Imperial College London
Department of Mathematics

A Bayesian Approach to Human Behaviour Modelling in Computer Networks

Henry E. Clausen

CID: 01280376

Supervised by Prof. Niall Adams and Dr. Mark Briers

30th October 2017

Submitted in partial fulfilment of the requirements for the MSc in Statistics of Imperial College London

The work contained in this thesis is my own work unless otherwise stated.

Signed: Date:

Abstract

Sophisticated cyber attackers often move through their targeted network by creating a hierarchical chain of controlled computers and thereby create a strong correlation between the activity on two different computers. To detect such connected activities, an accurate model of network traffic that distinguishes between automated computer traffic and different user activity states is necessary. In this work, we propose a novel Bayesian model that identifies different states of activity in the arrival times of network flow events on individual computers. Our model is based on the well-known *Markov modulated Poisson process*, but overcomes its drawbacks with the modelling of network data. Our model is embedded in a fast and scalable Bayesian inference framework. We validate the relation of our results to actual user activity through a controlled experiment, and additionally demonstrate the performance of our framework on flow data from 10 personal computers inside the LANL enterprise network.

Acknowledgements

I would like to thank Professor Niall Adams and Dr. Mark Briers for their expert advice, excellent technical guidance, and encouragement throughout this project and beyond, it was a pleasure to work with both of them. I also would like to thank Mr. Andy Thomas for his general help and the collection of the network flow data. Finally, I would like to express my gratitude for the exceptional support and encouragement my parents provided during my studies.

Table of Contents

Notation table

Throughout this work, we will use the following notation:

$[0, t_{\text{obs}}]$	time window of observations
$\{X_t\}$	Continuous-time Markov process that governs the MMPP
M	dimension of the state space of $\{X_t\}$
$\mathbf{Q}$	Generator matrix of $\{X_t\}$
$\boldsymbol{\lambda}$	Poisson rates of the MMPP
ν	$\ker(\mathbf{Q})$
$t'_0 = 0$	
$t'_{n+1} = t_{obs}$	
$\{t'_1, \ldots, t'_1\}$	arrival times of events generated by the MMPP
$\{\Delta t'_1, \ldots, \Delta t'_{n+1}\}$	inter-arrival times of events generated by the MMPP
$\{I_1, \ldots, I_n\}$	accumulation intervals
$\{t_0, \ldots, t_n\}$	times defining the accumulation intervals
$\{z_1, \ldots, z_n\}$	Binned observations
$\{c_1, \ldots, c_n\}$	Binned sessions, a latent variable
y_k	Number of observed events generated from t'_k
λ_Y	Poisson rate of y

1. Introduction

Computer usage can be diverse, and human induced activity on a computer is not constant, but varies in correspondence to the particular task conducted on that computer. In this work, we present a Bayesian framework that models a personal computer's network traffic in order to quantify different states in its usage. For this, we will develop a new hierarchical model based on the *Markov Modulated Poisson Process* that identifies temporal patterns in the arrival of network flow events, and relates them to a latent discrete process which represents the device state. Motivation for this work stems primarily from current interests in cyber-defence, and our inference method is intended to be a critical building block on which a broader cyber-security system would be based. Moreover, this work is heavily related to current procedures in network modelling, for which it might be of future interest.

1.1. Relation to cyber-security

In the wake of devastating personal information leaks, concerns over cyber-security are at an all-time high. Sophisticated data breaches such as the attack on *JP Morgan Chase* in 2014 affect hundreds of million customers and inflicts tremendous financial, reputational, and logistic damage (Walters, 2014). Cyber-security incidents increased by 38% in 2017, and the global cost of cyber crime is estimated to reach \$2 trillion by 2019 (Conteh and Royer, 2016). The prevention of cyber crime is therefore a globally demanded necessity.

One reason for the recent rise of cyber crime is the increased use of sophisticated techniques for the attack of specific targets. Attackers use customised social engineering and custom-build malware to pass common security frameworks. Existing solutions to commercial intrusion detection in computer networks are often based on detecting signatures of previously uncovered and analysed attacks. Examples of such signatures include file hashes[1] of malicious software, blacklisted IP addresses and domain names, and characteristics of known Command-and-Control (C&C) protocols. Detection of a signature usually indicates an imminent intrusion and triggers investigation.

