination IP
	Destination Port
	Protocol
Attributes	Size : t_n
	Length : $\sum_i l_i$
	Packet Info 1 : (a_1, d_1, l_1)
	$\vdots$
	Packet Info t_n : $(a_{t_n}, d_{t_n}, l_{t_n})$

and -1 , or *down* it is send from destination to the source. The source and destination hosts are assigned according to the related fields of the first packet. Finally, the $\mathbf{l}_n$ vector contains the length of each packet, which is the total number of bits including all headers in the Ethernet frame. Table 1.2 lists the information we extract from a network flow.

1.2. Network Measurement and Sampling

Statistical information about traffic patterns help a service provider to characterize its network resource usage and user behavior, infer future traffic demands, detect traffic/usage anomalies, and possibly provide insights to improve the performance of the network [1]. Passive measurement, where the measuring beacons inactively watch the traffic passing by [2] is a popular method for collecting network statistics such as per flow information. However, constantly increasing link speeds and traffic volume makes passive measurement challenging. While high link speeds restricts the time spent on processing individual packets, high traffic volume requires larger memory to store the necessary information. Therefore, additional techniques such as using specialized hardware [3–5] and/or sampling only a fraction of network packets [6–8] have been proposed.

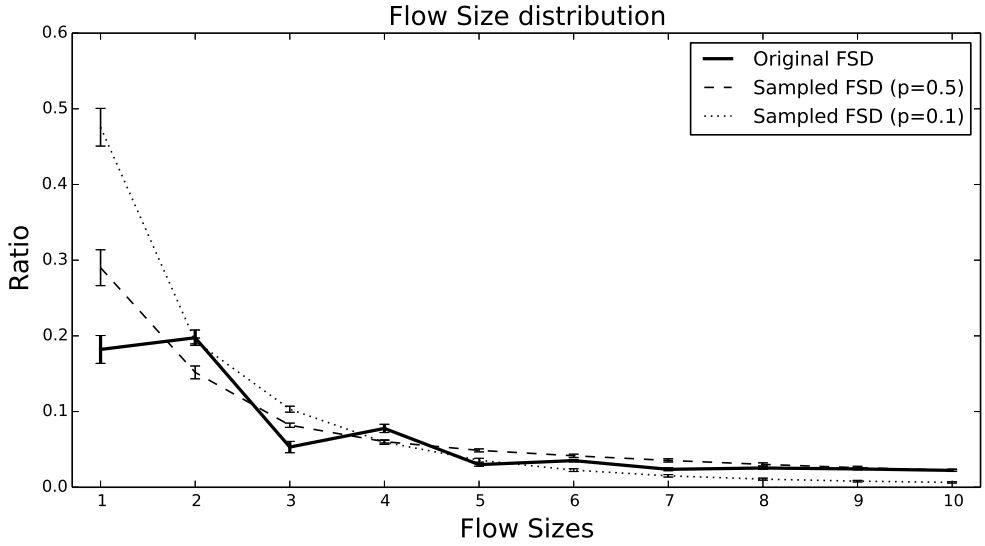


Figure 1.3. Flow size distribution measurement problem.

In this thesis, we focus on real-time tracking of the flow size distribution on a high speed network of an Internet Service Provider (ISP) on a Commercial-off-the-shelf (COTS) server. In this set up, without the availability of a dedicated hardware, we employ random packet sampling technique. We define the number of packets inside a flow as the *flow length* and the histogram of flow lengths of the active flows at a given time instance as the *flow length distribution* where an active flow is a flow which has not been terminated yet. The main problem with the packet sampling technique is that, whenever sampling is applied, the collected statistics differ from their original values as depicted in Figure 1.3. The figure show real world statistics collected from the servers of an ISP. The mean length ratios of flows of length up to 10 are presented when all packets are observed and when random packet sampling applied with sampling probabilities 0.5 and 0.1. Hence, a post-processing must be done in order to recover the original statistics after the sampling.

There are several proposed methods for restoring the original flow size distributions for the random packet sampling scenario. In this thesis, we propose improvements to the non-parametric flow length models in [6] and [8] where the network traffic is modeled as a mixture of several traffic patterns. We observe that such traffic patterns can be extracted by employing the nonnegative matrix factorization (NMF) model. Fur-

thermore, we extract those patterns directly from the sampled data via our modified version of the NMF algorithm, which we call which we call ThinNTF, and also recover the original flow size distributions. The Chapter 4 is dedicated to the measurement problem and includes our methodology as well as the results.

1.3. DDoS Attack Detection in SIP Networks

Voice over IP (VoIP) is the technology of carrying voice and multimedia communications through the internet protocol (IP) networks, such as the internet. Due to its low infrastructure cost and multimedia support, VoIP systems have been taking over circuit-switched telephone networks, worldwide. The VoIP systems transfer audio and multimedia data between communicating parties, through the packet-switched IP networks via data transfer protocols, such as the Real-time Transport Protocol (RTP). However, they require signaling protocols for managing their communication sessions. Due to its lightweight nature, simplicity and ease of implementation, the SIP [9] is one of the most popular open standard signaling protocols designed for VoIP. SIP provides signaling functions necessary to register clients, check their locations and availability, exchange information on their data transmission capabilities, and provide handshake for their conversations.

Despite all their attractive features, the downside is that VoIP systems are more vulnerable to security threats compared to their circuit switched predecessors. There are two basic sources of security threats for VoIP systems. Firstly, VoIP systems are affected by all the IP network threats. Secondly, they are prone to security threats specifically designed to exploit the vulnerabilities of the underlying signaling protocols [10]. These protocol-specific attacks are usually not classified as network attacks, hence they are not detected by the conventional network security systems. Therefore, VoIP systems need extra security mechanisms for detecting and preventing VoIP specific attacks.

Figure 1.4. A histogram of SIP messages before and after a DDoS attack.

One of the most frequently observed type of cyber-attack is the Distributed Denial of Servie (DDoS) flooding attack [11], which is typically realized by sending a vast amount of network messages to a victim. In this thesis, we focus on the detection of SIP-specific DDoS flooding attacks [10, 12]. Figure 1.4 shows histogram of SIP messages observed on a SIP server for a duration of 35 seconds. The first 22 seconds of the histogram shows the ratios of SIP messages under the normal traffic, where the last 13 seconds show the SIP message ratio under a DDoS attack executed by flooding the server with INVITE messages.

In our work, we employ a Bayesian change point model to model the normal behaviour of the SIP network traffic and set alarms where this behaviour deviates from the normal. We also take into account several other features collected from the SIP serve such as server resource utilization and histogram of log messages. Chapter 3 includes the details of our DDoS detection system together with the model, experiment setup and results.

1.4. Contributions

- We developed a network monitoring system for high speed network traffic [13] and collected collected large amount of real time traffic from a Global System for Mobile Communications (GSM) network.
- We developed a novel generic tensor factorization algorithm, called *ThinNTF* [14] that can detect periodic traffic behavior from network traffic sampled with any sampling algorithm, provided that it is expressed as matrix operation.
- We developed a Bayesian change point model [15–17], for detecting SIP-oriented DDoS attacks. The proposed framework extends and generalizes the previous change-point based detection methods. Our change point-based DDoS monitor can be customized with different server parameters and different probabilistic observation models.
- A real-time SIP network traffic simulator [18, 19] based on social network modeling is developed and the software made publicly available. The proposed framework is tested with real-time data generated by the simulator, interleaved with DDoS attack data generated by a commercial network vulnerability scanning tool.

1.5. Organization of the Thesis

The thesis is organized as follows. In Chapter 2 we give background information on Bayesian machine learning methods that will be employed for the solutions to the problems defined in subsequent chapters. In Chapter 3 we solve the DDoS detection problem in SIP networks using Bayesian change point model. In Chapter 4 we employ our ThinNTF model for the real-time network statistic recovery from sampled network data. In Chapter 5 we conclude the thesis.

2. METHODOLOGY

In this chapter we provide the basic Bayesian methodology and models used throughout the thesis. While we give general descriptions for the algorithms and models, how these model are tailored for network problems is going to be explained in later chapters. Before going further, we should explain our mathematical notation.

For a clear notation, the scalar values are denoted by lightface letters, such as the index variable j and its maximum value J. The vectors are represented by boldface lower case letters, such as vector $\mathbf{x}$. Boldface upper case letters represent matrices, such as $\mathbf{F}, \mathbf{H}$ and $\mathbf{D}$, and the tensors are represented with calligraphic upper case letters i.e $\mathcal{X}$. The individual entries in matrices and tensors are written like scalars, i.e. $f_{i,r}$ and $x_{i,j,k}$. The index : denotes all the entries in the given dimension. For example $s_{i,:}$ is the i^{th} row of the $\mathbf{S}$ matrix and $\mathcal{X}_{i,:,:}$ is the i^{th} slice of the tensor $\mathcal{X}$ in the first dimension. We use superscripts to denote the index of an object inside a list. For example $\mathbf{x}^{(1:N)}$ presents a list of N vectors and $\mathbf{x}^{(n)}$ shows the n^{th} vector in the list.

2.1. Expectation-Maximization and Variational Inference

Suppose a probabilistic generative model M with parameters θ generates observations $\mathbf{X}$, such that an observation $\mathbf{x}$ depends on some other random but unobserved variable $\mathbf{z}$ which has also been sampled during the generative process. We call these variables the *latent variables*, denoted as $\mathbf{Z}$. Then the likelihood of an observed set $\mathbf{X}$ can be written as

$$p(\mathbf{X}|\theta) = \int_{\mathbf{Z}} p(\mathbf{X}|\mathbf{Z}, \theta) p(\mathbf{Z}|\theta) d\mathbf{Z}$$

The introduction of the latent variables makes the calculation of the likelihood cumbersome, or sometime practically impossible. For example, for discrete latent variables, the integration requires considering all combinations of latent variables, which may be in exponential numbers. For continuous variables, the overall integral may not be well

formed. In such cases, a general framework called Expected-Maximization (EM) can be utilized to maximize the likelihood with respect to the parameters θ.

The EM [20] is an iterative algorithm that maximizes a lower bound of the log-likelihood of parameters. The lower bound is defined by the help of the Jensen's Inequality which states the following

For $\lambda_i \in [0, 1]$ and $\sum_i \lambda_i = 1$, and a convex function f , the following holds

$$f\left(\sum_i \lambda_i x_i\right) \leq \sum_i \lambda_i f(x_i) \tag{2.1}$$

In probability terms, we can write this inequality as

$$f(E[\mathbf{x}]) \leq E[f(\mathbf{x})] \tag{2.2}$$

We bound the log-likelihood $\log p(\mathbf{X}|\theta)$, using the Jensen's inequality, since $-\log$ is a convex function, as follows

$$\log p(\mathbf{X}|\theta) = \log \int_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta) d\mathbf{Z} \tag{2.3}$$

$$= \log \int_{\mathbf{Z}} q(\mathbf{Z}) \frac{p(\mathbf{X}, \mathbf{Z}|\theta)}{q(\mathbf{Z})} d\mathbf{Z} \tag{2.4}$$

$$\geq \int_{\mathbf{Z}} q(\mathbf{Z}) \log p(\mathbf{X}, \mathbf{Z}|\theta) d\mathbf{Z} - \int_{\mathbf{Z}} q(\mathbf{Z}) \log q(\mathbf{Z}) d\mathbf{Z} \tag{2.5}$$

where $q(\mathbf{Z})$ is an instrumental distribution and $H_{q(\mathbf{Z})}$ is its entropy. This bound is a function of $q(\mathbf{Z})$ and is maximum when

$$q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \theta) \tag{2.6}$$

The EM iterative updates θ, where at each iteration θ^{new} is calculated by setting $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \theta^{old})$. Therefore, we can re-write the lower bound as

$$\mathcal{L}(q, \theta) = \underbrace{\int_{\mathbf{Z}} \log p(\mathbf{Z}|\mathbf{X}, \theta^{old}) p(\mathbf{X}, \mathbf{Z}|\theta) d\mathbf{Z}}_{Q(\theta, \theta^{old})} - \underbrace{\int_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{old}) \log p(\mathbf{Z}|\mathbf{X}, \theta^{old}) d\mathbf{Z}}_{-H_{q(\mathbf{Z})}} \tag{2.7}$$

where $Q(\theta, \theta^{old})$ is the expectation of the complete log likelihood under the posterior distribution $q(\mathbf{Z})$ evaluated for a general θ and $H_{q(\mathbf{Z})}$ is the entropy of the posterior distribution. Maximizing this lower bound gives a new parameter set θ^{new} and process continues by calculating a new lower bound using the updated parameters. In order to maximize the lower bound for θ, we can omit the entropy term since it's not a function of θ, but θ^{old}. We can write the E and M steps as

(i) E-step: Evaluate $p(\mathbf{Z}|\mathbf{X}, \theta^{old})$

(ii) M-Step: $\theta^{new} \leftarrow \text{argmax}_\theta \, Q(\theta, \theta^{old})$

In some cases the calculation of the posterior distribution $p(\mathbf{Z}|\mathbf{X}, \theta^{old})$ or its expected value may also be infeasible. This leads us to use a approximate distribution which results in approximate inference of the parameters. A common deterministic approximation scheme is *variational approximation*, where $\mathbf{Z}$ is partitioned into disjoint groups $\mathbf{Z}_i$ and the posterior distribution is factorized as

$$q(\mathbf{Z}) = \prod_k q_k(\mathbf{Z}_k) \tag{2.8}$$

In this setting we iteratively calculate $q_k(\mathbf{Z}_k)$ as

$$q_k(\mathbf{Z}_k) \propto \exp(\langle \log(p(\mathbf{X}, \mathbf{Z}|\theta) \rangle q_{-k}(\mathbf{Z}) \tag{2.9}$$

where $q_{-k}(\mathbf{Z}) = \prod_{i \neq k} q_i(\mathbf{Z}_i)$.

Table 2.1. BCPM variables and parameters.

Variable	Description
$s^{(1:T)}$	Reset switches
$\mathbf{h}^{(0:T)}$	Hidden state vectors
$\mathbf{v}^{(1:T)}$	Observation vectors
π	Reset probability
Θ	Prior distribution of hidden states
$\mathbf{w}$	Parameters of the Θ distribution
Ω	Observation model

2.2. Multiple Change Point Models

Modeling a time series data in order to detect anomalies and making exact inference to detect change point in time which starts those abnormal behaviors is an excellent example of the Bayesian approach.Multiple change point models are a special form of hierarchical Markov models [21], where the observations conditionally depend on latent states, and the states either follow the previous regime or jump to a new one, randomly. As far as network monitoring is concerned, these regime changes imply anomalous events, and which may be related to some security threats. The generative equations of the multiple change point model can be given as

$$\mathbf{h}^{(0)} \sim \Omega(\mathbf{h}^{(0)}; \mathbf{w}) \tag{2.10}$$

$$s^{(t)} \sim [s^{(t)} = 0]\pi + [s^{(t)} = 1](1 - \pi) \tag{2.11}$$

$$\mathbf{h}^{(t)}|s^{(t)}, \mathbf{h}^{(t-1)} \sim [s^{(t)} = 0]\delta(\mathbf{h}^{(t)} - \mathbf{h}^{(t-1)}) + [s^{(t)} = 1]\Omega(\mathbf{h}^{(t)}; \mathbf{w}) \tag{2.12}$$

$$\mathbf{v}^{(t)}|\mathbf{h}^{(t)} \sim \Theta(\mathbf{v}^{(t)}; \mathbf{h}^{(t)}) \tag{2.13}$$

where δ is Dirac delta function.

The observation $\mathbf{v}^{(t)}$ at time t, is assumed to be a random variable sampled from a $\Theta(\mathbf{v}; \mathbf{h})$. The model allows $\mathbf{h}^{(t)}$ to change as many times as required during the run of the algorithm. Initially, $\mathbf{h}^{(0)}$ is drawn from a $\Omega(\mathbf{h}^{(0)}; \mathbf{w})$ distribution. Afterwards, at

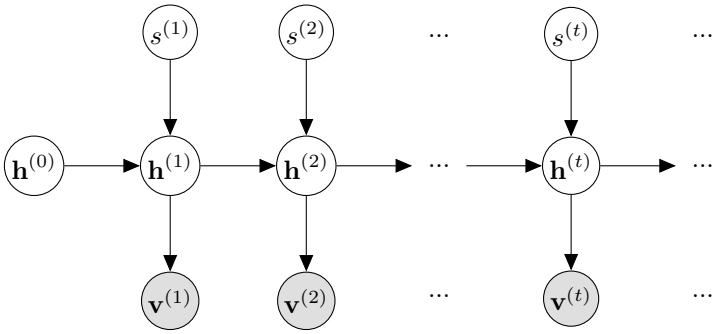


Figure 2.1. Graphical representation of the BCPM.

each time instance t, $\mathbf{h}^{(t)}$ is either re-drawn from the same initial distribution or set to the previous value $\mathbf{h}^{(t-1)}$. The decision for change is given by a Bernoulli random variable $s^{(t)}$. The graphical representation of the multiple change point model is given in Figure 2.1.

The observation model Θ distribution and its prior distribution Ω can be selected according the data where this model is fitted. The details on data features and the distributions used in the change point model for DDoS attack detection are given in Section 3.2.4. At this stage, it suffices to know that Ω distribution is the conjugate prior of the Θ distribution.

The prior probability of change, π, and the parameters w of the prior distribution $\Omega(\mathbf{h}^{(t)}; \mathbf{w})$ are the hyperparameters of our model. Provided that these hyperparameters are known, and the system is fully observable, meaning that the change points $s^{(1:T)}$, hidden states $\mathbf{h}^{(0:T)}$ and observations $\mathbf{v}^{(1:T)}$ are known, we can calculate the full joint likelihood as

$$p(s^{(1:T)}, \mathbf{h}^{(0:T)}, \mathbf{v}^{(1:T)}) = p(\mathbf{h}^{(0)}) \prod_{t=1}^{T} p(s^{(t)}) p(\mathbf{h}^{(t)} | \mathbf{h}^{(t-1)}, s^{(t)}) p(\mathbf{v}^{(t)} | \mathbf{h}^{(t)}) \tag{2.14}$$

In reality, the change switches $s^{(t)}$ and the hidden states $\mathbf{h}^{(t)}$ are not observed, and the problem of detecting a change point event at time t is formulated as calculating the posterior probability that $p(s^{(t)} = 1|\mathbf{v}^{(1:T)})$. From the Bayes rule, we can write

$$p(s^{(t)}|\mathbf{v}^{(1:T)}) = \frac{p(\mathbf{v}^{(1:T)}, s^{(t)})}{p(\mathbf{v}^{(1:T)})} \propto p(\mathbf{v}^{(1:T)}, s^{(t)}) \tag{2.15}$$

The probability of change at time t can be inferred online by calculating the filtering distribution $p(s^{(t)}|\mathbf{v}^{(1:t)})$, or in an offline manner by the smoothing distribution $p(s^{(t)}|\mathbf{v}^{(1:T)})$. The calculations can be done efficiently via the recursive forward-backward algorithm [22]. The filtering density is calculated by the forward recursion of the α messages:

$$\begin{aligned} \alpha(s^{(t)}, \mathbf{h}^{(t)}) &\equiv p(s^{(t)}, \mathbf{h}^{(t)}, \mathbf{v}^{(1:t)}) && (2.16) \\ &= \sum_{s^{(t-1)}} \int_{\mathbf{h}^{(t-1)}} p(\mathbf{h}^{(t)}|\mathbf{h}^{(t-1)}, s^{(t)}) \alpha(s^{(t-1)}, \mathbf{h}^{(t-1)}) d\mathbf{h}^{(t-1)} \\ &\quad \times p(\mathbf{v}^{(t)}|\mathbf{h}^{(t)}) \times p(s^{(t)}) && (2.17) \end{aligned}$$

Then, the change probability is calculated as

$$p(s^{(t)}|\mathbf{v}^{(1:t)}) \propto p(s^{(t)}, \mathbf{v}^{(1:t)}) = \int_{\mathbf{h}^{(t)}} \alpha(s^{(t)}, \mathbf{h}^{(t)}) d\mathbf{h}^{(t)} \tag{2.18}$$

In an offline setting, where we can calculate decisions using the full observations of the time series $\mathbf{v}^{(1:T)}$, we can smooth the filtering distribution with backward recursions to get a stronger estimate $p(s^{(t)}|\mathbf{v}^{(1:T)})$. The backward recursion can be written as

$$\begin{aligned} \beta(s^{(t)}, \mathbf{h}^{(t)}) &\equiv p(\mathbf{v}^{(t+1:T)}|s^{(t)}, \mathbf{h}^{(t)}) && (2.19) \\ &= \sum_{s^{(t+1)}} \int_{\mathbf{h}^{(t+1)}} p(\mathbf{h}^{(t+1)}|\mathbf{h}^{(t)}, s^{(t+1)}) \beta(s^{(t+1)}, \mathbf{h}^{(t+1)}) d\mathbf{h}^{(t+1)} \\ &\quad \times p(\mathbf{v}^{(t)}|\mathbf{h}^{(t)}) \times p(s^{(t)}) && (2.20) \end{aligned}$$

The smoothed density is calculated as

$$p(s^{(t)}|\mathbf{v}^{(1:T)}) \propto \int_{\mathbf{h}^{(t)}} p(s^{(t)}, \mathbf{h}^{(t)}, \mathbf{v}^{(1:t)}) p(\mathbf{v}^{(t+1:T)}|s^{(t)}, \mathbf{h}^{(t)}) \tag{2.21}$$

$$= \int_{\mathbf{h}^{(t)}} \alpha(s^{(t)}, \mathbf{h}^{(t)}) \beta(s^{(t)}, \mathbf{h}^{(t)}) d\mathbf{h}^{(t)} \tag{2.22}$$

Real-time anomaly detection tracks streaming data, so that the $v_{1:T}$ is not a practical expression, since T is not bounded. Furthermore, anomaly detection is a time-critical task, which implies that the change points must be recognized as soon as possible. Therefore, calculating a smoothing density is feasible only if the system is allowed to make deferred change point decisions for a fixed amount of time L, which is called the *lag*. In such a case, the process is called fixed-lag smoothing, where the change point inference for s_t is done at time $t + L$ by calculating the density $p(s^{(t)}|\mathbf{v}^{(1:t+L)})$. It is important to note that this process requires calculating a backward recursion for L steps starting at each time $t + L$, and this increases the processing complexity.