Adjusting existing attack procedures in order to shed previously identified signatures is simple: A file hash can be altered by minor modifications in the program and IP and domain addresses can be switched by changing servers. A sophisticated attack will employ new, customized protocols and software that is fitted to the targeted computer infrastructure, and thus will not show any previously identified signatures.

[1]A hash function encodes a file with a basic data structure into a number or string, which is known as the file hash. Every file is uniquely identifiable with its file hash.

A relatively new approach to intrusion detection is based on the accurate reflection of normal behaviour in a computer network. When anomalous behaviour is observed, alternative hypotheses can be formed that reflects attack behaviour. An intrusion is therefore treated as a cumulation of improbable events. Such events can include previously unobserved edges between computers, new processes in combination with E-mail clicks, or failed network logins. To build such a framework, accurate models of several key characteristics of computer network dynamics are necessary in order to capture the aspects that separate regular from irregular behaviour.

Pivoting and its relation to regular computer usage

Advanced persistent threats (APTs), which are responsible for many severe data breaches, are characterised by the intruder maintaining a presence in the compromised network for long-term control and data collection, possibly lasting months (Tankard, 2011). ATP attacks often go undetected for a significant amount of time as attackers adopt a slow, stealthy approach to evade detection while constantly interacting with the system under attack.

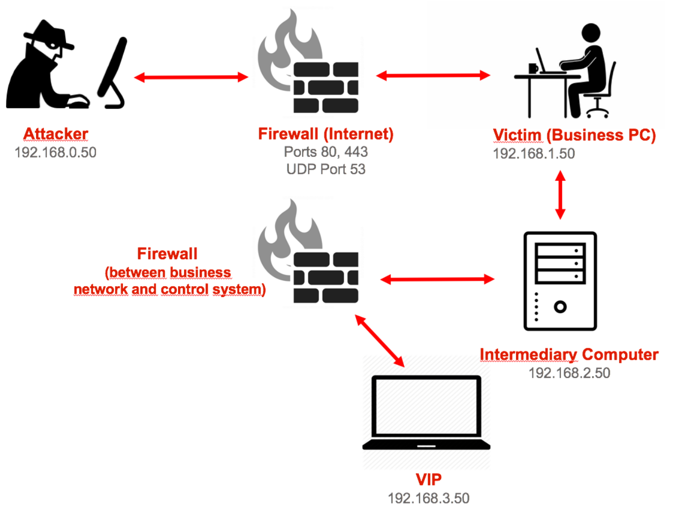

Figure 1.1.: A typical pivoting attack chain. The attacker penetrates the network at a weak point and moves through the network to find its target.

ATP attacks often circumvent strong firewalls which protect high-value assets, through a technique called *pivoting*. This allows the attacker to expand their access from vulnerable computers to ones with higher protection. An attacker often gains a foothold on a system via social engineering and highly targeted phishing emails to selected individuals without high-level access and the corresponding protection. The installation of the deployed malware grants the attacker access to the victim's system. The network firewall will usually not block the communication with the C&C server because it originated from a trusted source and was on a port[2] used for virtually all encrypted Internet connections. If the victim system is connected to the network, the attacker will identify available ports and start running services on other systems. This allows the attacker to gain control over new systems without operating a direct connection to the C&C server (Neil et al., 2013). Pivoting therefore enables the attacker to explore protected intranets in order to gain control over highly protected operator systems, and search and exfiltrate intellectual property. Figure 1.1 depicts typical behaviour during a pivoting attack.

Communication between two devices itself is something common in computer networks. A hierarchical chain of controlled devices is not, and the detection of such is a clear indication of an ongoing network intrusion. Neil et al. (2013) propose a Markov model that quantifies the probability of multiple connected pairs of computers to detect general anomalous chains in a network. The detection of a true hierarchical control structure in which activity on one machine triggers activity on another one is more delicate and requires the analysis of the amount and type of traffic transmitted along a chain. A key aspect that is necessary to achieve this goal is the distinction of human induced network activity from machine driven one, and the classification of different types of the former. This work attempts such a classification on individual machine based on the network traffic leaving and entering them. A future cyber-security system might relate identified network activity on pairs of machines with the potential network connections between them in order to identity pivoting behaviour.