2.3. Nonnegative Tensor Factorization

Nonnegative tensor factorization (NTF) is the generalization of the 2-dimensional NMF model to multiple dimensions. In NTF, an N-dimensional tensor is approximated by the multiplication of lower dimensional factors. Unlike NMF, tensor factorization can be done in multiple ways. In this work, we are going to use the PARAFAC [23–25] factorization scheme.

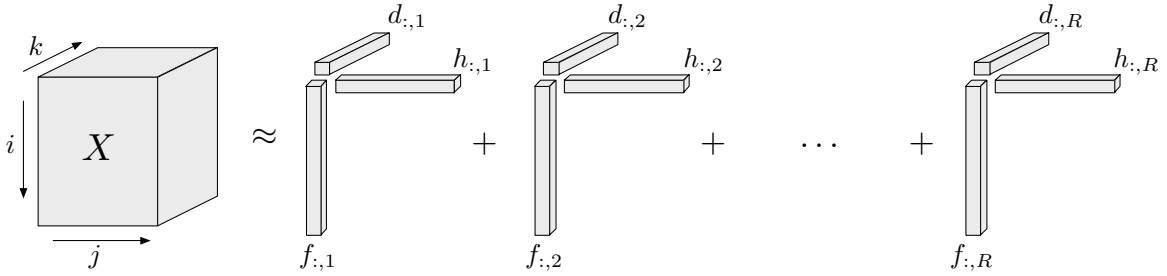
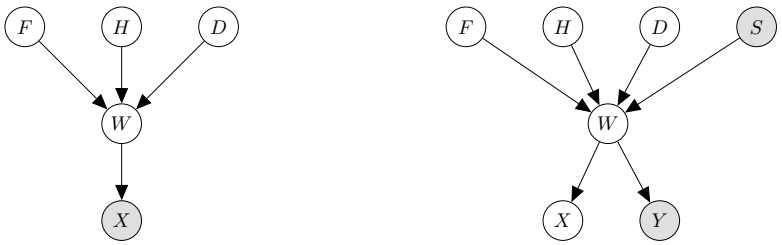


Figure 2.2. PARAFAC factorization.

In PARAFAC, an $I_1 \times I_2 \times \ldots \times I_N$ tensor is approximated by $I_n \times R$ matrices for $n \in [1, N]$. Here, R is the number of components, i.e. the number of clusters in the data. Figure 2.2 shows the PARAFAC factorization of our flow length distribution (FLD) tensor $\mathcal{X}$, into 3 factors: an $I \times R$ factor $\mathbf{F}$ for representing the flow length clusters, a $J \times R$ factor $\mathbf{H}$ for representing hourly behavior and a $K \times R$ factor $\mathbf{D}$ for representing the daily behavior of the data. Every single entry of the $\mathcal{X}$ tensor is approximated by

$$x_{i,j,k} \approx \hat{x}_{i,j,k} = \sum_r f_{i,r} h_{j,r} d_{k,r} \tag{2.23}$$

Bro [26] explains that the PARAFAC factorization is unique under certain circumstances, where uniqueness is defined as begin unable to rotate the factorization without loss of fit. NMF and NTF are statistical models that imposes nonnegativity constraint without uniqueness property. The uniqueness may be important if individual factors are of special interest. In our case, we are concerned with the estimation of the original data tensor $\mathcal{X}$ from sampled tensor $\mathcal{Y}$, but not the individual factors for



(a) NTF. (b) ThinNTF.

Figure 2.3. Graphical models for NTF and ThinNTF in PARAFAC scheme.

any interpretation. Our problem is more close to a missing value imputation problem, hence uniqueness is not a requirement.

2.4. Thin Nonnegative Tensor Factorization

Thin nonnegative tensor factorization (ThinNTF) is basically an NTF with an additional constant factor, which in our case is the sampling matrix S. Figure 2.3 shows the graphical models of the NTF and the ThinNTF models for factorizing original and sampled flow length observations. In the graphical models, the shaded nodes are the observed entities, and the unshaded ones are the latent entities.

In Section 4.4, we have described the sampling process as a matrix multiplication operation with a sampling matrix $\mathbf{S}$. In ThinNTF, this sampling matrix operates on the original tensor $\mathcal{X}$ and creates a thinned version of it, which we call $\mathcal{Y}$, by down-sampling its entries according to a sampling scheme, as shown in Figure 2.4. The entries of $\mathcal{Y}$ tensor $y_{\nu,j,k}$ presents the number of flows of sampled-length ν, at hour j at day k. The $\otimes_1$ operation denotes the 1-mode product of matrix $\mathbf{S}^T$ and tensor $\mathcal{X}$, which corresponds to the set of matrix multiplications $\mathcal{Y}_{:,:,k} = \mathbf{S}^T\mathcal{X}_{:,:,k}$ for $k \in [1, K]$.

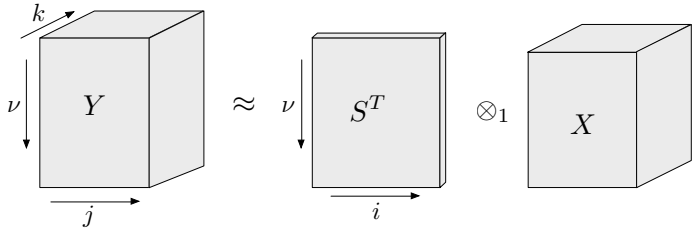


Figure 2.4. ThinNTF factorization.

In this scheme, one can immediately suspect that $\mathcal{X}$ can be estimated by $(\mathbf{S}^T)^{-1}\otimes_1 \mathcal{Y}$. However, this solution is not feasible for several reasons. First, the $\mathbf{S}$ matrix is not square, hence not invertible. Instead its pseudo-inverse can be calculated but this does not impose nonnegativity. Moreover, the top slice of the $\mathcal{Y}$ tensor, which stores the number of flows with zero-sampled size is never observed, hence must be estimated. Therefore we need a solid statistical model and an inference method to estimate $\mathcal{X}$ under this model.

In ThinNTF, we observe the $\mathcal{Y}$ tensor, but try to factorize the $\mathcal{X}$ tensor, which is latent (Figure 2.3(b)). In the end, the factors of $\mathcal{X}$ are going to provide us an approximation $\hat{\mathcal{X}}$ which solves the original flow length distribution reconstruction problem. We mathematically express this approximation as

$$y_{\nu,j,k} \approx \hat{y}_{\nu,j,k} = \sum_{i,r} s_{\nu,i} f_{i,r} h_{j,r} d_{k,r} \tag{2.24}$$

where $\mathbf{F}, \mathbf{H}$ and $\mathbf{D}$ are described in exactly the same way in the original NTF case. In subsections 4.4.1 and 4.4.2, we described two different $\mathbf{S}$ matrices for two different schemes. The ThinNTF model can be employed with any sampling method as long as it is described with a sampling matrix.

2.4.1. Generative Model

Taking Bayesian approach, we first provide a generative model for the ThinNTF, then describe how we can estimate the posterior probabilities of model parameters (in this case, the factor matrices) conditioned on the sampled flow length observations $\mathcal{Y}$ and the sampling matrix $\mathbf{S}$ using the well known Bayes rule. Table 2.2 contains all tensors and matrices used in the model together with their index sets.

Table 2.2. Tensors in the model and their corresponding index sets.

Tensor	**Index Set**	**Description**
$\mathcal{X}$	i, j, k	Original flow length tensor
$\mathcal{Y}$	ν, j, k	Sampled flow length tensor
$\mathcal{M}$	ν, j, k	Mask tensor
$\mathcal{W}$	ν, i, j, k, r	Latent variable tensor
$\mathbf{F}$	i, r	Flow length factor
$\mathbf{H}$	j, r	Hour of day factor
$\mathbf{D}$	k, r	Day of week factor
$\mathbf{S}$	i, ν	Sampling matrix
$\mathbf{A}^F, \mathbf{B}^F$	i, r	Gamma priors for $\mathbf{F}$
$\mathbf{A}^H, \mathbf{B}^H$	j, r	Gamma priors for $\mathbf{H}$
$\mathbf{A}^D, \mathbf{B}^D$	k, r	Gamma priors for $\mathbf{D}$

The original and latent data tensor $\mathcal{X}$, and the sampled and observed data tensor Y have nonnegative integer entries. The natural probability distribution for this type of count data is the Poisson distribution. We assume that each entry of a latent 5-dimensional tensor $\mathcal{W}$ is drawn from a Poisson distribution whose parameters are functions of sampling matrix $\mathbf{S}$ and factors $\mathbf{F}, \mathbf{H}$ and, $\mathbf{D}$, such as

$$w_{\nu,i,j,k,r} \sim \mathcal{PO}(w_{\nu,i,j,k,r}; s_{\nu,i} f_{i,r} h_{j,r} d_{k,r}) \tag{2.25}$$

We choose the prior distributions for the factor entries as the Gamma distribution since it is the conjugate prior of Poisson distribution [27]. For each entry of factor $\mathbf{F}$, we write

$$f_{i,r} \sim \mathcal{G}\left(f_{i,r}; a^f_{i,r}, \frac{b^f_{i,r}}{a^f_{i,r}}\right) \tag{2.26}$$

with shape parameter κ and scale parameter θ. In our generative model, the parameters for Gamma distributions are $\kappa = a^f_{i,r}$ and $\Theta = b^f_{i,r}/a^f_{i,r}$ respectively. This means that

the mean of $f_{i,r}$ is $\kappa\Theta = b^f_{i,r}$, which is independent of $a^f_{i,r}$. The variance of $f_{i,r}$ becomes $\kappa\Theta^2 = (b^f_{i,r})^2/a^f_{i,r}$, which means that as $a^f_{i,r}$ gets smaller, the factors gets sparser. In order to avoid repetition, we are going to omit the equations regarding the factors $\mathbf{H}$ and $\mathbf{D}$ throughout the paper. These factors behave exactly like factor $\mathbf{F}$ and it's easy to derive equations related to these factors once their corresponding equation for $\mathbf{F}$ is given.

Finally, we generate $\mathcal{X}$ and $\mathcal{Y}$ tensor from $\mathcal{W}$. Each entry $w_{\nu,i,j,k,r}$ of $\mathcal{W}$ can be interpreted as the number of original flows of length i, generated on hour j, day k, by cluster r and observed as length ν. By summing $\mathcal{W}$ over dimensions cluster (r) and original lengths (i), we get the sampled observations tensor $\mathcal{Y}$. Similarly, by summing $\mathcal{W}$ over dimensions cluster (r) and sampled lengths (ν), we get the original flow length tensor $\mathcal{X}$. The whole generative process is summarized in Figure 2.5. The set of all indexes and tensors in the model are summarized in Table 2.2.

2.4.2. Variational Bayes for ThinNTF

After defining the generative model, we can inter the factors $\mathbf{F}, \mathbf{H}$, and $\mathbf{D}$ of a sampled flow length observation tensor $\mathcal{Y}$. In the original NMF paper, Lee and Seung [28] provide fixed-point update equations for inferring the factors. Bro [26] gives similar fixed-point equations for updating the factors in PARAFAC factorization. Cemgil [29] shows that these updates correspond to the Kullback-Leibler minimization between the original matrix (or tensor $\mathcal{X}$) and the approximated one ($\hat{\mathcal{X}}$), and also provides a full Bayesian variational algorithm for the matrix factorization. Ermis et. al. [30] provide a similar variational algorithm for the Gamma-Poisson tensor factorization.

We start our Bayesian inference by calculating the posterior distributions over the factors $\mathbf{F}, \mathbf{H}$ and $\mathbf{D}$ conditioned on observed tensor $\mathcal{Y}$. For notational clarity, we introduce $\theta = (\mathbf{A}^F, \mathbf{B}^F, \mathbf{A}^H, \mathbf{B}^H, \mathbf{A}^D, \mathbf{B}^D)$ as the list of model hyper-parameters. The

function RandInit($\mathbf{S}, \mathbf{A}^F, \mathbf{B}^F, \mathbf{A}^H, \mathbf{B}^H, \mathbf{A}^D, \mathbf{B}^D$)

// Sample factor $\mathbf{F}$ from Gamma($\mathbf{A}^F, \mathbf{B}^F$)

for all $i \in [1, I],\ r \in [1, R]$ **do**

$f_{i,r} \sim \mathcal{G}\left(f_{i,r}; a^f_{i,r}, \frac{b^f_{i,r}}{a^f_{i,r}}\right)$

// Sample factor $\mathbf{H}$ from Gamma($\mathbf{A}^H, \mathbf{B}^H$)

for all $k \in [1, K],\ r \in [1, R]$ **do**

$h_{j,r} \sim \mathcal{G}\left(h_{j,r}; a^h_{j,r}, \frac{b^h_{j,r}}{a^h_{j,r}}\right)$

// Sample factor $\mathbf{D}$ from Gamma($\mathbf{A}^D, \mathbf{B}^D$)

for all $j \in [1, J],\ r \in [1, R]$ **do**

$d_{k,r} \sim \mathcal{G}\left(d_{k,r}; a^d_{k,r}, \frac{b^d_{k,r}}{a^d_{k,r}}\right)$

// Sample latent tensor $\mathcal{W}$ from Poisson distributions

for all $\nu \in [1, I+1],\ i \in [1, I],\ j \in [1, J],\ k \in [1, K],\ r \in [1, R]$ **do**

$w_{\nu,i,j,k,r} \sim \mathcal{PO}(w_{\nu,i,j,k,r}; s_{\nu,i} f_{i,r} h_{j,r} d_{k,r})$

return {$\mathbf{F}$, $\mathbf{H}$, $\mathbf{D}$, $\mathcal{W}$}

function GenerateData($\mathbf{S}, \mathbf{A}^F, \mathbf{B}^F, \mathbf{A}^H, \mathbf{B}^H, \mathbf{A}^D, \mathbf{B}^D$)

// Randomly initialize factors and latent tensor

$\{\mathbf{F}, \mathbf{H}, \mathbf{D}, \mathcal{W}\} \leftarrow$ RandInit($\mathbf{S}, \mathbf{A}^F, \mathbf{B}^F, \mathbf{A}^H, \mathbf{B}^H, \mathbf{A}^D, \mathbf{B}^D$)

// Generate original tensor $\mathcal{X}$

for all $i \in [1, I],\ j \in [1, J],\ k \in [1, K]$ **do**

$x_{i,j,k} = \sum_{\nu,r} w_{\nu,i,j,k,r}$

// Generate sampled tensor $\mathcal{Y}$

for all $\nu \in [1, I+1],\ j \in [1, J],\ k \in [1, K]$ **do**

$y_{\nu,j,k} = \sum_{i,r} w_{\nu,i,j,k,r}$

return {$\mathbf{F}$, $\mathbf{H}$, $\mathbf{D}$, $\mathcal{W}$, $\mathcal{X}$, $\mathcal{Y}$}

Figure 2.5. ThinNTF generative model.

log-likelihood observing $\mathcal{Y}$ under the model parameters θ is written as

$$\log p(\mathcal{Y}|\theta, \mathbf{S}) = \log \int_{\mathbf{F},\mathbf{H},\mathbf{D}} d\mathbf{F}\ d\mathbf{H}\ d\mathbf{D} \sum_{\mathcal{W}} p(\mathcal{Y}, \mathcal{W}, \mathbf{F}, \mathbf{H}, \mathbf{D}|\theta, \mathbf{S}) \tag{2.27}$$

This log-likelihood is intractable due to the integration over the latent factors, but it is lower bounded as

$$\log p(\mathcal{Y}|\theta, \mathbf{S}) \leq \mathcal{L}_\theta \tag{2.28}$$

$$= \langle \log p(\mathcal{Y}, \mathcal{W}, \mathbf{F}, \mathbf{H}, \mathbf{D}|\theta, \mathbf{S}) \rangle_{q(\mathcal{W},\mathbf{F},\mathbf{H},\mathbf{D})} + \mathcal{H}_{q(\mathcal{W},\mathbf{F},\mathbf{H},\mathbf{D})} \tag{2.29}$$

where q is an auxiliary joint distribution of latent factors. This bound is tight when $q(\mathcal{W}, \mathbf{F}, \mathbf{H}, \mathbf{D}) = p(\mathcal{W}, \mathbf{F}, \mathbf{H}, \mathbf{D}|\mathcal{Y}, \theta, \mathbf{S})$. However, this is also intractable to calculate. Instead, we use a variational approximation [31] for q such that

$$q(\mathcal{W}) \propto \exp\left(\langle \log p(\mathcal{Y}, \mathcal{W}, \mathbf{F}, \mathbf{H}, \mathbf{D}|\theta) \rangle_{q(\mathbf{F},\mathbf{H},\mathbf{D})} \right) \tag{2.30}$$

$$q(\mathbf{F}) \propto \exp\left(\langle \log p(\mathcal{Y}, \mathcal{W}, \mathbf{F}, \mathbf{H}, \mathbf{D}|\theta) \rangle_{q(\mathcal{W},\mathbf{H},\mathbf{D})} \right) \tag{2.31}$$

$$q(\mathbf{H}) \propto \exp\left(\langle \log p(\mathcal{Y}, \mathcal{W}, \mathbf{F}, \mathbf{H}, \mathbf{D}|\theta) \rangle q(\mathcal{W}, \mathbf{F}, \mathbf{D}) \right) \tag{2.32}$$

$$q(\mathbf{D}) \propto \exp\left(\langle \log p(\mathcal{Y}, \mathcal{W}, \mathbf{F}, \mathbf{H}, \mathbf{D}|\theta) \rangle_{q(\mathcal{W},\mathbf{F},\mathbf{H})} \right) \tag{2.33}$$

where we iteratively update the posterior distribution of each factor by calculating the expectation of the logarithm of the full joint likelihood $p(\mathcal{Y}, \mathcal{W}, \mathbf{F}, \mathbf{H}, \mathbf{D})$ under the posteriors of all other latent factors.

2.4.3. Update Equations

Here we provide the update equations for $q(\mathcal{W})$ and $q(\mathbf{F})$. The updates of $q(\mathbf{H})$ and $q(\mathbf{D})$ can be easily deduced from the update equations of $q(\mathbf{F})$. The full joint

likelihood whose expectation will be calculated at each step is

$$\mathcal{J}_\theta = \log p(\mathcal{Y}, \mathcal{W}, \mathbf{F}, \mathbf{H}, \mathbf{D}|\theta) \tag{2.34}$$

$$= \log p(\mathcal{Y}|\mathcal{W}) + \log p(\mathcal{W}|\mathbf{F}, \mathbf{H}, \mathbf{D}) + \log p(\mathbf{F}|\theta) + \log p(\mathbf{H}|\theta) + \log p(\mathbf{D}|\theta) \tag{2.35}$$

where $\mathcal{Y}|\mathcal{W}$ is a degenerate distribution ($\delta()$) to make sure the summation of $\sum_{i,r} \mathcal{W}$ equals $\mathcal{Y}$. By inserting the necessary Poisson and Gamma distributions given in the generative model into the Equation 2.35 we get the following expression

$$\begin{aligned}
\mathcal{J}_\theta = & \sum_{\nu,j,k} m_{\nu,j,k} \log \delta \left(y_{\nu,j,k} - \sum_{i,r} w_{\nu,i,j,k,r} \right) \\
& + \sum_{i,r} \sum_{\nu,j,k} m_{\nu,j,k} \Big(w_{\nu,i,j,k,r} \log s_{\nu,i} f_{i,r} h_{j,r} d_{k,r} \\
& \qquad - s_{\nu,i} f_{i,r} h_{j,r} d_{k,r} - \log \Gamma(w_{\nu,i,j,k,r} + 1) \Big) \\
& + \sum_{i,r} \left((a^f_{i,r} - 1) \log f_{i,r} - f_{i,r} \frac{a^f_{i,r}}{b^f_{i,r}} - a^f_{i,r} \log \frac{b^f_{i,r}}{a^f_{i,r}} - \log \Gamma(a^f_{i,r}) \right) \\
& + \sum_{j,r} \left((a^h_{j,r} - 1) \log h_{j,r} - h_{j,r} \frac{a^h_{j,r}}{b^h_{j,r}} - a^h_{j,r} \log \frac{b^h_{j,r}}{a^h_{j,r}} - \log \Gamma(a^h_{j,r}) \right) \\
& + \sum_{k,r} \left((a^d_{k,r} - 1) \log d_{k,r} - d_{k,r} \frac{a^d_{k,r}}{b^d_{k,r}} - a^d_{k,r} \log \frac{b^d_{k,r}}{a^d_{k,r}} - \log \Gamma(a^d_{k,r}) \right)
\end{aligned} \tag{2.36}$$

Considering the terms in the log-likelihood expression in Equation 2.36, that only includes $w_{\nu,i,j,k,r}$, we find that

$$\begin{aligned}
q(w_{v,:,j,k,:}) \propto & \exp \left(m_{\nu,j,k} \log \delta \left(y_{\nu,j,k} - \sum_{i,r} w_{\nu,i,j,k,r} \right) \right. \\
& \left. + \sum_{i,r} m_{\nu,j,k} \Big(w_{\nu,i,j,k,r} \log s_{\nu,i} f_{i,r} h_{j,r} d_{k,r} - \log \Gamma(w_{\nu,i,j,k,r} + 1) \Big) \right) \\
\propto & \text{ Multinomial}(w_{\nu,j,k,:}, x_{i,j,k}, p_{\nu,i,j,k,r})^{m_{\nu,j,k}}
\end{aligned} \tag{2.37}$$

where, $w_{\nu,j,k,:,:}$ becomes multinomial distributed. The expectation of $\mathcal{W}$ is calculated as

$$p_{\nu,i,j,k,r} = \frac{\exp\left(s_{\nu,i} + \langle \log f_{i,r} \rangle + \langle \log h_{j,r} \rangle + \langle \log d_{k,r} \rangle\right)}{\sum_{i,r} \exp\left(s_{\nu,i} + \langle \log f_{i,r} \rangle + \langle \log h_{j,r} \rangle + \langle \log d_{k,r} \rangle\right)} \tag{2.38}$$

$$\langle w_{\nu,i,j,k,r} \rangle = y_{\nu,j,k} p_{\nu,i,j,k,r} \tag{2.39}$$

Similarly, considering the terms in log-likelihood Equation 2.36 that only includes $f_{i,r}$, we find that

$$\begin{aligned} q(f_{i,r}) &\propto \left(\sum_{\nu,j,k} m_{\nu,j,k} \langle w_{\nu,i,j,k,r} \rangle + a^f_{i,r} - 1 \right) \log f_{i,r} \\ &\quad - \left(\sum_{\nu,j,k} m_{\nu,j,k} s_{\nu,i} \langle h_{j,r} \rangle \langle d_{k,r} \rangle + \frac{a^f_{i,r}}{b^f_{i,r}} \right) f_{i,r} \qquad (2.40) \\ &\propto \text{Gamma}(f_{i,r}; \alpha^f_{i,r}, \beta^f_{i,r}) \qquad (2.41) \end{aligned}$$

where $f_{i,r}$ becomes Gamma distributed with shape and scale parameters

$$\alpha^f_{i,r} = a^f_{i,r} + \sum_{\nu,j,k} m_{\nu,j,k} \langle w_{\nu,i,j,k,r} \rangle \tag{2.42}$$

$$\beta^f_{i,r} = \left(\frac{a^f_{i,r}}{b^f_{i,r}} + \sum_{\nu,j,k} m_{\nu,j,k} s_{\nu,i} \langle h_{j,r} \rangle \langle d_{k,r} \rangle \right)^{-1} \tag{2.43}$$

We calculate the expectation of $f_{i,r}$ and the logarithm of $f_{i,r}$ as

$$\langle f_{i,r} \rangle = \alpha^f_{i,r} \beta^f_{i,r} \tag{2.44}$$

$$\langle \log f_{i,r} \rangle = \Psi(\alpha^f_{i,r}) + \log \beta^f_{i,r} \tag{2.45}$$

The variational Bayes algorithm that uses the above equations is given in Figure 2.6. The calculation of the lower bound is given in Appendix 1. The exact derivations of all equations can be found in [15].

1: **function** ThinNTF_VB($\mathcal{Y}$, $\mathbf{S}$, $\mathbf{A}^F, \mathbf{B}^F, \mathbf{A}^H, \mathbf{B}^H, \mathbf{A}^D, \mathbf{B}^D$)
// Randomly initialize factors and latent tensor
2: $\{\mathbf{F}, \mathbf{H}, \mathbf{D}, \mathcal{W}\} \leftarrow \text{RandInit}(\mathbf{S}, \mathbf{A}^F, \mathbf{B}^F, \mathbf{A}^H, \mathbf{B}^H, \mathbf{A}^D, \mathbf{B}^D)$
3: **repeat**
4: Calculate $\alpha^f_{i,r}, \beta^f_{i,r}$ and $\langle f_{i,r} \rangle$ as in Equations 2.42, 2.43 and 2.44.
5: Calculate $\alpha^h_{j,r}, \beta^h_{k,r}$ and $\langle h_{j,r} \rangle$ similarly.
6: Calculate $\alpha^d_{k,r}, \beta^d_{k,r}$ and $\langle d_{k,r} \rangle$ similarly.
7: Calculate $\langle \log f_{i,r} \rangle$ as in Equation 2.45.
8: Calculate $\langle \log f_{i,r} \rangle$ similarly.
9: Calculate $\langle \log f_{i,r} \rangle$ similarly.
10: Calculate $\langle w_{\nu,i,j,k,r} \rangle$ as in Equation 2.39.
11: Calculate lower bound
12: **until** Max iterations are reached or lower bound converged
13: **return** $\mathbf{F}$, $\mathbf{H}$, $\mathbf{D}$, $\mathcal{X}$

Figure 2.6. Variational Bayes algorithm.

2.4.4. Computational Complexity

The nonnegative tensor factorization is an NP-hard problem [32]. The variational Bayes algorithm we introduced in Figure 2.6 is an iterative solution that converges to a local maximum solution. The complexity of each iteration is determined by the leading term, which is the Equation 2.39. In general, calculating a ThinNTF model with R components for a κ dimensional tensor with all dimensions of length N has $O(\kappa N^{(\kappa+1)} R)$ complexity for a single iteration.

3. NETWORK SECURITY: DDOS ATTACK DETECTION

VoIP is the technology of carrying voice and multimedia communications through the IP networks. Due to its multimedia support and low infrastructure cost VoIP systems are worldwide taking over circuit-switched telephone networks. With the introduction of 5G, the VoIP is predicted to become the dominant methodology for voice and multimedia communications.

The VoIP systems transfer voice and multimedia data between communicating parties through the packet-switched IP networks based on data transfer protocols, such as the RTP. In addition, they require session-level signaling protocols for managing their communication sessions. Considering its lightweight nature, simplicity and ease of implementation, the SIP [9] is one of the most popular open standard signaling protocols designed for VoIP. SIP provides signaling functions necessary to register clients, check their locations and availability, exchange information on their data transmission capabilities, and provide handshakes necessary for connection setups.

Despite all their attractive features, the downside is that VoIP systems are more vulnerable to security threats compared to their circuit switched predecessors. There are two basic sources of security threats for VoIP systems. Firstly, VoIP systems are affected by all the lower protocol layer threats, e.g., the host of IP layer threats. Secondly, being an open standards protocol, suffers from many protocols-specific vulnerabilities, in other words, they are prone to security threats specifically designed to exploit the vulnerabilities of the underlying signaling protocols [33], [10]. These protocol-specific attacks are usually not classified as network attacks by the conventional network-level security systems. Therefore, VoIP systems need extra security mechanisms for detecting and preventing VoIP specific attacks.

One of the most frequently observed type of cyber-attack is the DDoS flooding attack [11], which is typically realized by sending a vast amount of network protocol messages to a victim. These types of attacks aim to exploit the weaknesses in the SIP

protocol or faults due to some poor implementation. An example of such DDoS flooding attack is the *INVITE* attack. In this case, the attacker tries to set up communication with many SIP users by sending INVITE requests to the SIP proxy server. The server, which maintains a table for each SIP session, holds an entry for each INVITE request and awaits response from the call receiver for a fixed amount of time. Eventually, the server reaches its memory capacity while trying to keep track of an excessive amount of connections. Another typical DDoS attack is the SYN-flooding [34], where a target network proxy is forced to maintain a barrage of TCP sessions, and eventually becomes unresponsive due to over-utilization of its resources. Thus DDoS flooding attacks aim to cripple a target system by overusing and eventually depleting its resources, such as bandwidth, CPU or memory, and making it unable to respond to the requests of its legitimate subscribers.

DDoS attacks can have negative impact on business since a target system cannot provide services to its customers during attacks. The downtime of servers creates revenue loss and reputation damage, which in turn leads to loss of revenue as well, for service providers. Furthermore, the productivity of workforce is reduced as employees cannot use affected systems for operations. Among victims of DDoS attacks, well-known companies can be found. For instance, GitHub was under attack for six days [35]. Another victim of such attacks was BBC where an online DDoS tool named BangStresser, which delivers attacks as a service, might be used [36].

A recent survey reports an increase in DDoS attacks, arguing that it might be a possible result of the proliferation of cheap and easy-to-launch attack tools [37]. According to another report [38], the number of attacks decreases whist the average peak attack size increases. For attacks targeting SIP based VoIP systems, there has been an upward trend as well [39]. Defense strategies for these common DDoS attacks have been studied extensively [40, 41].

Many network security systems have been developed for the detection of SIP-based DDoS attacks [42]. The majority of these systems uses supervised methods, such as thresholding [43] and rule-based pattern matching, as in [44, 45]. The supervised

methods require a training phase for learning patterns for each type of attack and for building a dictionary of known attacks. Attack detection is based on finding matching patterns between the current network state with one of the known attack patterns in the training set. However, when an unprecedented attack occurs, a supervised system can easily fail to detect it, since the pattern of the new attack will possibly be different than all the learned attack patterns. It becomes imperative then to re-train the system by an extended training data set which includes samples from the new attack traffic.

In this paper, we focus on the detection of SIP-specific DDoS flooding attacks [10, 12]. We aim therefore to develop a more robust and generalizable DDoS monitor based on anomaly detection principles. Anomaly detection [46] is an unsupervised methodology where the system is programmed to recognize significant deviations from its learned data patterns, and mark them as anomalous events. In our case, anomalous events are interpreted and marked, subject to further analysis, as security threats. We assume that the SIP server state has a stationary behavior under the so-called normal, "non-attack" SIP traffic, but that these statistics will change noticeably under a DDoS flooding attack. To sense these attacks, we have designed our feature vectors as consisting of a combination of incoming and outgoing SIP message counts plus the vector of resource usage measurements of the SIP proxy software. Our DDoS monitor is based on the Bayesian change point model [21] which models the normal SIP server behavior and infers changes that are possibly due to the DDoS attacks.

Collecting real-world VoIP network traces and annotating them without violating the privacy of the users is a tedious task. Therefore, for the proof of our concept, we conducted our experiments in a simulated environment. We developed a real-time SIP network simulator system, which models a social network for a group of users. The simulator generates actual voice conversation calls by setting up SIP sessions between users through a SIP proxy server. Our DDoS detection mechanism is deployed next to the SIP proxy server, so that it does not track RTP traffic between users. Therefore, the simulated SIP sessions are silent communications, i.e., actual data transfer via RTP is not generated. We generate DDoS attacks with the help of a commercial network vulnerability scanning tool Nova-VSpy [47], simultaneously with the VoIP simulation.

3.1. Related Work

There are comprehensive literature surveys on vulnerabilities of the SIP protocol [33], VoIP security research [10], DoS attacks targeting SIP networks [12] and security systems to counter SIP-based DoS attacks [42]. One of the earliest and simplest attempts to prevent single-source DoS flooding attacks in SIP systems was proposed by [43] where a rate limiter is deployed at the server to limit per-host SIP traffic. More elaborate methods were proposed to detect both single and distributed DoS attacks employing rule-based schemes, statistical methods, anomaly detection approaches, and machine learning tools.

Rule-based methods maintain a list of rules, or protocol finite state machines, and check the current server state against consistent patterns described in the rule set [44, 48–50]. [51] propose a large scale SIP firewall solution by combining several rule-based filters and attack mitigation mechanisms. While such rule-based systems are useful in detecting DoS attacks, they require carefully designed and perpetually updated rule books, and fine-tuned thresholds. Since these systems can easily miss a novel attack whose descriptive rule has not yet been learned, they need to be reinforced with additional tools based on statistical approaches.

Machine learning methods were proposed as an alternative to rule-based and statistical methods for DDoS flooding detection, including support vector machines [52], evolutionary algorithms [53], naive Bayes and decision trees [54]. [55], [56] give a comparison of 5 supervised classifiers and conclude that these methods provide good results on low-rate DoS attacks with little classification time overhead. Inherently, the success of supervised algorithms depends on the quality of the data set used during their training. For example in [56], authors employ different training sets for different basic scenarios. Obtaining such high quality training data can be difficult in a real world implementation of a supervised system. In contrast, we propose an unsupervised system, with an optional training phase to optimize its parameters. We show that setting those parameters empirically with the help of domain expertise is sufficient.

[57] were the first to propose applying change point detection in SIP networks. They present a cumulative sum (CUMSUM) algorithm in order to detect INVITE flooding. Later, [58] have developed a parametric version of this algorithm. [59] proposed to use additional features to enhance the accuracy of CUMSUM. [60] propose bloom filters to efficiently track incomplete SIP sessions and to raise an alarm if these exceed a certain threshold. The major disadvantage of these algorithms is that one needs to engineer different sets of features in order to detect different types of flooding attacks.

The works closest to our approach are the distance-based anomaly detection methods [61, 62], where a distance metric is used to measure the dissimilarity between the distributions of normal and observed traffic features. If the distance between the normal and observed distributions is above a threshold, an alarm is generated. Similar to our approach, these methods can be used to detect any type of network attack provided that full SIP message histogram is included in the feature set. Our method extends and generalizes these anomaly detection methods by introducing Bayesian framework, which models the SIP server state with a set of features that incorporates both network traffic and SIP server resource usage data. In our method, the attack decision relies on a robust posterior probability calculation rather than simple thresholding. To our best knowledge, this work presents the first Bayesian framework tailored specifically to model a SIP server in order to detect SIP anomalies, hence fills an important gap in the literature.

3.2. SIP Network Traffic

3.2.1. SIP Terminology

SIP is designed to initiate, modify and terminate communication sessions among agents. Four general types of SIP entities are defined in RFC 3261 [9] : user agents, proxy servers, redirect servers and registrars. A user agent (UA) is the endpoint entity that generate and receive SIP messages. In a typical SIP session, UA's communicate with by sending request and response messages to each other. The registrar is respon-

sible for registering the UA's, and storing their location information. The registered UA's communicate with each other through the proxy servers. The proxy servers delivers the request and response messages between UA's. Finally, the redirect servers allow proxy servers to communicate with other servers from external domains.

The SIP messages are basically divided into two categories: SIP requests and SIP responses. Each SIP request sent by a UA is answered by a corresponding SIP response. For examples, a UA can make a REGISTER request to the registrar in order to get online, make an INVITE request to another UA to start a call, or make a BYE request to terminate an ongoing conversation. A SIP response message generated for a request can be from one of the 6 SIP response categories: 1xx-provisional, 2xx-success, 3xx-redirection, 4xx-client, 5xx-server error or 6xx-global failure. For example, a UA may response with a 200-OK message for accepting an incoming request.

3.2.2. An Example Flow

An illustrative example of SIP message communication is given in Figure 3.1. It shows the flow of exchanged messages between a server and two users during a normal call. In this scenario, Alice initiates a call to Bob by sending an *INVITE* packet to the SIP Server. After authentication, the SIP server forwards this request to Bob. Similarly, the response of Bob, in this case *ACK* packet showing that the call is accepted, is transmitted to Alice through the SIP server. At the end, *BYE* messages terminate the conversation between Alice and Bob.

Once a SIP session is established, two endpoints start exchanging multimedia data such as audio conversations, video streams, etc. Recall that SIP, being a signaling protocol, is not involved in the multimedia data exchange between agents. Handshake on the kind, encoding, address and ports to be used for transfer and other details regarding the data exchange is usually achieved using Session Description Protocol (SDP)[63]. Additionally, real-time media delivery relies on RTP[64].

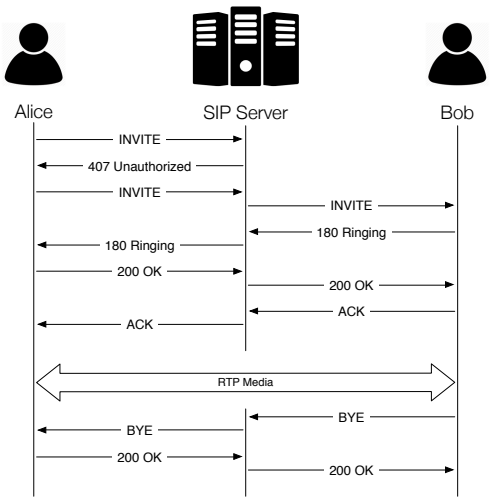


Figure 3.1. A call scenario in SIP network.

Real-world SIP packet exchange scenarios usually involve more than two servers and two agents. The above call setup case is illustrative but simplistic. For example, it does not specify how the server reaches out the caller. A setup in which Alice and Bob are not registered to the same server would require a location server and the retransmission of the INVITE message. Similarly, other features supported by SIP - such as call transfer, call park, conference - lead to distinct call flows. In summary, SIP message traffic data can be quite complex.

3.2.3. DDoS Attacks in SIP Networks

DDoS flooding attacks could rapidly affect network traffic characteristics and cause service degradation. Their impact is contingent on the attack parameters and differs substantially from one attack to another. A DDoS detection method is expected to signal an attack practically independent of its configuration. Therefore, DDoS detector must be robust, with high detection rate and low probability of false alarm under a wide range of realistic network conditions.

Mirkovic *et al.* [11] classified DDoS attack mechanisms on the basis of its impact on the victim. First, one would expect an increase in the incoming network traffic - even beyond the server's bandwidth - as a result of a flooding attack. The severity of this increase, however, is directly related to the resources the attacker possesses and

cannot be forecast beforehand. Second, the flooding rate does not necessarily stay the same throughout the attack. The authors also note that slow rate boosts, i.e. creeping attacks typically results in detection latency.

Another significant parameter is the SIP packet type used in the flooding. Typical DDoS scenarios consider a SIP server being flooded by one type of packet such as INVITE, SUBSCRIBE, or BYE. Nevertheless, DDoS attacks can also be performed using a judiciously selected mixture of SIP requests. Thus, intelligently mounted schemes such as slow boost attacks, multi-SIP packet attacks, multi-agent attacks that try to obfuscate their synchronism by time jittering require more advanced defense mechanisms and concomitantly more computation time and power. On the other hand, the DDoS shield must have low latency in order to timely initiate attack prevention.

3.2.4. DDoS detection via Multiple Change Point Model

The first step in Bayesian approach is to provide a probabilistic generative model for the observations collected from the system. However, before going into the mathematical details of our generative model, it's important to provide a clear definition of the observations. As we continuously monitor our SIP server, we collect real time statistics for a period of Δt and compile an observation vector v_t as a summary of statistics collected during that period. The v_t is an N dimensional vector composed of number of SIP request and response messages, number of server log messages, server statistics such as number of TCP connections observed during the period, together with the CPU and memory usage measured at the end. The complete list of the features collected from the system is given in Table 3.2 and explained in further detail in Section 3.4.

Our DDoS detection system includes a monitoring unit for observing and collecting network traffic data as well as SIP server activities. The monitoring unit collects and compiles network and server statistics into an observation vector, i.e., a feature vector, v_t at each $\sim \Delta t$ (1 second) time interval, as the resume of events that have occurred in the SIP server during that last observation interval. For each such feature

vector, the model infers whether the observation vector is generated by the previous regime, that is $s_t = 0$ and $h_t = h_{t-1}$, or whether the server state has jumped to a new regime, which means $s_t = 1$ and $h_t \sim \Omega(h_t; w)$.

We described a multiple change point model with arbitrary hidden state distribution Θ and observation model Ω, with the assumption that Θ is the conjugate prior of Ω for computational simplicity. Here we assign actual probability distributions for the hidden state and observation models.

We let the observation model Θbe a coupled distribution of multinomial and Poisson distributions, for modeling both the ratios and magnitudes of selected features from the observations. Without loss of generality, we assume that the features whose ratios will be modeled is stored in the first M positions of the observation vector v, denoted as $v_{1:M}$ and the remaining $N - M$ positions are filled with features whose magnitudes are modeled. Then, we can write the observation model as

$$\Theta(v) = \mathcal{M}(v_{1:M}; \pi) \times \prod_{i=M+1}^{N} \mathcal{P}(v_i; \lambda_i) \tag{3.1}$$

where Multinomial and Poisson distributions are defined as

$$\mathcal{M}(x; \pi) = \frac{\Gamma(\sum_i x_i + 1)}{\prod_i \Gamma(x_i + 1)} \prod_i \pi_i^{x_i} \tag{3.2}$$

$$\mathcal{P}(x; \lambda) = \frac{\lambda^x e^{-\lambda}}{\Gamma(x + 1)} \tag{3.3}$$

In this setup, the hidden states $h = (\pi; \lambda)$ are the respective parameters for the Multinomial and Poisson distributions. The prior distribution for our state vector h is the product of conjugate priors of these distributions, namely Dirichlet and Gamma.

$$\Omega(\pi, \lambda) = \mathcal{D}ir(\pi; \alpha) \times \prod_{i=M+1}^{N} \mathcal{G}(\lambda_i; a_i, b_i) \tag{3.4}$$

The Dirichlet and Gamma distributions are given as

$$\mathcal{D}ir(\pi;\alpha) = \frac{\Gamma\left(\sum_{i=1}^{M}\alpha_k\right)}{\sum_{i=1}^{M}\Gamma(\alpha_i)}\prod_{i=1}^{M}\pi_i^{\alpha_i-1} \tag{3.5}$$

$$\mathcal{G}(\lambda_i;a_i,b_i) = \frac{{b_i}^{a_i}}{\Gamma(a_i)}\lambda^{a_i-1}e^{-b_i\lambda} \tag{3.6}$$

where α is an M dimensional vector and a and b are N dimensional vectors such that each $\{a_i, b_i\}$ is a hyper-parameter for a Gamma distribution. Hence, the hyper-parameters w of the model is the set $w = (\alpha, a, b)$. The complete set of model variables and parameters are given in Table 2.1.

We conducted experiments to find the best observation model by enumerating all possible input feature combinations to the coupled observation model and comparing their inference results. The details of this experiment is given in Section 3.4.

3.2.5. Implementation Details and Complexity Analysis

The inference for the change point model requires calculating the $\alpha(s_t, h_t)$ and $\beta(s_t, h_t)$ message at the end of each time period. For discrete distributions, the memory required for storing α messages is twice the size of the discrete domain. We simply need to store a table of probabilities for each $\alpha(s_t = i, h_t = j)$, such that $i \in \{0, 1\}$ and $j \in Dom(\Omega)$ and update this table according to the Equation 2.17. The same is true for the β messages.

When the hidden state distribution has continuous domain, we can no longer have to express the α and β messages as mixtures Ω potentials. An Ω potential is described as

$$\phi(\pi, \lambda) = \exp(l)\Omega(\pi, \lambda; z, a, b) \tag{3.7}$$

We use the quadruple (z, a, b, l) to denote an Ω potential, where l is the logarithm of the normalizing constant.

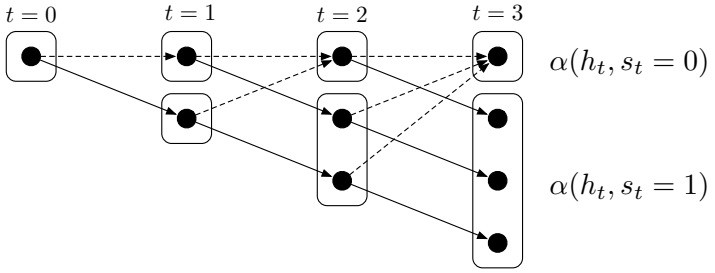


Figure 3.2. Expansion in the forward variable messages.

At each time step, the switching variable attains one of the two values, therefore, an additional Ω potential is added to the α and β messages to indicate the change potential. At each time t, the $\alpha(h_t, s_t = 1)$ is a single potential and $\alpha(h_t, s_t = 0)$ is a mixture of t potentials transferred from the previous state. This linear growth of the α messages for the forward recursion is illustrated in Figure 3.2.

This linear growth is not sustainable for online continuous tracking of the server state. One has to limit, then, the number of mixture potentials in the forward message by K, indicating the maximum number of components. Once a message reaches the maximum number of allowed components, at each subsequent step, the component with the minimum normalizing constant is pruned. Therefore, in the worst case, an α message will have K components and a β message L components, since we had decided to run the backward-recursion for only L steps. It follows then that, during filtering, at most K observation updates are required and during the smoothing operations, where we multiply an α message with a β message, $K \times L$ multiplications are performed. Therefore, the number of operations at any time instance is $O(KL)$. We empirically set $K = 100$, and the lag parameter $L = 5$. In our experiment setup, using more than 100 potentials had no significant contribution, and using a lag value of 5 significantly increased the accuracy of the system and gave as good performance as using longer lag values.

3.3. Real Time Analysis

Figure 3.3 presents the offline version of the main detection loop of our algorithm. Her offline is in the sense that the whole data set is available at the beginning of the

Table 3.1. Average run time of the BCPM algorithm.

Routine	Time (μs)
PREDICT	35
UPDATE	648
BACKWARD_FILTER	17
SMOOTH	1400
COMPUTE_CPP	7
Total	2107

algorithm. In the online version, the single loop in the algorithm will be run exactly once after each observation period. We time the individual functions of the algorithm whose descriptions are also given in Appendix C. The actual run time of the algorithm depends on the number of features used, the value of the lag L, and maximum number of components K. In this measurement we set $L = 5$, $K = 100$ and used all available features. The experiment is run offline, on an INTEL i7 CPU @2.7 GHz on a data sequence of 2000 observations. We can see from Table 3.1 that one iteration of the main loop executes in 2 ms (2107 μs) on the average, which allows online deployment of our algorithm.

3.3.1. Parameter Learning

During the inference stage, we had assumed that the hyper-parameters of our multiple change point model, namely the reset probability π and the latent state prior parameters w were given. In practice, these parameters must be set to appropriate values for accurate change point estimation. For a small number of parameters, a grid search method can give good parameter estimates; however for large models, i.e., for large dimensional w, the search method is not applicable. We have used a maximum likelihood approach to find the best hyper-parameters as a function of observations.

```
1: function BCPM(π, w, v_{1:T}, LAG, THRESHOLD)
2:     alpha ← [ ]
3:     for t = 1...T do
4:         alpha_p = Predict(alpha, π, w)
5:         alpha = Update(alpha_p, v_t)
6:         if LAG > 0 then
7:             beta = BackwardFilter(π, w, v_{t+1−LAG:t})
8:             gamma = Smooth(alpha, beta)
9:             cpp = ComputeCPP(gamma, len(beta))
10:        else
11:            cpp = ComputeCPP(alpha, 1)
12:        if cpp > THRESHOLD then
13:            Alarm()
```

Figure 3.3. Bayesian change point detection algorithm.

Given observations $v_{1:T}$, we maximize the log likelihood

$$\mathcal{L}_{\pi,w}(v_{1:T}) \equiv \log p(v_{1:T}|\pi, w) \tag{3.8}$$

$$= \log \sum_{s_{1:T}} \int_{h_{0:T}} p(s_{1:T}, h_{0:T}, v_{1:T}, |\pi, w) dh_{0:T} \tag{3.9}$$

Since this log likelihood expression is intractable due to the summation over latent parameters, we employ an iterative EM scheme to find the $\{\pi, w\}$ estimates. By Jensen's inequality, the log likelihood is lower bounded as

$$\mathcal{L}_{\pi,w}(v_{1:T}) \geq \langle \log p(s_{1:T}, h_{0:T}, v_{1:T}, |\pi, w) \rangle q(z) - \langle \log q(x) \rangle q(z) \tag{3.10}$$

This bound is tight for $q(z) = p(s_{1:T}, h_{0:T}|v_{1:T}, \pi, w)$. The log-likelihood can be maximized iteratively calculating the following equations:

$$q(z)^{new} = p(s_{1:T}, h_{0:T}|v_{1:T}, \pi^{old}, w^{old}) \tag{3.11}$$

$$(\pi^{new}, w^{new}) = \arg\max_{\pi,w} \big\langle p(s_{1:T}, h_{0:T}|v_{1:T}, \pi^{old}, w^{old}) \big\rangle q(z)^{new} \tag{3.12}$$

The detailed derivations of the EM algorithm for Dirichlet-Multinomial and Gamma-Poisson change point potentials are given in Equation B.

3.4. Experimental Setup

3.4.1. Data Generation

Our data generator, detailed in [19], is made up of four distinct modules: (1) a SIP server, (2) a traffic simulator, (3) a DDoS attack generator and (4) a network traffic monitor. As a registrar and SIP proxy server, we have used an Asterisk-based PBX software named *Trixbox* [65]. To mimic the normal message traffic on a SIP server, we have built *Boun-Sim* [19], a probabilistic SIP network simulation tool that initiates calls between a number of SIP endpoint entities in real-time. Concurrently with normal traffic simulation, a rich variety of DDoS attacks were generated by a commercial vulnerability scanning tool, called *NOVA V-Spy* [47]. The fourth and final component in the setup is the network monitor, a module that tracks the server, extracts and delivers features to the change point monitor.

Our simulation tool is driven by a probabilistic generative model to recreate typical user behaviors, such as making calls, answering, rejecting or ignoring an incoming call, and holding on an ongoing call. User actions generated by the Simulator are realized as actual SIP communications, where SIP messages are exchanged between UA's and Trixbox. Simulator omits the *RTP* messages carrying the actual conversational data between users, since these do not pass through the SIP server, and are not relevant to the outcome of the simulation. Details of the simulator is presented in Appendix D.

3.4.2. Data Traces

We have conducted our experiments on simulated data sets adjusted to four different network traffic and attack intensity scenarios. The generated network and attack message streams are controlled by two bi-level variables, one for network, the other for attacks, and each can be either *low* or *high*. To set the network traffic intensity, we tune the call rate parameters of the users since phone calls constitute the main source of the network traffic. To set the flood rate, we change the number of network packets delivered from V-Spy to the server per second. In the sequel, we refer to these data sets as *LOW-LOW*, *LOW-HIGH*, *HIGH-LOW* and *HIGH-HIGH*, where the two adjectives qualify, in order, the network traffic intensity and the flood rate.

All simulations are realized by assuming 500 active users registered to the server. Each data set contains 40 random DDoS attacks. The attacks last for about 20 seconds. and there is an interval of at least 25 seconds between two consecutive attacks; this results in a simulation sequence of around half an hour duration. IP addresses and the user id's of attackers were randomly chosen. The four parameters that modulate the attacks are listed below:

- *Attack Type:* We flood the server with five different SIP request packets, namely *REGISTER*, *INVITE*, *OPTIONS*, *CANCEL* and *BYE* requests. Each type of attack generates different types of changes in SIP server state.
- *Transport Protocol:* Since SIP operates independent of the transport protocol, we generated attacks over both TCP and UDP.
- *Fluctuation:* Nova V-Spy can generate floods with both constant and fluctuating rates. In half of our experiments, we generated floods with fluctuating rates.
- *Content Size:* Nova V-Spy can optionally insert dummy strings to the end of SIP messages, which is relevant to attack detection task since it affects the bandwidth consumption.

3.4.3. Data Features

Table 3.2 shows the features monitored for DDoS attack detection. We can divide the feature space roughly into five categories, the first two categories collected from server's network connection side, and the last three categories collected from server's resource management side. The first two categories consist of a variety the packet types in a SIP network: *SIP Requests* and *SIP Responses*; these packet types have different well-defined roles [9]. Notice that the actual features used in the detection model are the count statistics or histogram of these message type occurrences within an observation interval (e.g., 1 sec). The underlying assumption here is that a significant change in the pattern of SIP message histograms is a direct reflection of messaging traffic behavior, indicating possibly an anomaly i.e., an attack.

The other three feature categories are entitled as *Resource Usage*, *Asterisk Stats* and *Asterisk Logs*. The first of these consists of the pair of CPU usage and memory usage of the virtual machine in which Trixbox is installed. The second one is made up of features that reflect the load created by Asterisk. The last category counts the keywords in the log files generated by Asterisk. We conjecture that all these features would diverge from their their average values in the case of an attack and hence potentially qualify as anomaly indicators.

3.4.4. Evaluation

We measure the performance of our DDoS monitor on the basis of the F_1-score, which is defined as the harmonic mean of the precision (P) and recall (R) measures. The F_1-score gets closer to 1 when both precision and recall are close to 1, and indicating good performance; on the other hand, the F_1-score diminishes to 0, when the system performs poorly either due to low precision or low recall or both.

Table 3.2. Features collected at the SIP server.

Category	Feature	Description
SIP Requests	REGISTER	Num. of "register" requests
	INVITE	Num. of "invite" requests
	SUBSCRIBE	Num. of "subscription" requests
	NOTIFY	Num. of "notification" requests
	OPTIONS	Num. of "options" requests
	ACK	Num. of "acknowledgment" requests
	BYE	Num. of "bye" requests
	CANCEL	Num. of "cancellation" requests
	PRACK	Num. of "provisional acknowledgement" requests
	PUBLISH	Num. of "event publish" requests
	INFO	Num. of "information update" requests
	REFER	Num. of "call transfer" requests
	MESSAGE	Num. of "instant message" requests
	UPDATE	Num. of "session state update" requests
SIP Responses	100	Num. of trying responses
	180	Num. of "ringing" responses
	183	Num. of "session progress" responses
	200	Num. of "success" responses
	400	Num. of "bad request" errors
	401	Num. of "unauthorized" errors
	403	Num. of "forbidden" errors
	404	Num. of "not found" errors
	405	Num. of "not allowed" errors
	481	Num. of "dialog does not exist" errors
	486	Num. of "busy" errors
	487	Num. of "request terminated" errors
	500	Num. of "server internal" errors
	603	Num. of "decline" errors
Resource Usage	TOT_CPU	Percentage of total CPU usage
	TOT_MEM	Percentage of total virtual memory usage
Asterisk Stats	A_CPU	Percentage of CPU used by Asterisk
	MEM	Percentage of memory utilized by Asterisk
	FH	Num. of Asterisk file descriptors
	THREADS	Num. of Asterisk threads
	TCP_CONN	Num. of Asterisk TCP connections
	UDP_CONN	Num. of Asterisk UDP connections
Asterisk Logs	A_WARNING	Num. of Asterisk "warning" log messages
	NOTICE	Num. of Asterisk "notice" log messages
	VERBOSE	Num. of Asterisk "verbose" log messages
	ERROR	Num. of Asterisk "error" log messages
	DEBUG	Num. of Asterisk "debug" log messages

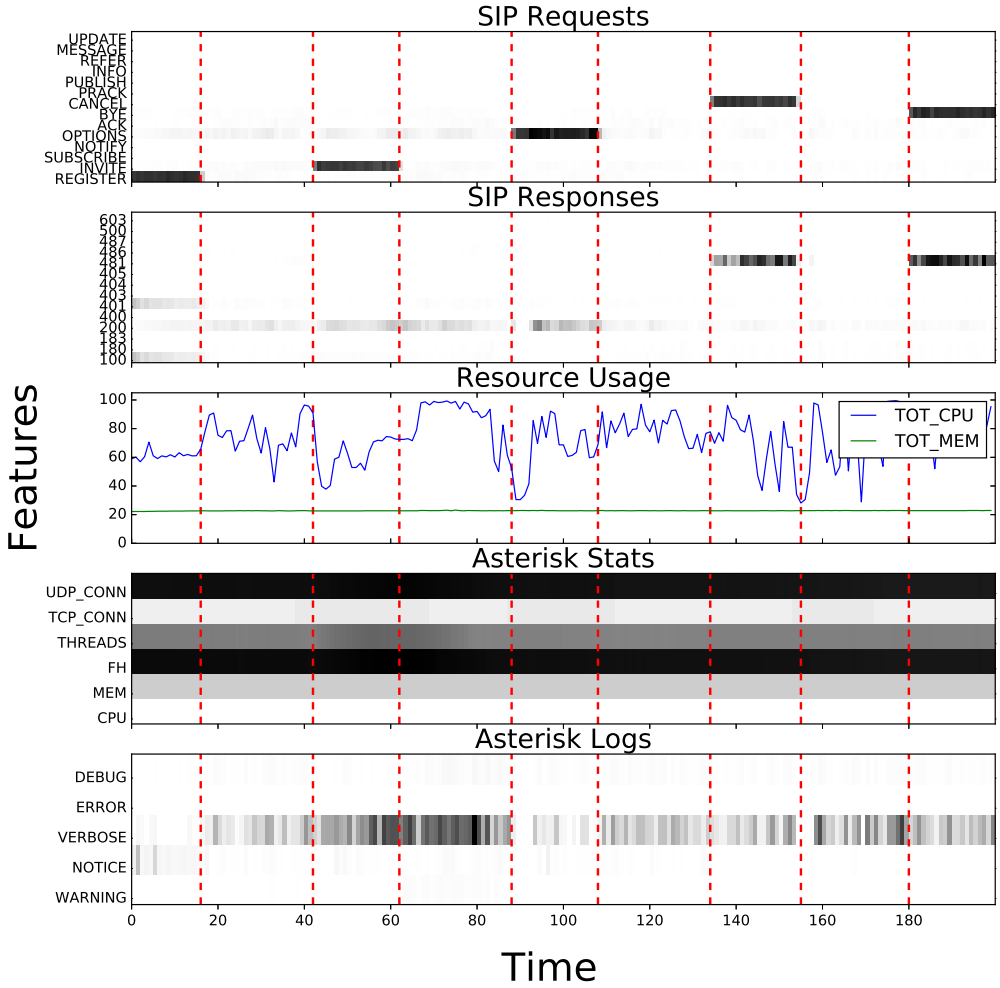


Figure 3.4. A sample histogram of features in the BCPM data.

$$F_1\text{-score} = 2 \times \frac{P \times R}{P + R} \tag{3.13}$$

$$\text{Precision (P)} = \frac{\#\text{ true alarms}}{\#\text{ alarms}} = \frac{T_a}{T_a + F_a} \tag{3.14}$$

$$\text{Recall (R)} = \frac{\#\text{ true alarms}}{\#\text{ ground truth}} = \frac{T_a}{G_a} \tag{3.15}$$

where T_a and F_a are the true alarms (true positive) and false alarms (false positive), and G_a is the true number of change points. An alarm g_t means signaling of a change event, and it is triggered whenever the change point probability in Eq. 14 at time t exceeds a certain threshold λ_a.

$$g_t = \begin{cases} 0 & \text{if } p(s_t|v_{1:t+L}) > \lambda_a \\ 1 & \text{otherwise} \end{cases} \tag{3.16}$$

The true and false alarms are calculated by matching the alarms $g_{1:T}$, with the ground truth of change events $\hat{g}_{1:T}$. In the design of our experiment, the time stamps for the beginning of attacks are manually set, but the actual effect of an attack is observed with some delay due to the combined emergent behavior of the SIP server, simulation and the vulnerability scanning tool Nova V-Spy. Therefore we set the ground truths as attack time stamps which are initially set and adjust them manually afterwards.

We declare a correct detection if the alarm g_i is within a tolerance vicinity of the corresponding ground truth event $\hat{g}_j$, that's $|i - j| < w$, and increment the number of true alarms. Alarms not matched with any ground truth are regarded as false positives.

3.5. Results

We evaluated the performance of our proposed DDoS monitor with model simulation data generated by various input feature combinations. We have tested exhaustively each of the 5 feature categories by including them or not into the into the, as appropriate, Poisson and Multinomial observation models. This resulted in a total of $3^5 - 1 = 242$ observation models.

The hyper-parameters for each observation model need to be adjusted for getting best F-scores. For this purpose, we first did a grid search inside the parameter space. Since grid search is feasible for only a limited number of parameters, we used shared parameters for the priors of the Dirichlet and Gamma distributions. In this setup, we assign a single parameter α for the Dirichlet priors by setting $w_D = [\alpha, \alpha, \ldots, \alpha]$ and a single parameter a for all Gamma priors. We also set the scale parameter of Gamma priors, $b = 1$. The search space is given in Table 3.3.

Table 3.3. Grid search space for BCPM parameters.

Parameter	Search values
α	$1, 10, 100$
a	$1, 10, 100$
π	$10^{-2}, 10^{-4}, 10^{-8}$

The configurations with the best average F_1-scores on 4 different traces after the grid search are reported in Table 3.4. It's has been observed that the *SIP Requests* contribute to the feature set for all cases, hence they seem to be the most important features collected from data. Secondly, the *Resource Usage* features helps improving the accuracy of our system. We also observe that the Dirichlet-Multinomial (DM) model, which only tracks the ratios of the features usually gives better accuracy than the Poisson-Gamma model, which tracks the magnitude of the tracked signals. We observe that the accuracy of change estimations increase provided we allow deferred change point decision by employing online smoothing with a lag value of 5 seconds, as shown in Table 3.5.

Table 3.4. BCPM filtering results.

Configurations					**F-Scores**				
SIP Requests	SIP Responses	Resource Usage	Asterisk Stats	Asterisk Logs	Low-Low	Low-High	High-Low	High-High	Average
DM		DM			0.85	0.94	0.89	0.96	0.91
PG		DM			0.91	0.94	0.79	0.98	0.90
PG					0.90	0.93	0.79	0.98	0.90
PG				DM	0.87	0.92	0.79	0.98	0.89
PG	DM				0.88	0.92	0.76	0.98	0.88
PG	DM	DM			0.87	0.93	0.75	0.98	0.88
DM					0.83	0.90	0.83	0.95	0.88
DM			DM		0.83	0.93	0.74	0.98	0.87
DM	DM	DM			0.82	0.91	0.83	0.92	0.87
Average					0.86	0.93	0.80	0.97	0.89

Table 3.5. BCPM online smoothing results.

Configurations					**F-Scores**				
SIP Requests	SIP Responses	Resource Usage	Asterisk Stats	Asterisk Logs	Low-Low	Low-High	High-Low	High-High	Average
DM		PG	DM		0.94	0.95	0.93	0.99	0.95
PG			DM		0.95	0.96	0.91	0.99	0.95
PG				DM	0.94	0.95	0.92	0.99	0.95
PG		PG	DM		0.94	0.96	0.91	0.99	0.95
PG		DM		DM	0.93	0.96	0.91	0.99	0.95
PG		PG		DM	0.94	0.95	0.89	0.99	0.94
PG	DM	PG			0.94	0.94	0.90	0.99	0.94
PG		DM			0.93	0.94	0.90	0.99	0.94
PG		PG			0.94	0.94	0.89	0.99	0.94
Average					0.94	0.95	0.91	0.99	0.95

Since the grid search may not be feasible for bigger dimensional vectors, we have also developed a maximum likelihood scheme for estimating the hyper-parameters. To this effect, we have employed the EM algorithm described in Section 3.3.1 for the case of models that attained the highest F-scores according to the grid search. We observe that the maximization of the hyper-parameters with respect to their likelihood under the proposed models does not necessarily maximize the F-scores; in other words, the F-scores obtained after the maximum likelihood estimation of hyper-parameter values are below the scores obtained by the grid search. We conjecture that this may be due to the mismatch between the model and the actual data, and will be the subject of future research.

Table 3.6. BCPM maximum likelihood parameter estimation results.

Configurations					**F-Scores**				
SIP Requests	SIP Responses	Resource Usage	Asterisk Stats	Asterisk Logs	Low-Low	Low-High	High-Low	High-High	Average
DM		PG	DM		0.80	0.78	0.85	0.81	0.82
PG			DM		0.89	0.87	0.86	0.91	0.89
PG				DM	0.79	0.77	0.81	0.83	0.80
PG		PG	DM		0.84	0.87	0.86	0.89	0.87
PG		PG		DM	0.69	0.77	0.80	0.80	0.76
PG	DM	PG			0.65	0.80	0.81	0.79	0.75
PG		DM			0.80	0.83	0.85	0.87	0.83
PG		PG			0.68	0.81	0.82	0.83	0.78
Average					0.77	0.81	0.83	0.84	0.82

4. NETWORK TRAFFIC SAMPLING AND RECONSTRUCTION

Monitoring network statistics is crucial for the maintenance and infrastructure planning for the network service providers. Statistical information about traffic patterns help a service provider to characterize its network resource usage and user behavior, infer future traffic demands, detect traffic/usage anomalies, and possibly provide insights to improve the performance of the network [1]. However, the increasingly high volume and speed of data over modern networks make measurement a difficult task, and in most cases requires specialized hardware. Sampling [2] has became a viable approach for extracting statistics from networks when data volume is huge.

In this work, we are concerned with one of the most important network statistics, the *flow length distribution* (FLD). A network flow is defined as a set of IP packets with the same signature, which we call the *key*, observed within a period of time. A flow key is defined as the IP and port pairs of both source and destination nodes together with level-3 protocol type such as TCP or UDP. A flow starts with the arrival of the first packet with the specific key and terminated when the inter-packet timeout is exceeded. The total number of packets in a flow is referred as the *flow length* and the length distribution of a set of flows that are terminated in a time window is called *flow length distribution.*

Passive measurement, where the measuring beacons inactively watch the traffic passing by, is a popular method for collecting per flow information, and our method of choice. In this method, network packets are processed at a measurement beacon, i.e a router or a dedicated hardware box and the beacon keeps a look up table for flow statistics. Whenever a packet arrives, its key is hashed and a lookup operation to the table is performed. If there is already a flow entry with the same key, its statistics is updated, otherwise a new flow is created. Once a flow is terminated, its statistics such as the number of packets, total size in bytes, the arrival of the first and last packet are

transferred to statistics collector.

The brute force *direct counting* method for maintaining a flow length histogram requires processing every packet that pass through the measurement beacon. In order to implement such a direct method, the network monitoring system needs to maintain a very large table to hold all the flow information. Each packet needs to be associated with a particular flow by hashing its packet signature and creating or updating its table entry accordingly. However, a very large amount of concurrent flows with very short packet inter-arrival times of current high speed networks (on the order of 10Gbs to 100Gbs inside carrier's network today) make this brute-force counting method very costly to implement. First of all, this method would need a huge amount of memory to record a flow table for each active flow. Secondly, in a high speed link, the inter-arrival times between packets, which may be as small as 8 ns in an OC-768 link, are smaller than the time required to process flow hash operations such as identifying the packet and updating the flow the statistics.

The characteristics of the network traffic data inevitably leads to the development of alternative methods for measurement such as random sampling, where a fraction of the network traffic is randomly selected and processed. The simplest sampling method is the *uniform packet sampling* [6–8, 66], used in commercial systems [67] and [68]. In uniform sampling, each packet is selected with a predefined constant probability. This approach is easy to implement, since it does not require flow identification of each packet. However, recovering the true flow length distribution from the random packet sampled traffic is a challenging problem. The unbiased estimator of the original length n of a flow for sampling probability π is $\hat{n}(m) = m/\pi$ where m is the number of observed packets for a given flow. The relative error of this estimator, calculated as $\sqrt{1/(\pi - 1)/n}$ [6], grows unbounded for short flows as the sampling rate gets smaller. This high error on the small flows are due to the fact that most of the samples are collected from longer flows.

Flow-based adaptive sampling methods [3–5] were proposed for more accurate flow length estimation. In these methods, each incoming packet is processed and then

sampled with probability that is a function of the current sampled length of the flow that it belongs to. The main idea is to use a smaller memory by compressing the flow statistics counters in the router. However, these methods needs to be implemented on specialized -and expensive- hardware due to the mandatory packet identification and lookup step.

Both unified and flow-based adaptive sampling methods rely on numerical methods to recover the true flow length distribution. In this work, we propose an improvement to the non-parametric flow length models in [6] and [8] using a novel NTF based technique, which we call *ThinNTF*. In our method, the network traffic is modeled as a mixture of traffic patterns. We learn those patterns directly from the sampled traffic and reconstruct original flow length distributions. We apply our method with both unified and flow-based sampling schemes. We make the following contributions for this problem:

- We model one week of flow length observations as a 3-dimensional tensor and observe the periodic behaviour.
- We propose a novel tensor factorization scheme, ThinNTF, which is able to find the factors of a latent tensor from its sampled counterpart. By doing so, we also solve the reconstruction problem.
- We apply ThinNTF to real world data sampled with two different sampling methods: unified random packet sampling and flow-based adaptive sampling.

4.1. Related Works

Sampling methods have long been applied to network traffic monitoring. A survey on fundamental network sampling strategies is given in [2]. Uniform packet sampling is extensively studied by various authors. Duffield *et al.* [6] proposes first non-parametric model for flow length distribution and provides a maximum likelihood estimation to recover the flow lengths. Riberio *et al.* [7] shows that using protocol specific information gives better flow length distribution estimates in TCP flows. Yang *et al.* [8] adopts the maximum likelihood approach to estimate both flow length and flow volume (number

of bytes in a flow) distributions. They also propose a mixture model for the network traffic, where they model the traffic as a combination of two patterns: the pattern of small flows, and the large flows.

Alternative methods to uniform packet sampling has been proposed, since packet sampling has theoretical limitations for recovering flow statistics [66]. Kumar *et al.* [69, 70] proposed two different counter update algorithms while processing every packet. They also propose a non-uniform packet sampling algorithm based on sketch counting [71]. Hu *et al.* [4, 72] proposes another non-uniform packet sampling algorithm, called *ANLS* for estimating flow lengths per each flow, and then adopts this method to flow volume [5]. We are going to use ANLS as an example non-uniform packet sampling methods in our experiments, since it is the current state-of-the-art non-uniform sampling method.

Nonnegative tensor factorization is the generalization of the NMF [28] to the multiway arrays. In NMF, a nonnegative matrix is approximated with a multiplication of two nonnegative matrices. Minimizing the Kullback-Leibler divergence between the initial matrix and multiplied factors is a popular formulation of this method, and can be solved with fixed-point iterations [73] or full Bayesian methods [29]. NMF has been used in many applications such as spectral data analysis [74], face recognition [28] and document clustering [75].

Modeling the flow length distribution as a mixture of distributions is first proposed by [8] However, according to our knowledge, modeling a large volume of flow length data as a tensor has not been experimented previously. In this thesis, we fill a gap in literature by introducing tensor factorization methodology to network monitoring.

4.2. Real World Data Collection

4.2.1. Network Flow Extraction

The method for keeping the flow records from a live network interface requires processing each packet inside the traffic. In order to extract the flow information, the network monitoring system maintain a large table to hold the flow records with data fields listed in Table 1.2. Whenever a packet is observed, its key should pass though a hash function and a table lookup must be performed. If the table already contains a flow with the packet's key, the attributes of the flow is updated, otherwise a new flow record should be generated and the packet should be registered as its first packet. The key hashing must be done independent of the order of the source and destination socket information. If a flow whose key contains source and destination socket information reversed with respect to the packet's key exits in the flow table, the packet should be registered to that flow with the direction flag set to -1.

The flow record table that we implemented for monitoring an ISP server in real time is given in Figure 4.1. This record table also has a time out mechanism to make garbage collection periodically by storing the actual flow information in a separate double linked nodes. The list keep flow records sorted according to the arrival of their last packets by constantly moving the most recently updated flow to the front of the list. The garbage collector starts removing the timed out flows starting from the end of the list. In this setting find, insert, update and delete operations are executed in amortized O(1) time.

We implemented a network sniffer that observes a live network interface and extract and save the information that is necessary to identify the flows afterwards. This network sniffer was deployed on a server of a mobile network provider in Turkey and 10 days of continuous packet information was collected. We used this data set to validate our proposed sampling recovery methods. In this section, we first give details of data extraction process and present our preliminary observations on the data.

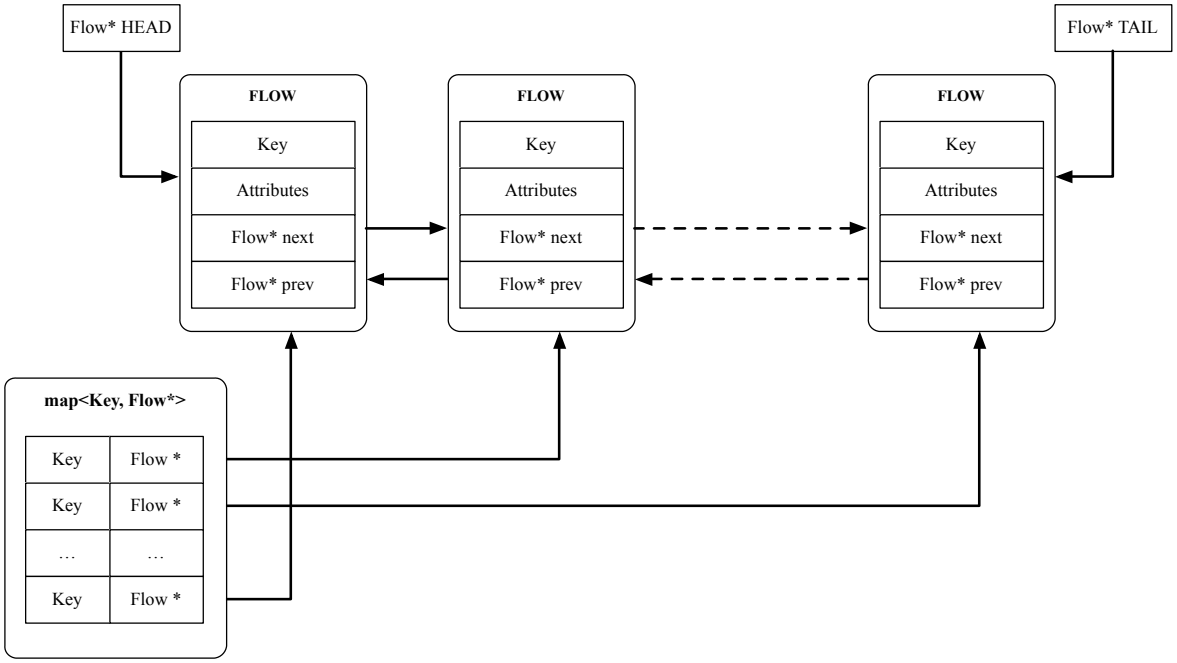


Figure 4.1. The flow table design with time-out mechanism.

4.2.2. System Architecture

The system architecture of a mobile operator's general packet radio service (GPRS) network infrastructure, including radio access and core network elements, is illustrated in Figure 4.2. IP traffic generated or received by mobile devices between mobile station (MS) and packet data network (PDN), e.g. IP Multimedia Subsystem (IMS), is tunneled in the core network of mobile operators through serving GPRS support node (SGSN) and gateway GPRS support node (GGSN) via the user data part of the GPRS tunneling protocol (GTP) [76]. The GTP message exchanges include information such as the size of the traffic, IP session start and end time, device and user identifiers.

The Gn interface [1] carries user packets to be transferred between the mobile users and the internet together with control packets necessary for the universal mobile telecommunications service (UMTS) core network [77]. All packets in this channel are carried by the GTP, which is an IP based protocol for carrying GPRS data within UMTS networks, used for data encapsulation to keep the core network independent of the protocols that are being used between MS and the packet-switched network.

[1] Gn is an interface between SGSN and GGSN where GTP is the main protocol for network packets flowing through.

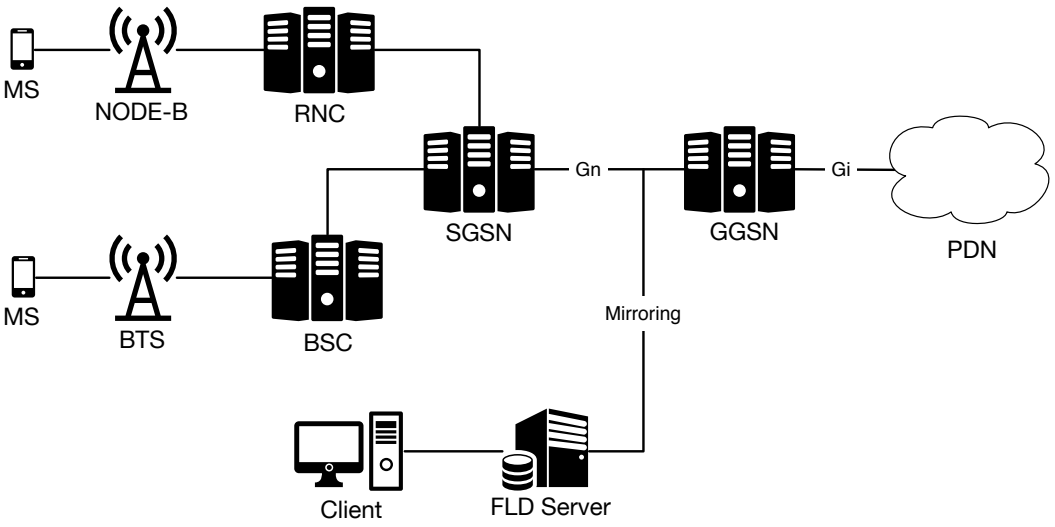
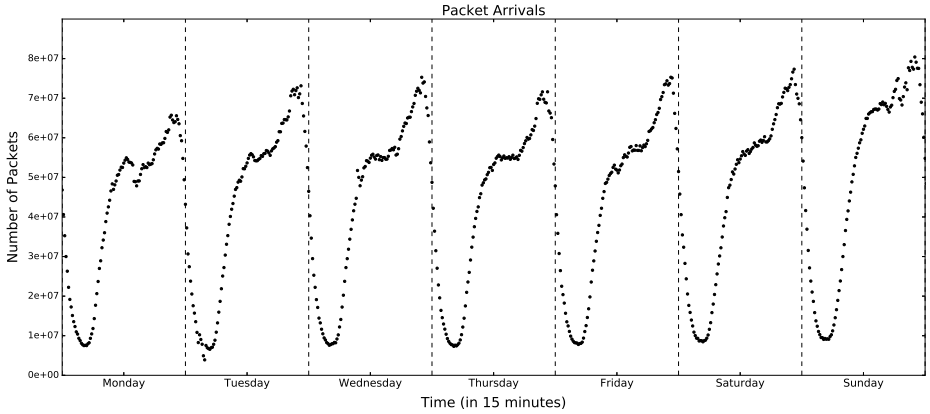


Figure 4.2. FLD server architecture.

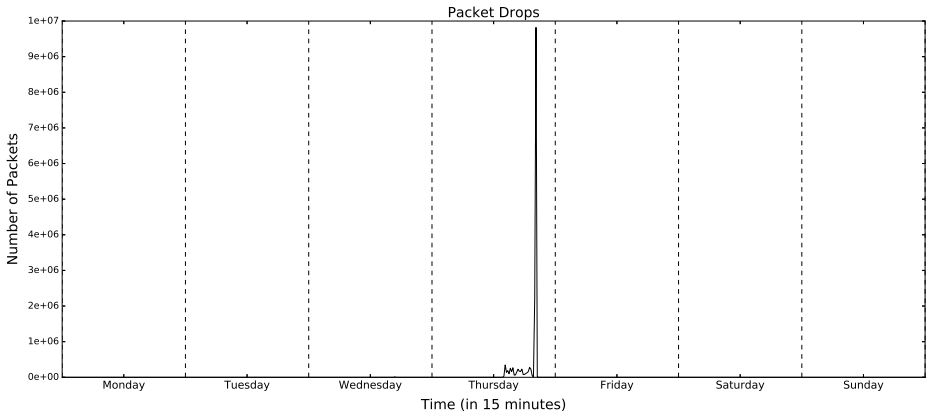
The Gn interface carries mainly two types of GTP message structures or packets: GTP-C and GTP-U. GTP-C is used for signaling between SGSN and GGSN in core network which carries packet data protocol (PDP) Context messages such as activating and deactivating user session, configuring service parameters or updating the session. GTP-U is used for transmitting user data between the radio access network and core network. In our experiments, we filtered out GTP-C packets (since they are not considered to be part of a flow due to flow definition), which makes 10% of the total Gn traffic. Therefore, the sampling is applied to GTP-U packets only. GTP is carried mainly over UDP.

4.2.3. Data Extraction process

The mobile operator's network consists of several districts with more than 10 regional core areas through-out Turkey. The average total traffic over all regional areas consists of approximately over 15 billion packets in uplink direction and over 20 billion packets in the downlink direction daily. This corresponds to approximately over 80 terabytes of total data flowing in uplink and downlink daily inside mobile operator's core network as a whole. In this work, the Gn interface which connects the SGSN and GGSN nodes is mirrored and the network traffic is transferred into a FLD server located at mobile operator's technology center in the core network. A fast speed of



(a) Packet Arrival Intensity.



(b) Packet Drops.

Figure 4.3. Network data collection statistics.

200Mbit/sec at peak hours can be observed through one of the mirrored interface in the core network.

We monitored the network traffic in one of the servers of a mobile operator continuously for 10 days and collected the packet information necessary for identifying the flow of each packet. Note that based on the mirrored interface, the size of raw data is more than 2 terabytes for observed duration of one day. If offline processing is performed on this huge amount of data, it can take several days of processing to extract the relevant headers for analysis using a single server. Therefore, this is infeasible for analyzing 10 days of traffic data both from computation and storage point of view. For this reason, in order to be able to perform analysis for long periods of days, we have devised a data collector tool inside FLD server which stores the necessary information

from each packet that is sufficient to identify the flows in the data offline. This information consists of the 5-tuple key together with the arrival time of the packet and its length, as described in Table 1.1. At the end of data extraction process, the total number of packets collected are approximately 4×10^{11}, which make up around 2.5×10^{10} flows. The total disk space required to store the packet information in compressed binary files is 389 Gigabytes. Figure 4.3 shows the number of packets collected at 15 minute intervals together with the number of packets lost during each interval. There is only a single glitch in the data collection process where 10^7 packets are lost for an unknown reason.

4.3. Data Tensor

The original flow length data is represented in an $I \times J \times K$ tensor $\mathcal{X}$, with individual elements $x_{i,j,k}$, regarded as the number of flows that has length i measured at the hour j of the day k. In this setup, I is the maximum flow length value, J is the hours of the day and K is the days of the week. For our real-world data, collected continuously for 1 week, these values are $I = 2000000, J = 24$ and $K = 7$.

Working with large maximum flow size is not feasible for two reasons. First, learning a mixture model where each flow component has 2 million parameters is not a good formulation for this problem. Secondly, 99.9% of flows in our data have less than 100 packets, which means the tensor $\mathcal{X}$ will be very sparse for $i > 100$. The clamping process can be defined as

$$\bar{x}_{i,j,k} = \begin{cases} x_{i,j,k} & \text{for } i < I_{max} \\ \sum_{l \geq I_{max}} x_{l,j,k} & \text{for } i = I_{max} \end{cases} \tag{4.1}$$

where $\bar{\mathcal{X}}$ is the clamped tensor. The clamping does not require any change in the model and inference equations that are given in Section 2.3. Therefore, for notational

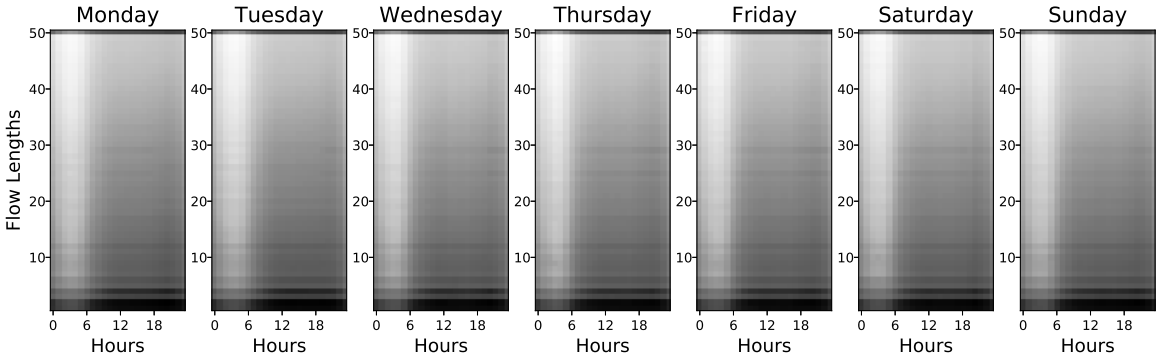


Figure 4.4. Slices of the original flow length tensor.

clarity we only use $\mathcal{X}$ as the generic original data tensor.

Figure 4.4 shows the vertical slices of our unsampled real-world data tensor $\mathcal{X}$, collected at the backbone of a mobile operator during a one week period, from Monday to Sunday, and clamped at $I_{max} = 50$. The intensity images are generated from the logarithm of the flow length counts. The daily and hourly patterns are easily recognizable in the original FLD data.

4.4. Sampling Methods

Independent of the sampling method, we can define an $I \times (I+1)$ size $\mathbf{S}$ matrix, where I is the maximum flow length with entries $s_{i,\nu}$ interpreted as the probability of sampling ν packets from an original flow of length i. Naturally, $\mathbf{S}$ is a lower diagonal matrix of the form

$$\mathbf{S} = \begin{bmatrix} s_{1,0} & s_{1,1} & 0 & 0 & \dots & 0 \\ s_{2,0} & s_{2,1} & s_{2,2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{I,0} & s_{I,1} & s_{I,2} & s_{I,3} & \dots & s_{I,I} \end{bmatrix} \tag{4.2}$$

where its l^{th} row defines a probability distribution for the sampled flow length of a flow of size l. Given a flow size distribution $\mathbf{x} \in \mathbb{Z}^{\geq I}$, where $\mathbb{Z}^{\geq}$ is the set of nonnegative integers, the expected sampled flow length distribution would be given by

$\hat{\mathbf{y}} = \mathbf{S}^T\mathbf{x}$. It immediately follows that the sampled flow size distribution y has length $I + 1$, with y_0 is the proportion of sampled flows with none of their packets sampled. During the sampling process, this value will never be observed naturally since the flow identification is performed only on selected packets. In all experiments throughout this paper, the $\mathbf{y}$ vector (or $\mathcal{Y}$ tensor, which will be described later on) will be element-wise multiplied with a binary mask vector $\mathbf{m}$ (or binary mask tensor $\mathcal{M}$), whose entries are set to 1 except the ones corresponding to zero sampled flow lengths, in order to simulate the real life scenario.

For any given sampling method, we can calculate the $\mathbf{S}$ matrix directly if a closed-form expression is available. Otherwise, it can be approximated by simulating the sampling process and counting the sampling statistics. In this paper, both uniform random sampling and the adaptive nonlinear sampling (ANLS) provide closed-form expressions for the calculation of $\mathbf{S}$ matrix.

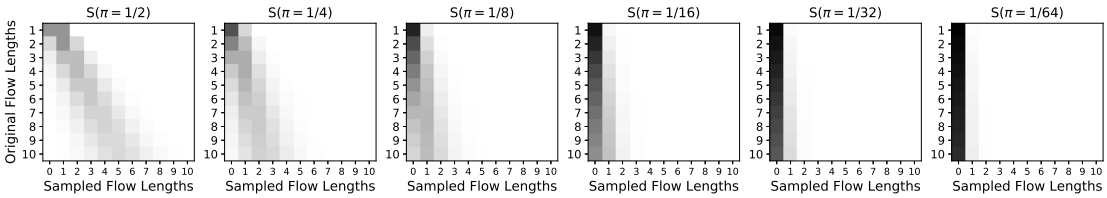
An important practical issue is that, if the original tensor $\mathcal{X}$ is clamped at I_{max}, the $\mathbf{S}$ matrix must also be clamped. In that case a last row entry $s_{I_{max},\nu}$ must present the probability of selecting ν packets from a flow of length greater or equal to I_{max}. This clamping operation can be done by calculating a full size $\mathbf{S}$ matrix first, and setting $\bar{s}_{I_{max},\nu} \propto \sum_{i=I_{max}}^{I} s_{i,\nu}$ with a naive assumption that after I_{max} the original flow sizes are uniformly distributed.

4.4.1. Uniform Sampling Method

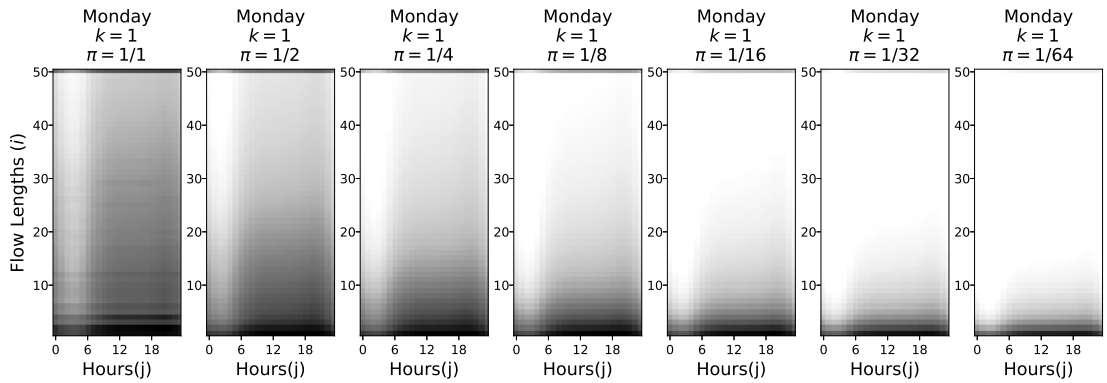
In uniform sampling, each packet is processed with a fixed probability of π, irrespective of the flow it belongs to. In this method, the sampling matrix entries $s_{i,\nu}$ are calculated directly through Binomial distribution with i trials and π success probability, i.e.,

```
1: function SampleUniformlyRandom(π, flow_table, packet)
2:     if π > rand_double(0, 1) then
3:         flow = flow_table.lookup(packet)
4:         if flow is null then
5:             flow = new Flow(packet)
6:         else
7:             flow.length += 1
8:         flow_table.insert_or_update(flow)
```

Figure 4.5. Uniform packet sampling algorithm.



(a) Uniform sampling matrices.



(b) Uniformly sampled Monday data with varying sampling probabilities.

Figure 4.6. Uniform sampling.

$$s_{i,\nu} = \begin{cases} \binom{i}{\nu}\pi^{\nu}(1-\pi)^{i-\nu} & \text{for } \nu \leq i \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

Figure 4.5 describes how the flow table is updated with uniform sampling upon the arrival of a new packet. The algorithm uniformly draws a random number in

interval $[0, 1]$ and if it is less than π, processes the packet, otherwise the packet is discarded. For processing the packet, a look-up operation is performed on the flow table to find and update the flow that the packet belongs to. If no such flow is found, a new flow is created using the packet's signature.

Figure 4.6(a) shows the top 10×11 entries of the lower diagonal $\mathbf{S}$ matrices with different sampling probabilities. As the sampling probability π gets smaller, fewer packets from a flow gets observed, and the flow may even be missed when none of its packets are observed. The rightmost sampling matrix shows the case when $\pi = 1/64$, where the matrix has a very high concentration of zero-length sampled flows.

Figure 4.6(b) shows the original Monday data (the leftmost matrix) and its sampled versions under uniform sampling with the probabilities shown on the top sampling matrices. Here we see that for $\pi = 1/64$, the observed flow lengths are mostly less than 10, while the majority are not observed at all.

4.4.2. ANLS Sampling Method

The ANLS [4] will be used as the representative of the flow-based adaptive sampling methods. In ANLS, each packet is sampled according to the number of packets previously sampled from its corresponding flow. If a sampled flow has length x, the probability of its next packet to be sampled ($p(x; u)$) is calculated as

$$f(x; u) = [(1 + u)^x - 1]/u \tag{4.4}$$

$$p(x; u) = 1/[f(x - 1; u) - f(x; u)] \tag{4.5}$$

Here, $f(x; u)$ is a monotonically increasing function of flow length x, parametrized with u, which makes $p(x; u)$ monotonically decreasing. The u parameter determines

```
1: function SetupANLS(u)
2:     f[0] = 0
3:     for i ∈ [1, I] do
4:         f[i] = ((1 + u)^i − 1)/u
5:         p[i] = 1/(f[i − 1] − f[i])
6:     return f, p
7: function SampleWithANLS(p, flow_table, packet)
8:     flow = flow_table.lookup(packet)
9:     if flow is null then
10:        flow = new Flow(packet)
11:    else if p[flow.length] > rand_double(0, 1) then
12:        flow.length += 1
13:    flow_table.insert_or_update(flow)
```

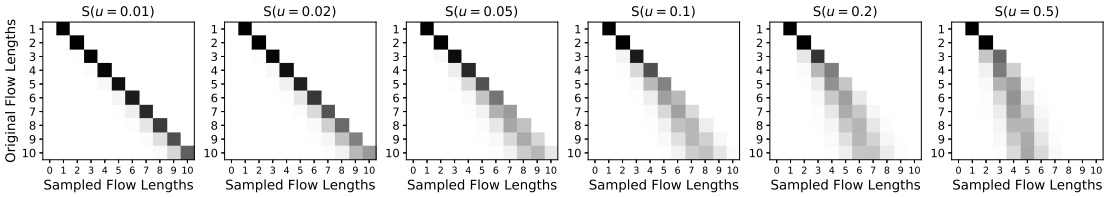
Figure 4.7. ANLS sampling algorithm.

the tendency of sampling packets. As u gets smaller, more packets are sampled and estimating original flow lengths gets easier.

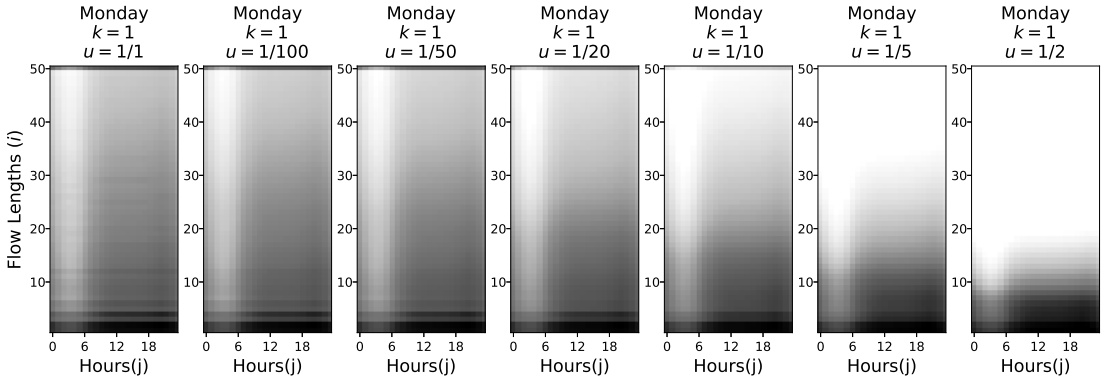
The ANLS method is described in details in Figure 4.7. Prior to the sampling, the **f** and **p** vectors are calculated in the SetupANLS function, according to equations 4.4 and 4.5. During sampling, for each incoming packet a look-up operation is performed unconditionally. If the corresponding flow is found, it is updated with probability relative to its current observed size. Otherwise, a new flow is created, ensuring that every flow is observed with at least one packet.

We calculate the sampling matrix **S** recursively for ANLS. In this method, the first packet is always sampled since $p(1, u) = 1$ independent of u. We start by assigning all zero sampling probabilities as $s_{:,0} = 0$ and $s_{1,1} = 1$. The recursive equation for calculating the sampling matrix can be deduced as

$$s_{i,\nu} \propto s_{i-1,\nu-1} p(i-1, u) + s_{i-1,\nu}(1 - p(i-1, u)) \tag{4.6}$$



(a) ANLS sampling matrices.



(b) ANLS sampling with varying u.

Figure 4.8. ANLS sampling.

Figure 4.8 shows the ANLS sampling matrices for first 10 flow lengths and the Monday data sampled with them respectively, similarly to Figure 4.6. First, we can see that when u is small, the $\mathbf{S}$ matrix looks like identity and as u gets larger, the sampling probability of large flows decreases. Secondly, compared to uniform sampling, the ANLS method has much higher sampling ratios than uniform sampling. However, operating with such high sampling ratios would require specialized hardware in real time.

4.5. Experiments and Results

We designed two sets of experiments in order to verify our model: synthetic and real-world experiments. In each set, we sampled the original data with both uniform and ANLS models with different sampling parameters. Then we tried to recover the original tensor with ThinNTF models. The ThinNTF model takes a single parameter R which is the number of components in each factor. Additionally, we also represented data as $I \times JK$ matrix by unfolding the $\mathcal{X}$ tensor in the first dimension as described in [15], and applied the 2-dimensional version of the ThinNTF, which we simply call

thin nonnegative matrix factorization (TNMF). For Uniform sampling, we used the maximum likelihood estimation (MLE) defined in [6] as the baseline. For ANLS, we used both MLE and its own unbiased estimator of the model as baselines.

Both ThinNMF and ThinNTF models explain the data as a linear combination of R flow length distributions, stored in the columns of $\mathbf{F}$ matrix. In the ThinNMF model, we have JK coefficient sets for this combination. On the other hand, in ThinNTF, we have J coefficients for hour-of-day and K coefficients for day-of-week. The Cartesian product of these coefficient sets make a total of JK coefficients and creates a dependency between the hour and day attributes. Therefore, we expect that ThinNTF captures the periodicity and give better estimates.

During the experiments, we always run the stochastic algorithms, i.e. ThinNMF, ThinNTF, and MLE, for 10 times and keep the parameters of the model with the highest lower bound value. Then we reported the success of our algorithm with the weighted mean relative distance (WMRD) metric as this was used in all previous flow size estimation works. The WMRD is a metric which gives more weights to the relative differences that occur with larger frequency. It is formulated as

$$\text{wmrd}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{\sum_i |x_i - \hat{x}_i|}{\sum_i (x_i + \hat{x}_i)/2} \tag{4.7}$$

where $\mathbf{x}$ is an original flow size distribution measured at the end of the hour and $\hat{\mathbf{x}}$ is its estimated version. For the whole tensor $\mathcal{X}$, we calculate the average WMRD value.

Additionally, we report the Kullback-Leibler divergence between the original and the estimated tensors, since this is the metric minimized during the variational Bayes algorithm. The KL divergence between two distributions $\mathbf{x}$ and $\hat{\mathbf{x}}$ is calculated as

$$\text{KL}(\mathbf{x}, \hat{\mathbf{x}}) = \sum_i x_i \log\left(\frac{\hat{x}_i}{x_i}\right) \tag{4.8}$$

Table 4.1. Uniform sampling results on synthetic data.

Period	ThinNMF-R3	ThinNTF-R3	MLE
2	0.53	0.49	0.88
4	0.63	0.59	1.20
8	0.65	0.61	1.29
16	0.68	0.61	1.41
32	0.74	0.61	1.37
64	0.85	0.61	1.50

4.5.1. Experiments on Synthetic Data

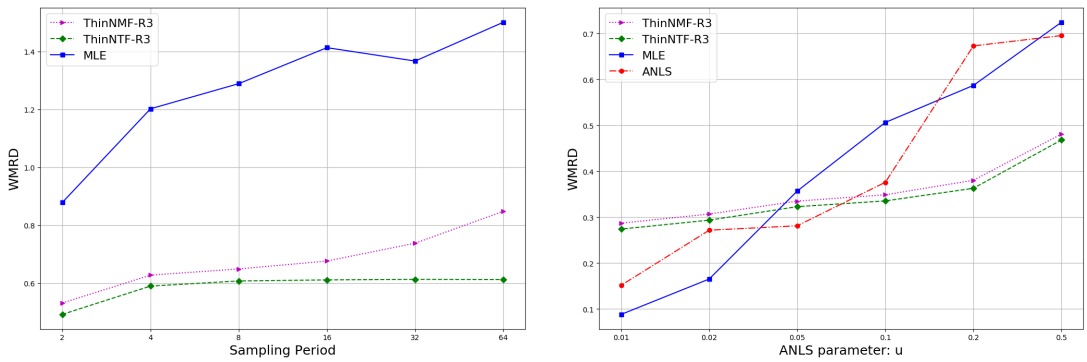
We prepared our synthetic experiments to test the validity of our models. In this experiment set, we used the generative model of the ThinNTF model as described in Figure 2.5 to generate a small network with maximum flow size $I = 10, J = 24$ and $K = 7$. The original synthetic flow length distribution $\mathcal{X}$ is generated by a generative model with 3 components, where each component is a column in the $\mathbf{F}$ factor. We selected these 3 components as exponential, inverted exponential, and uniform random distributions. Therefore, in experiments, we used ThinNMF-R3 and ThinNTF-R3 models, where the suffix R3 shows that the model has 3 components.

We sampled the synthetic data with uniform and ANLS sampling methods with different sampling parameters. The sampling was done simply by randomly drawing a sampled size for each flow according to the sampling probabilities in the $\mathbf{S}$ matrix. By this way, we ignored the flow splitting problem and this gave us an ideal data for the ThinNTF model. We report and compare the mean standard deviation of these WMRD values for all experiments.

The ThinNTF model always performed best with the uniform sampling model, as shown in Table 4.1 as expected. On ANLS sampling, the MLE and the ANLS estimators performed better with high sampling probabilities, when $u \in (0.01, 0.02, 0.05)$, as shown in Table 4.2. On the other hand, when the sampling probability of the ANLS model decreases, the ThinNMF helped with better estimations. From the initial re-

Table 4.2. ANLS sampling results on synthetic data.

U	ThinNMF-R3	ThinNTF-R3	MLE	ANLS
0.01	0.29	0.27	0.09	0.15
0.02	0.31	0.29	0.17	0.27
0.05	0.33	0.32	0.36	0.28
0.1	0.35	0.34	0.51	0.38
0.2	0.38	0.36	0.59	0.67
0.5	0.48	0.47	0.72	0.70



(a) Uniform Sampling. (b) ANLS Sampling.

Figure 4.9. Synthetic experiment results.

sults, we conclude that the factorization is definitely helpful for more difficult uniform sampling method and helps lower the sampling probabilities in flow-based packet sampling. The results are also visible in Figure 4.9.

4.5.2. Experiments on Real World Data

The original data collected from a mobile network provider as we described in Section 4.2, is sampled with both sampling methods. However, this time we simulated the real network offline by feeding the packets one by one to the monitoring server, as described in Section 4.2, This way, we were able to create the actual conditions on a sampler installed at a network provider's backbone. This also created the flow splitting problem, since we applied a 30 seconds time-out in our flow hash. We set the maximum flow length as $I = 100$, meaning that $\mathcal{X}_{100,:,:}$ entries show the count of flows that have more than 99 packets. This clamping decision was made according to the

Table 4.3. Uniform sampling results on real data.

Period	ThinNMF			ThinNTF			MLE
	R=2	R=3	R=4	R=2	R=3	R=4	
2	0.23	0.24	0.23	0.21	0.25	0.22	0.41
4	0.55	0.52	0.53	0.50	0.48	0.49	0.69
8	0.94	0.93	0.94	0.91	0.90	0.87	0.97
16	1.15	1.11	1.11	1.09	1.05	1.04	1.05
32	1.25	1.24	1.24	1.16	1.13	1.10	1.22
64	1.31	1.29	1.30	1.09	1.06	1.04	1.22

cumulative distribution of flow lengths as shown in Figure 4.10 We also clamped the sampling matrices $\mathbf{S}$ so that they exactly match the model.

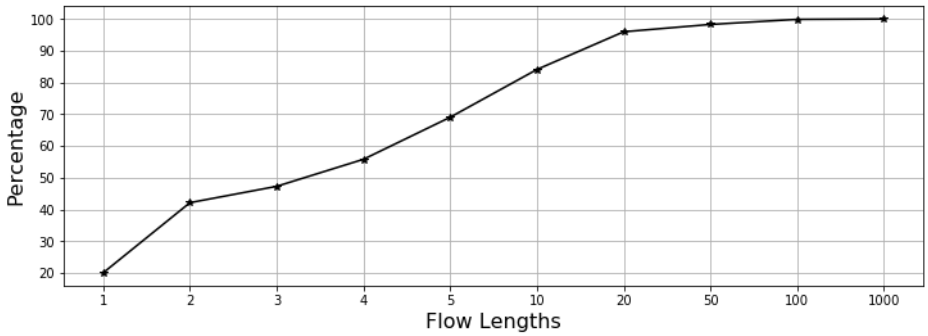


Figure 4.10. Cumulative flow lengths in the real world data.

Since the number of components in the original flow distribution is unknown, we run our experiments with $R \in [2, 3, 4]$ components for ThinNMF and ThinNTF. The rest of the experiment is similar to the synthetic one. The sampled Y matrix with shape $100 \times 24 \times 7$ is factorized and $\hat{\mathcal{X}}$ is reconstructed with the estimated factors. We reported the mean and standard deviation of 24×7 WMRD and KL values.

The factorization models, both ThinNMF and ThinNTF helped lower the WMRD score in both uniform and ANLS sampling methods. ThinNTF-R4 model consistently gave lower error than the MLE baseline for uniform model as shown in Table 4.3 and Figure 4.11. Indeed, our factorization framework improved results overall for uniform sampling. However, since recovering true estimates in uniform sampling is quite difficult, we see less impact of the factorization as the sampling ratio increases.

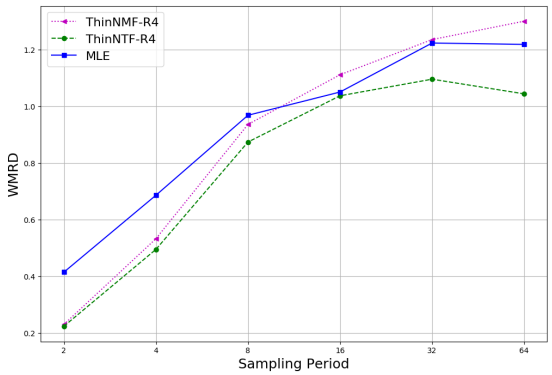
Table 4.4. ANLS sampling results on real data.

U	ThinNMF			ThinNTF			MLE	ANLS
	R=2	R=3	R=4	R=2	R=3	R=4		
0.01	0.03	0.02	0.01	0.05	0.04	0.03	0.05	0.12
0.02	0.04	0.03	0.02	0.06	0.04	0.03	0.08	0.21
0.05	0.04	0.03	0.02	0.07	0.05	0.04	0.13	0.39
0.1	0.06	0.05	0.04	0.08	0.07	0.05	0.17	0.61
0.2	0.08	0.08	0.08	0.10	0.09	0.07	0.21	0.70
0.5	0.13	0.13	0.11	0.16	0.15	0.13	0.33	0.94

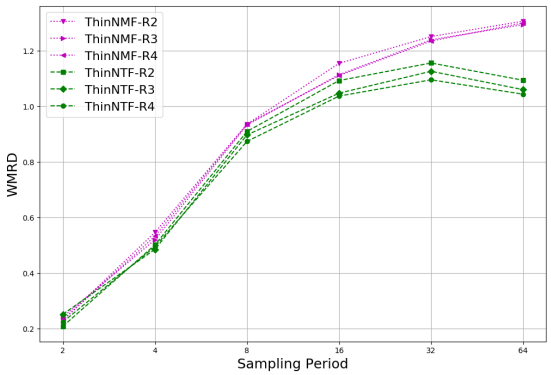
Figure 4.11 also gives the KL values between the true and estimated flow length distributions. While the scale of this metric is different, it gives consistent results with the WMRD. This shows that our model, which minimizes the KL metric also minimizes the commonly used WMRD metric, hence the model is suitable for this problem.

Another important issue is that for uniform sampling, 3-way factorization is more successful than the 2-way factorization. The periodicity information which is captured by the ThinNTF model help improve the estimates and makes it a more successful model for this sampling method.

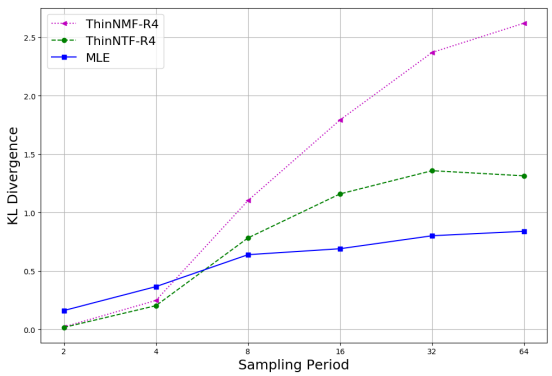
In ANLS, all our factorization models gave lower error values than the MLE and unbiased estimator of ANLS as shown in Table 4.4 and Figure 4.12. Since ANLS is a more powerful sampling method than uniform sampling, our the effect framework is slightly less for small sampling parameter u. However, both ThinNMF and ThinNTF gave better result while sampling smaller number of packets (when u is large). Furthermore, since we are trying to recover the same original data in both experiments, we can compare our ThinNMF and ThinNTF models under two sampling methods. We see that in both methods as the number of components increases, the models gave lower error rates. However, with the uniform sampling method, 3-dimensional methods give better results, while with ANLS, 2-dimensional models perform slightly better.



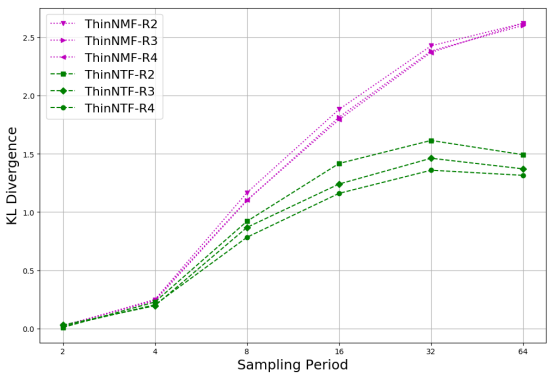
(a) WMRD metrics.



(b) WMRD metrics wrt. R.



(c) KL metrics.



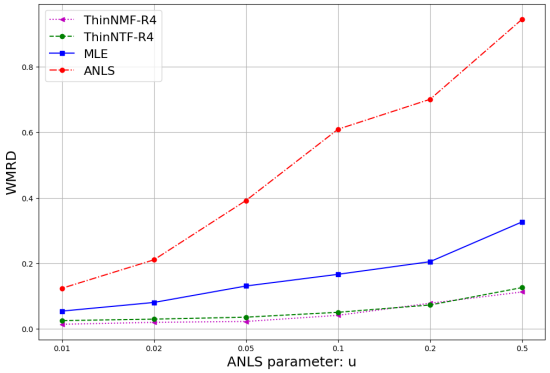
(d) KL metrics wrt. R.

Figure 4.11. Real world data results with uniform sampling.

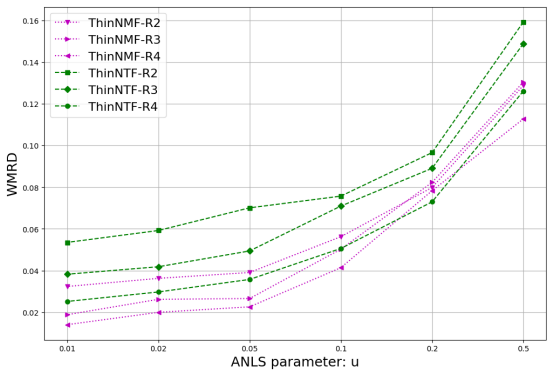
4.5.3. Effect of Clamping

The choice of where to clamp the data can be given by multiple factors. First of all, one can set the clamping value I_{max} according to a value of special interest. Otherwise, we would like to choose a small I_{max} so that we deal with a dense tensor and we deal with less parameters. On the other hand, we would like to set I_{max} as high as possible so that the clamped portion of the data is as small as possible.

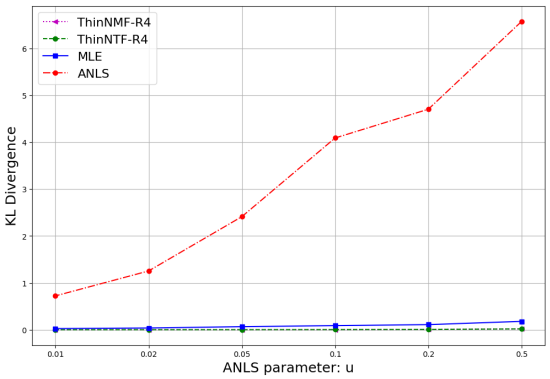
We run the best algorithms found in previous section for uniform and ANLS sampling methods with $I_{max} \in \{25, 50, 75, 100\}$. The WMRD values are given in Figure 4.13. In both methods, $I_{max} = 25$ gave relatively poor performance and $I_{max} = 100$ was generally the best choice. Also the results with $I_{max} \geq 50$ are closer to each other. This is consistent with the graph in Figure 4.10, where the cumulative flow lengths do not change much after $I_{max} = 50$. A final remark from this experiment is



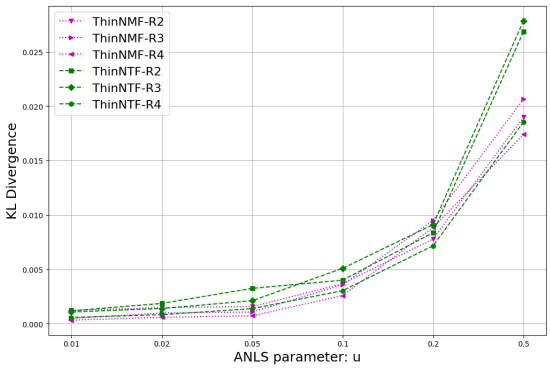
(a) WMRD metrics.



(b) WMRD metrics wrt. R.



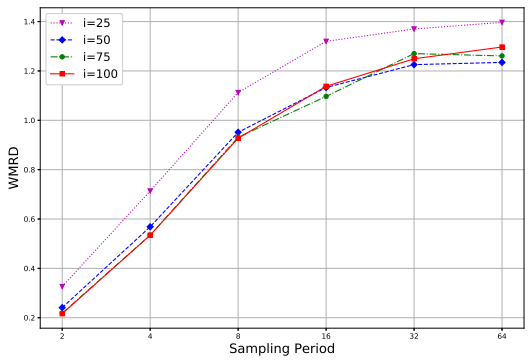
(c) KL metrics.



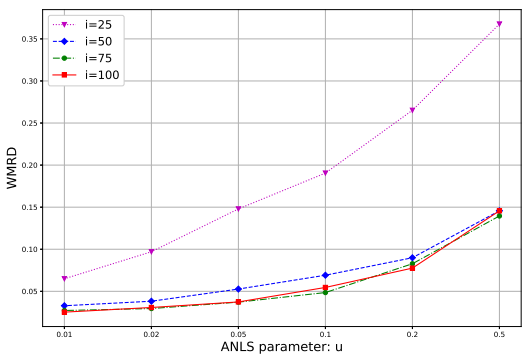
(d) KL metrics wrt. R.

Figure 4.12. Real world data results with ANLS sampling.

that as the clamping value increases estimation becomes harder with small sampling rates. This explains the results in uniform sampling with sampling rate 1/64.



(a) Uniform Sampling ThinNMF-R4.



(b) ANLS Sampling ThinNTF-R4.

Figure 4.13. Clamping experiment results.

An important issue left as future work is the online execution of the ThinNTF model. Theoretically, the ThinNTF model can be used online once sufficient data from the target network is collected and the flow length distribution components, ie. the **F** factor, are inferred. The power of our model is that this inference can be done directly from the sampled observations. Once the **F** factor is estimated, for each incoming observation the corresponding entries in other factors can be inferred by keeping **F** constant during the inference. Moreover, **F** can be updated periodically, say weekly, in a sliding window fashion and kept up to date with the networks flow length behavior.

5. CONCLUSIONS

In this thesıs, we proposed a Bayesian multiple change point model for detecting DDoS flooding attacks in VoIP networks using SIP as their signaling mechanism. We provided a framework that can work with different types of input features monitored at a SIP proxy server. The framework tracks the network traffic and SIP server behavior and raises an alarm whenever a change in those behaviors is detected.

Additionally, we propose a probabilistic SIP network simulation system, which can generate real-time SIP conversations in a community. The simulation system provides a realistic test environment for SIP security tools, in case of the absence of real world test data. This simulation system is also made available in open source. We tested our proposed system with traffic generated by the SIP network simulator together with DDoS attacks generated by a commercial network vulnerability scanning tool, Nova-VSpy.

We introduced a novel nonnegative tensor factorization model called ThinNTF, which extends the classic nonnegative tensor factorization with an additional constant factor that can represent a network packet sampling method. We showed that this model can be employed to improve the current reconstruction algorithms in recovering the original flow length distributions. We tested our model with two different type of sampling methods: the uniform packet sampling method and the ANLS, which is a flow-based packet sampling method. We described how to use these methods by showing how to build their sampling matrices.

In order to test ThinNTF model, we collected high-volume data from a mobile network provider for a long period in order to observe the periodical behavior of the flow length distribution. In experiments on synthetic and real-world data, our models gave promising results by lowering the estimation errors compared to the baselines of each sampling method. We conclude that our model can be used to decrease estimation errors or to decrease the sampling probabilities without increasing the estimation error.

REFERENCES

1. Varghese, G. and C. Estan, "The measurement manifesto", *ACM SIGCOMM Computer Communication Review*, Vol. 34, No. 1, p. 9, 2004.

2. Duffield, N., "Sampling for Passive Internet Measurement: A Review", *Statistical Science*, Vol. 19, No. 3, pp. 472–498, 2004.

3. Kumar, A., M. Sung, J. J. Xu and E. W. Zegura, "A Data Streaming Algorithm for Estimating Subpopulation Flow Size Distribution", *ACM SIGMETRICS Performance Evaluation Review*, Vol. 33, No. 1, pp. 61–72, 2005.

4. Hu, C., B. Liu, S. Wang, J. Tian, Y. Cheng and Y. Chen, "ANLS : Adaptive Non-Linear Sampling Method for Accurate Flow Size Measurement", *IEEE Transactions on Communications*, Vol. 60, No. 3, pp. 789–798, 2012.

5. Hu, C., B. Liu, H. Zhao, K. Chen, Y. Chen, Y. Cheng and H. Wu, "Discount Counting for Fast Flow Statistics on Flow Size and Flow Volume", *IEEE/ACM Transactions on Networking*, Vol. 22, No. 3, pp. 970–981, 2014.

6. Duffield, N., C. Lund and M. Thorup, "Estimating Flow Distributions From Sampled Flow Statistics", *IEEE/ACM Transactions on Networking*, Vol. 13, No. 5, pp. 933–946, 2005.

7. Ribeiro, B. F., D. F. Towsley, T. Ye and J. Bolot, "Fisher information of sampled packets: an application to flow size estimation", *Internet Measurement Conference*, pp. 15–26, 2006.

8. Yang, L., G. Michailidis, A. Arbor and G. Michailidis, "Sampled Based Estimation of Network Traffic Flow Characteristics", *INFOCOM 2007, 26th IEEE International Conference on Computer Communications*, pp. 1775–1783, 2007.

9. Rosenberg, J., H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, H. M. and E. Schooler, "SIP: Session Initiation Protocol", Technical Report, 2002, `www.ietf.org/rfc/rfc3261.txt`.

10. Keromytis, A. D. and S. Member, "A Comprehensive Survey of Voice over IP Security Research", *IEEE Communications Surveys & Tutorials*, pp. 514–537, 2012.

11. Mirkovic, J. and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms", *SIGCOMM Computuer Communications Review*, Vol. 34, No. 2, pp. 39–53, 2004.

12. Sisalem, D., J. Kuthan and S. Ehlert, "Denial of service attacks targeting a SIP VoIP infrastructure: Attack scenarios and prevention mechanisms", *IEEE Network*, Vol. 20, No. 5, pp. 26–31, 2006.

13. Kurt, B., E. Zeydan, U. Yabas, I. A. Karatepe, G. K. Kurt and A. T. Cemgil, "A Network Monitoring System for High Speed Network Traffic", *13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2016.

14. Kurt, B., A. T. Cemgil, G. K. Kurt and E. Zeydan, "Estimation of Flow Length Distributions UsingBayesian Nonnegative Tensor Factorization", *Wireless Communications and Mobile Computing*, Vol. 2019, 2019, `https://doi.org/10.1155/2019/8458016`.

15. Kurt, B., Ç. Yıldız, T. Y. Ceritli, B. Sankur and A. T. Cemgil, "A Bayesian change point model for detecting SIP-based DDoS attacks", *Digital Signal Processing*, Vol. 77, pp. 48 – 62, 2018.

16. Yıldız, Ç., M. Semerci, Y. T. Ceritli, B. Kurt, A. T. Cemgil and B. Sankur, "Change Point Detection for Monitoring SIP Networks", *European Conference on Networks and Communications*, 2016.

17. Yıldız, Ç., Y. T. Ceritli, B. Kurt, B. Sankur and A. T. Cemgil, "Attack Detection in VOIP Networks Using Bayesian Multiple Change-Point Models", *24th IEEE Conference on Signal Processing and Communications Applications (SİU)*, 2016.

18. Kurt, B., Ç. Yıldız, Y. T. Ceritli, M. Yamaç, M. Semerci, A. T. Cemgil and B. Sankur, "A Probabilistic SIP Network Simulation System", *24th IEEE Conference on Signal Processing and Communications Applications (SİU)*, 2016.

19. Yıldız, Ç., B. Kurt, T. Y. Ceritli, B. Sankur and A. T. Cemgil, "A Real-Time SIP Network Simulation and Monitoring System", *Elsevier Software-X*, 2017.

20. Dempster, A., N. Laird and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society Series B Methodological*, Vol. 39, No. 1, pp. 1–38, 1977.

21. Fearnhead, P., "Exact and efficient Bayesian inference for multiple changepoint problems", *Statistics and Computing*, Vol. 16, No. 2, pp. 203–213, 2006.

22. Barber, D., *Bayesian Reasoning and Machine Learning*, Cambridge University Press, 2012.

23. Hitchcock, F. L., "The expression of a tensor or a polyadic as a sum of products", *J. Math. Phys*, Vol. 6, No. 1, pp. 164–189, 1927.

24. Douglas Carroll, J. and J.-J. Chang, "Analysis of Individual Differences in Multidimensional Scaling via an N-way Generalization of Eckart-Young Decomposition", *Psychometrika*, Vol. 35, pp. 283–319, 01 1970.

25. A. Harshman, R., "Foundations of the PARAFAC procedure: Model and conditions for an"explanatory"multi-mode factor analysis", *UCLA Work. Pap. Phon.*, Vol. 16, 11 1969.

26. Bro, R., "PARAFAC. Tutorial and applications", *Chemometrics and Intelligent Laboratory Systems*, Vol. 38, No. 2, pp. 149 – 171, 1997.

27. Johnson, N., A. Kemp and S. Kotz, *Univariate Discrete Distributions*, Wiley Series in Probability and Statistics, Wiley, 2005.

28. Lee, D. D. and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization.", *Nature*, Vol. 401, pp. 788–91, 1999.

29. Cemgil, A. T., "Bayesian Inference for Nonnegative Matrix Factorisation Models.", *Computational intelligence and neuroscience*, Vol. 2009, 2009.

30. Ermis, B. and A. T. Cemgil, "A Bayesian Tensor Factorization Model via Variational Inference for Link Prediction.", *ArXiv*, 2014.

31. Blei, D. M., A. Kucukelbir and J. D. McAuliffe, "Variational Inference: A Review for Statisticians", *Journal of the American Statistical Association*, Vol. 112, No. 518, pp. 859–877, 2017.

32. Vavasis, S., "On the Complexity of Nonnegative Matrix Factorization", *SIAM Journal on Optimization*, Vol. 20, No. 3, pp. 1364–1377, 2010.

33. Geneiatakis, D., T. Dagiuklas, G. Kambourakis, C. Lambrinoudakis, S. Gritzalis, K. S. Ehlert and D. Sisalem, "Survey of Security Vulnerabilities in Session Initiation Protocol", *Commun. Surveys Tuts.*, Vol. 8, No. 3, pp. 68–81, 2006.

34. Eddy, W., *TCP SYN Flooding Attacks and Common Mitigations*, Tech. rep., IETF, Internet Engineering Task Force, 2007, `https://tools.ietf.org/html/rfc4987`.

35. Sanders, J., *Chinese government linked to largest DDoS attack in GitHub history*, 2015, `http://www.techrepublic.com/article/chinese-government-linked-to-largest-ddos-attack-in-github-history/`, accessed in April 2019.

36. Korolov, M., *DDoS attack on BBC may have been biggest in history*, 2016, `http://www.csoonline.com/article/3020292/cyber-attacks-espionage/ddos-attack-on-bbc-may-have-been-biggest-in-history.html`, accessed in April 2019.

37. Corero Network Security, I., *DDoS Trends Report*, 2017, `http://info.corero.com/RP-2017-DDoS-Trends_LP-Homepage-Banner.html`, accessed in April 2019.

38. Verisign, *Verisign DDoS Trends Report*, 2017, `https://www.verisign.com/en_US/security-services/ddos-protection/ddos-report/index.xhtml`, accessed in April 2019.

39. Cooney, M., *IBM Warns of Rising Voip Cyber Attacks*, 2016, `http://www.networkworld.com/article/3146095/security/ibm-warns-of-rising-voip-cyber-attacks.html`, accessed in April 2019.

40. Chang, R. K., "Defending Against Flooding-based Distributed Denial-of-service Attacks: A Tutorial", *IEEE Communications Magazine*, Vol. 40, No. 10, pp. 42–51, 2002.

41. Peng, T., C. Leckie and K. Ramamohanarao, "Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems", *ACM Computing Surveys*, Vol. 39, No. 1, 2007.

42. Ehlert, S., D. Geneiatakis and T. Magedanz, "Survey of network security systems to counter SIP-based denial-of-service attacks", *Computers & Security*, Vol. 29, No. 2, pp. 225–243, 2010.

43. Iancu, B., "SER PIKE", Technical Report, 2003, `http://www.iptel.org/ser/doc/modules/pike`.

44. Wu, Y. S., S. Bagchi, S. Garg, N. Singh and T. Tsai, "SCIDIVE: A stateful and cross protocol intrusion detection architecture for voice-over-IP environments", *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 433–442, 2004.

45. Chen, E., "Detecting DoS attacks on SIP systems", *1st IEEE Workshop on VoIP Management and Security*, pp. 2–7, 2006.

46. Chandola, V., A. Banerjee and V. Kumar, "Anomaly detection", *ACM Computing Surveys*, Vol. 41, No. 3, pp. 1–6, 2009.

47. *Nova V-SPY*, 2016, `http://www.netas.com.tr/en/innovation-productization/nova-cyber-security-products/`, accessed in April 2019.

48. Sengar, H., D. Wijesekera, H. Wang and S. Jajodia, "VoIP intrusion detection through interacting protocol state machines", *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 393–402, 2006.

49. Bouzida, Y. and C. Mangin, "A framework for detecting anomalies in VoIP networks", *Third International Conference on Availability, Reliability and Security*, pp. 204–211, 2008.

50. Ehlert, S., G. Zhang, D. Geneiatakis, G. Kambourakis, T. Dagiuklas, J. Markl and D. Sisalem, "Two layer Denial of Service prevention on SIP VoIP infrastructures", *Computer Communications*, Vol. 31, No. 10, pp. 2443 – 2456, 2008.

51. Ormazabal, G., S. Nagpal, E. Yardeni and H. Schulzrinne, *Secure SIP: A Scalable Prevention Mechanism for DoS Attacks on SIP Based VoIP Systems*, pp. 107–132, Springer Berlin Heidelberg, 2008.

52. Nassar, M., R. State and O. Festor, "Monitoring SIP Traffic Using Support Vector Machines", *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection*, pp. 311–330, 2008.

53. Akbar, M. A. and M. Farooq, "Application of Evolutionary Algorithms in Detection of SIP Based Flooding Attacks", *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pp. 1419–1426, 2009.

54. Akbar, M. A. and M. Farooq, "Securing SIP-based VoIP infrastructure against flooding attacks and Spam Over IP Telephony", *Knowledge and Information Systems*, Vol. 38, No. 2, pp. 491–510, 2014.

55. Tsiatsikas, Z., A. Fakis, D. Papamartzivanos, D. Geneiatakis, G. Kambourakis and C. Kolias, "Battling Against DDoS in SIP - Is Machine Learning-based Detection an Effective Weapon?", *Proceedings of the 12th International Conference on Security and Cryptography - Volume 1: SECRYPT, (ICETE 2015)*, pp. 301–308, 2015.

56. Tsiatsikas, Z., D. Geneiatakis, G. Kambourakis and S. Gritzalis, "Realtime DDoS Detection in SIP Ecosystems: Machine Learning Tools of the Trade", *Network and System Security*, 2016.

57. Reynolds, B. and D. Ghosal, "Secure IP Telephony using Multi-layered Protection", *Network and Distributed System Security Symposium*, 2003.

58. Rebahi, Y. and D. Sisalem, "Change-point detection for voice over ip denial of service attacks", *Communication in Distributed Systems (KiVS), ITG-GI Conference*, pp. 1–7, 2007.

59. Zhang, H., Z. Gu, C. Liu and T. Jie, "Detecting VoIP-specific Denial-of-Service Using Change-Point Method", *2009 11th International Conference on Advanced Communication Technology*, pp. 1059–1064, 2009.

60. Geneiatakis, D., N. Vrakas and C. Lambrinoudakis, "Utilizing Bloom Filters for Detecting Flooding Attacks Against SIP Based Services", *Comput. Secur.*, Vol. 28, No. 7, pp. 578–591, 2009.

61. Sengar, H., H. Wang, D. Wijesekera and S. Jajodia, "Fast Detection of Denial-of-Service Attacks on IP Telephony", *IWQoS*, 2006.

62. Tsiatsikas, Z., D. Geneiatakis and G. Kambourakis, "Exposing Resource Consumption Attacks in Internet Multimedia Services", *Proceedings of 14th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), Security Track*, pp. 1–6, 2014.

63. Handley, M. and V. Jacobson, *SDP: Session Description Protocol*, Tech. rep., IETF, Internet Engineering Task Force, 1998, `https://www.ietf.org/rfc/rfc2327.txt`.

64. Schulzrinne, H., S. Casner, R. Frederick and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*, Tech. rep., IETF, Internet Engineering Task Force, 2003, `https://tools.ietf.org/html/rfc3550`.

65. *Trixbox*, 2016, `http://www.fonality.com/trixbox`, accessed in April 2019.

66. Hohn, N. and D. Veitch, "Inverting sampled traffic", *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pp. 222–233, ACM Press, 2003.

67. *Cisco NetFlow*, 2017, `http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html`, accessed in April 2019.

68. *nTop*, 2017, `http://www.ntop.org/`, accessed in April 2019.

69. J.Xu, A. M., J. Wang, A. Kumar, M. Sung, J. J. Xu and J. Wang, "Data streaming algorithms for efficient and accurate estimation of flow size distribution", *ACM SIGMETRICS Performance Evaluation Review*, Vol. 32, No. 1, pp. 177–188, 2004.

70. Kumar, A., J. Jimxu and J. Wang, "Space-Code Bloom Filter for Efficient Per-Flow Traffic Measurement", *IEEE Journal On Selected Areas In Communications*, Vol. 24, No. 12, pp. 2327–2339, 2006.

71. Kumar, A. and J. J. Xu, "Sketch Guided Sampling Using On-Line Estimates of Flow Size for Adaptive Data Collection", *Proc. 2006 IEEE INFOCOM*, 2006.

72. Hu, C., S. Wang, J. Tian, B. Liu, Y. Cheng, C. Yan and Y. Chen, "Accurate and Efficient Traffic Monitoring Using Adaptive Non-Linear Sampling Method", *2008 IEEE INFOCOM - The 27th Conference on Computer Communications*, pp. 26–30, 2008.

73. Lee, D. D. and H. S. Seung, "Algorithms for Non-negative Matrix Factorization", *Advances in Neural Information Processing Systems 13*, pp. 556–562, 2001.

74. Berry, M. W., M. Browne, A. N. Langville, V. P. Pauca and R. J. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization", *Computational Statistics & Data Analysis*, Vol. 52, No. June 2006, pp. 155–173, 2007.

75. Xu, W., X. Liu and Y. Gong, "Document Clustering Based On Non-negative Matrix Factorization", *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 267–273, 2003.

76. 3GPP, *3GPP Evolved Packet System (EPS); Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C); Stage 3*, Tech. rep., 3GPP TS 29.274 13.4.0, 2015.

77. Springer, A. and R. Weigel, *The Umts (Universal Mobile Telecom Standard) Physical Layer Basics,Standard,and Frontend Matters*, Springer-Verlag, Berlin, Heidelberg, 2002.

78. Minka, T. P., "Estimating a Dirichlet distribution", Technical Report, 2003.

79. Minka, T. P., "Estimating a Gamma Distribution", Technical Report, 2002.

80. Goldenberg, A., "A Survey of Statistical Network Models", *Foundations and Trends in Machine Learning*, Vol. 2, No. December, pp. 129–233, 2010.

APPENDIX A: STANDARD DISTRIBUTIONS

The probability mass or density functions and sufficient statistics of the standard distributions used in this thesis are listed below.

Binomial.

$$\mathcal{B}(x; n, p) = \binom{x}{n} p^x (1-p)^{n-x} \tag{A.1}$$

$$E(x) = np \tag{A.2}$$

$$Var(x) = np(1-p) \tag{A.3}$$

Multinomial.

$$\mathcal{M}(\mathbf{x}; n, \mathbf{p}) = \binom{n}{x_1, \ldots x_K} \prod_{k=1}^{K} p_k^{x_k} \tag{A.4}$$

$$E[x_k] = np_k \tag{A.5}$$

$$Var[x_k] = np_k(1-p_k) \tag{A.6}$$

Beta.

$$\mathcal{B}e(x; a, b) = \frac{\Gamma(a+b)}{\Gamma(a) + \Gamma(b)} x^{a-1}(1-x)^{b-1} \tag{A.7}$$

$$E[x] = \frac{a}{a+b} \tag{A.8}$$

$$E[\ln x] = \psi(a) - \psi(a+b) \tag{A.9}$$

Dirichlet.

$$\mathcal{D}(\mathbf{x};\alpha) = \frac{\Gamma(\sum_{k=1}^{K}\alpha_k)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}x_k^{\alpha_k-1} \tag{A.10}$$

$$E[x_i] = \frac{\alpha_i}{\sum_{k=1}^{K}\alpha_k} \tag{A.11}$$

$$E[\ln x_i] = \psi(\alpha_i) - \psi(\sum_{k=1}^{K}\alpha_k) \tag{A.12}$$

Poisson.

$$\mathcal{P}(x;\lambda) = \frac{\lambda^x e^{-\lambda}}{x!} \tag{A.13}$$

$$E[x] = x \tag{A.14}$$

Gamma.

$$\mathcal{G}(x;a,b) = \frac{1}{\Gamma(a)b^a}x^{a-1}e^{-\frac{x}{b}} \tag{A.15}$$

$$E[x] = ab \tag{A.16}$$

$$E[\ln x] = \psi(a) + \ln(b) \tag{A.17}$$

APPENDIX B: BCPM PARAMETER ESTIMATION

Here we present the detailed equations of the Expectation-Maximization method for estimating model parameters.

B.1. E-Step

At the E-step, we calculate the expectation of the complete data log-likelihood as follows

$$\mathcal{L}_{\pi,\mathbf{w}} = \log p(s^{(1:T)}, \mathbf{h}^{(0:T)}, \mathbf{v}^{(1:T)}, |\pi, \mathbf{w}) \tag{B.1}$$

$$\begin{aligned} &= \sum_{t=1}^{T} \Big[[s^{(t)} = 0] \big(\log(1-\pi) + \log \delta(\mathbf{h}^{(t)} - \mathbf{h}^{(t-1)}) \big) \\ &\quad + [s^{(t)} = 1] \big(\log \pi + \log \Omega(\mathbf{h}^{(t)}; \mathbf{w}) \big) + \log \Theta(\mathbf{v}^{(t)}; \mathbf{h}^{(t)}) \Big] \\ &\quad + \log \Omega(\mathbf{h}^{(0)}; \mathbf{w}) \end{aligned} \tag{B.2}$$

We can write the expectation of this complete likelihood under the auxiliary distribution $q(z) = p(s^{(1:T)}, \mathbf{h}^{(0:T)} | \mathbf{v}^{(1:T)}, \pi, w)$ as follows, by letting $p(s^{(0)} = 1 | \mathbf{v}^{(1:T)}, \pi, \mathbf{w}) = 1$ in a compact way as

$$\begin{aligned} \langle \mathcal{L}_{\pi,\mathbf{w}} \rangle q(z) = \sum_{t=0}^{T} \Big[&\big\langle s^{(t)} = 1 \big\rangle q(z) \Big(\log \pi + \Big\langle \log \Omega(\mathbf{h}^{(t)}; \mathbf{w}) \Big\rangle q(z) \Big) \\ &+ \big\langle s^{(t)} = 0 \big\rangle q(z) \log(1-\pi) + \Big\langle \log \Theta(\mathbf{v}^{(t)}; \mathbf{h}^{(t)}) \Big\rangle q(z) \Big] \end{aligned} \tag{B.3}$$

B.2. M-Step

We maximize the expectation of the complete data log-likelihood with respect to the parameters π and w.

B.2.1. Maximizing for π

Maximizing the expectation of the complete data log likelihood for π is the same for all prior and observation models.

$$0 = \frac{\partial \langle \mathcal{L}_{\pi,\mathbf{w}} \rangle q(z)}{\partial \pi} \tag{B.4}$$

$$= \frac{\partial}{\partial \pi} \sum_{t=1}^{T} \Big(\big\langle s^{(t)} = 1 \big\rangle q(z) \log \pi + \big\langle s^{(t)} = 0 \big\rangle q(z) \log(1-\pi) \Big) \tag{B.5}$$

$$\pi = \frac{1}{T} \sum_{t=1}^{T} \big\langle s^{(t)} = 1 \big\rangle q(z) \tag{B.6}$$

B.2.2. Maximizing for w

Maximizing the expectation of the complete data log likelihood for w depends on the observation and prior distributions. We begin by taking the derivative

$$0 = \frac{\partial \langle \mathcal{L}_{\pi,\mathbf{w}} \rangle q(z)}{\partial \mathbf{w}} \tag{B.7}$$

$$= \frac{\partial}{\partial \mathbf{w}} \sum_{t=1}^{T} \big\langle s^{(t)} = 1 \big\rangle q(z) \Big\langle \log \Omega(\mathbf{h}^{(t)}; \mathbf{w}) \Big\rangle q(z) \tag{B.8}$$

In this work, we used a coupled model where the signals are assumed to be generated by a Poisson-Gamma or Multinomial-Dirichlet models, hence in our case $w = [\alpha_{1:M}, a_{1:N}, b_{1:N}]$, where $\alpha_{1:M}$ is the parameter of the Dirichlet prior and each $\{a_i, b_i\}$ pair is the parameter for a Gamma prior. For Dirichlet priors,

$$0 = \frac{\partial}{\partial \alpha_k} \sum_{t=0}^{T} \big\langle s^{(t)} = 1 \big\rangle q(z) \big\langle \log Dir(s^{(t)}; \alpha) \big\rangle q(z) \tag{B.9}$$

$$= \frac{\partial}{\partial \alpha_k} \sum_{t=0}^{T} \big\langle s^{(t)} = 1 \big\rangle q(z) \Bigg[\sum_k (\alpha_k - 1) \Big\langle \log \mathbf{h}_k^{(t)} \Big\rangle q(z) + \log \Gamma(\sum_k \alpha_k) - \sum_k \log \Gamma(\alpha_k) \Bigg] \tag{B.10}$$

$$= \sum_{t=0}^{T} \big\langle s^{(t)} = 1 \big\rangle q(z) \Bigg(\Big\langle \log \mathbf{h}_k^{(t)} \Big\rangle q(z) + \psi(\sum_k \alpha_k) - \psi(\alpha_k) \Bigg) \tag{B.11}$$

We can solve the above equation for α_k with fixed-point iterations [78]:

$$\alpha_k^{new} = \psi^{-1} \Big(\frac{1}{C} \sum_{t=0}^{T} \big\langle s^{(t)} = 1 \big\rangle q(z) \Big\langle \log \mathbf{h}_k^{(t)} \Big\rangle q(z) + \psi(\sum_k \alpha_k^{old}) \Big) \tag{B.12}$$

where $C = \sum_t \big\langle s^{(t)} = 1 \big\rangle q(z)$. We calculate the $\Big\langle \log \mathbf{h}_k^{(t)} \Big\rangle q(z)$ from the mixture of Dirichlet potentials as

$$\Big\langle \log \mathbf{h}_k^{(t)} \Big\rangle q(z) = \frac{1}{Z} \sum_{r=1}^{R} c^{(r)} \left(\psi(\alpha_k^{(r)}) - \psi(\alpha_0^{(r)}) \right) \tag{B.13}$$

where R is the number of potentials and $Z = \sum_r c^{(r)}$ is the sum of their normalizing constants.

For the Gamma distribution, the maximization leads to the following fixed-point iteration for a and equation for b [79]:

$$a^{new} = \psi^{-1} \left(\frac{1}{C} \sum_{t=0}^{T} \big\langle s^{(t)} = 1 \big\rangle q(z) \Big(\Big\langle \log \mathbf{h}_k^{(t)} \Big\rangle q(z) - \log \Big\langle \mathbf{h}_k^{(t)} \Big\rangle q(z) \Big) + \log a^{old} \right) \tag{B.14}$$

$$b = \frac{1}{a^{new}} \sum_{t=0}^{T} \big\langle s^{(t)} = 1 \big\rangle q(z) \Big\langle \mathbf{h}^{(t)} \Big\rangle q(z) \tag{B.15}$$

The expected sufficient statistics for the Gamma parameters are calculated from mixtures of Gamma potentials by:

$$\begin{aligned}\left\langle \mathbf{h}^{(t)} \right\rangle q(z) &= \frac{1}{Z} \sum_{r=1}^{R} c^{(r)} (a^{(r)} b^{(r)}) \\ \left\langle \log \mathbf{h}^{(t)} \right\rangle q(z) &= \frac{1}{Z} \sum_{r=1}^{R} c^{(r)} \left(\psi(a^{(r)}) - \log(b^{(r)}) \right)\end{aligned} \tag{B.16}$$

again, R is the number of potentials and $Z = \sum_r c^{(r)}$ is the sum of their normalizing constants.

APPENDIX C: BCPM FORWARD-BACKWARD HELPER ROUTINES

C.1. Backward Filtering Loop

```
1: function BackwardFilter(π, w, v_{1:T})
2:     beta ← [ ]
3:     prior ← [w, 0]
4:     for t = T ... 1 do
5:         obs_pot ← Obs2Pot(v_t)
6:         // Change case
7:         tmp_msg ← [  prior*pot for pot in beta[end]  ]
8:         new_msg ← [ obs_pot.w, log π + LogLikelihood(tmp_msg) ]
9:         // No change case
10:        no_change ← [  obs_pot*pot for pot in beta[end]  ]
11:        for pot in no_change: pot[2]+=log(1 − π) do
12:            // Update beta
13:            for pot in no_change: new_msg.append(pot) do
14:                beta.append(new_msg)
15:    return beta
```

Figure C.1. BCPM backward filtering loop.

C.2. Forward Backward Recursion Functions

```
1: function Predict(alpha, π, w)
2:     alpha_p ← [ ]
3:     alpha_p.append(w, LogLikelihood(alpha) + log(π))
4:     for pot in alpha: alpha_p.append(pot[1], pot[2]+log(1 − π))
5:     return alpha_p

6: function Update(msg, v_t)
7:     msg_u ← [ ]
8:     obs_pot = ← Obs2Pot(v_t)
9:     msg_u = [pot*obs_pot for pot in msg]
10:    return msg_u

11: function ComputeCPP(msg, d)
12:    // Change case is represented by the mixture of first d potentials
13:    p1 = exp(LogLikelihood(msg[1 : d])
14:    p0 = exp(LogLikelihood(msg[d + 1 :])
15:    return p1 /(p0 + p1)

16: function Smooth(alpha, beta)
17:    gamma ← [ ]
18:    for i = 1 . . . len(alpha) do
19:        for j = 1 . . . len(beta) do
20:            gamma.append(alpha[i] * beta[j])
21:    return gamma
```

Figure C.2. BCPM forward-backward recursion functions.

APPENDIX D: SIP NETWORK SIMULATION

Here we describe the generative process underlying the Boun-Sim network simulator.

D.1. Social Network of SIP Users

In order to simulate a realistic network, we modeled the relationships between users by a stochastic block model [80]. In a stochastic graph model, graph nodes are distributed over groups and the probability of having an between two nodes are governed by inner and between group connection parameters. Similarly we distributed N SIP users over K social groups. Whenever a user decides to make a call, they pick a callee within their group or from other groups with different probabilities. As a concrete example, we can think of SIP users inside a company, divided into different departments. A user may talk to their teammates more often then people from other teams.

At the beginning of the simulation, the generative model first generates the social network. The probability that a user belongs to a certain group, π is drawn from a Dirichlet distribution with a parameter α. Relative group sizes can be adjusted by the *alpha* parameter. We represent the group assignments of users by an $N \times K$ dimensional binary $\mathbf{G}$ matrix, where each row $\mathbf{G}_{n,:}$ represents a user such that $g_{n,k} = 1$ if and only if user n belongs to group k.

$$\pi \sim \mathcal{D}ir(\pi; \alpha) \tag{D.1}$$

Then each user is assigned to a group from a categorical distribution.

$$\mathbf{G}_{n,:} \sim \mathcal{K}(\mathbf{G}_{n,:}; \pi) \quad \text{for each } n \in [1, N] \tag{D.2}$$

Every stochastic block model has a $K \times K$ parameter matrix $\mathbf{B}$, such that $b_{i,j}$ is the probability of having an edge between a node from group i to some other node in group j. We created the $\mathbf{B}$ matrix such that inner group communications are more probable than between group communications by setting the Beta distribution parameters as $a > b$:

$$b_{i,i} \sim \mathcal{B}eta(b_{i,i}; a, b) \quad \forall i \in [1, K] \tag{D.3}$$

$$b_{i,j} \sim \mathcal{B}eta(b_{i,j}; b, a) \quad \forall i \neq j \tag{D.4}$$

D.2. Phone Book of SIP Users

After the social network is constructed, the simulator creates a phone book for each user, according to the users social behavior. Let $\mathbf{P}$ be an $N \times N$ phone book matrix, such that $p_{m,n}$ denotes the probability that user m calls user nm whenever m decides to call someone.

$$p_{m,n} \propto \prod_{i,j} b_{i,j}^{g_{m,i} g_{n,j}} \tag{D.5}$$

or, in compact matrix notation,

$$\mathbf{P} \propto \mathbf{GBG}^T \tag{D.6}$$

D.3. Registration of Users

The simulation starts with all users offline. A user becomes online by sending a *REGISTER* request to the SIP server. Each user waits a random amount of time before registering to the server. Let r_n denote the amount of time user n waits offline.

We generate this waiting time from a Gamma distribution:

$$r_n \sim \mathcal{G}(\beta_i; \rho, \phi) \tag{D.7}$$

D.3.1. Call Rates

A users call rate is governed by the time they wait idle after communications. Whenever user n becomes idle, that is after registration or end of a call, a random waiting time t_n is sampled from an exponential distribution, and makes a random call at the end of this time period. However, they can choose to answer an incoming call during this period.

$$t_n|\beta_n \sim \mathcal{E}(t_n; \beta_n) \tag{D.8}$$

Here, β_n is the call rate parameter of user N, is sampled for every user from a Gamma distribution

$$\beta_i \sim \mathcal{G}(\beta_i; k, \theta) \tag{D.9}$$

Hence, in addition to their social behavior, each user has a different personal behavior.

D.4. Making a Call

At the end of their waiting time, a user initiates a call by randomly selecting another user from their phone book. Let c_n denote the callee that user n is about to call We draw c_n from a categorical distribution

$$c_n|\mathbf{P}_{n,:} \propto \mathcal{K}(c_n; \mathbf{P}_{n,:}) \tag{D.10}$$

such that $\mathbf{P}_{n,:} \propto \{p_{n,1}, \dots p_{n,N}\}$ is the normalized probability vector for user n, where $p_{n,m}$ is the probability that user n calls m. When the user n selects the callee, they send *INVITE* request to start a conversation.

D.5. Responding to a Call

Whenever a user receives a call, they can accept or reject the call, and as a third option, may not notice the call at all. Each user has three call response parameters, f_n, a_n and h_n, such that f_n is the call notice parameter, a_n call accepting parameter and h_n is the call hold parameter.

$$f_n \sim \mathcal{U}(f_n; f_{min}, f_{max}) \tag{D.11}$$

$$a_n \sim \mathcal{U}(a_n; a_{min}, a_{max}) \tag{D.12}$$

$$h_n \sim \mathcal{U}(h_n; h_{min}, h_{max}) \tag{D.13}$$

Whenever a user receives a call, they notice the call with probability f_n and, if they do notice, accepts the call with probability a_n. If the user is already on another call, they can put their ongoing call on hold with probability h_n and accepts the call.

D.6. Call Durations

If a call is successfully established, both participants draw their own call duration, and at the end of this duration, terminate the call.

$$d_n|\delta_n \sim \mathcal{E}(d_n; \delta_n) \tag{D.14}$$

$$d_m|\delta_m \sim \mathcal{E}(d_m; \delta_m) \tag{D.15}$$

$$d_c = \min(d_n, d_m) \tag{D.16}$$

Here, d_n and d_m denotes the call durations sampled by call participants n and m and d_c is the actual call duration.