1.2. Contribution of this work

The aim of this project is to build a framework that is able to infer the activity state of computer users from the amount of network traffic that is generated during the usage period. For this, we will discuss a well-established model, the *Markov modulated Poisson process* (MMPP), which infers the state of a latent Markov process from the arrival rates of piece-wise Poisson distributed data. We will expose the disadvantages this model has regarding its application to network traffic, whose distribution deviates from a Poisson distribution. We then proceed to develop a hierarchical model that relaxes the requirement of Poisson distributed data and models the tail behaviour of network traffic in a more accurate way.

Our model will be embedded in a fast and scalable inference framework that can accurately identify the state of the latent Markov process. We will choose a fully Bayesian approach in order to account for the significant amount of prior information available in this issue.

A fundamental assumption in this work is that human activity governs the amount of generated network traffic, and that the state of the user can be inferred from it. We conducted a controlled experiment during which we gather flow data from phases with varying user activity in order to confirm our assumptions, and to test our framework. We will furthermore test our framework on network traffic originating from multiple selected personal computers from an enterprise-sized computer network. Finally, we will discuss

[2]see Section 2.1

the possibility of extending our developed model to include additional traffic quantities and propose a concrete and promising model extension.

On the structure of this Thesis

The remainder of our work is organized as follows:

1. In Chapter 2, we introduce the concept of a *flow* and the information it carries. We will then analyse the two data sets we are working with, point out their characteristics, and eliminate possible problems in them.

2. Chapter 3 introduces the concept of the MMPP. We will show how to infer the latent Markov process from observations, and introduce a Gibbs sampling framework by Fearnhead and Sherlock (2006) for parameter estimation. Finally, we point out the shortcomings of a conventional MMPP framework.

3. In chapter 4, we develop a new model that is based on the idea of an MMPP and demonstrate the advantages of this model over a conventional MMPP. We will then extend the Gibbs sampler of Fernhead and Sherlock to fit our new model, and apply it on the available data. Finally, we will discuss the accuracy and convergence of our results, and relate it to human computer activity.

4. Chapter 5 provides a brief summary of our results, points out its applications, and discusses possible extensions of our model for future work.

5. The appendix contains additional theory and plots as well as collection of results of our framework applied to 10 computers from the LANL data.

2. Data Analysis

2.1. Network flow data

A *flow* is a summary of a directed connection between two computers, in which a sequence of packets are sent from a source computer to a destination computer. Flows are most often recorded as event streams in the prominent *NetFlow* format by the network routers, with each *flow event* typically containing a time-stamp, the duration of the connection, the IP addresses of the source, and the destination computers, the source and destination ports, the protocol of the traffic, and the number of packets and bytes sent. An example of two flow events in the Netflow format can be seen in Figure 2.1.

Time stamp	Duration	Protocol	Src IP	Src Port	Dst IP	Dst Port	Packets	Bytes	
00:00:00.363	0.000	UDP	127.0.0.1	24920	192.168.0.1	22126	1	46	1
00:00:00.459	0.000	UDP	192.168.0.1	22126	127.0.0.1	24920	1	80	1

Figure 2.1.: Example of two flow events in the Netflow format.

Internet protocols

An *Internet communication protocol* is a system of rules that allow end-to-end data communication between two or more entities. It specifies how data should be packeted, addressed, and transmitted. The two most prominent Internet protocols are the *Transmission Control Protocol* (TCP), and the *User Datagram Protocol* (UDP). TCP provides reliable, ordered, and error-checked packet streams and is therefore especially suitable for World Wide Web, file transfers, or email. UDP provides provides connectionless datagram services without a guarantee of delivery. It emphasizes reduced latency over reliability, which is why it is preferred for real time streaming applications.

IP address

An *Internet Protocol address* (IP address) is a numerical label that serves as a location address of a device that uses an Internet Protocol for communication. The IP address is used to identify both the network host of the computer network the device is connected to, and the device location inside the network.

Ports

In computer networking, a *port* is an endpoint of communication in a network connection that identifies a specific type of network service transmitted in the connection. A port is identified by a port number, and is always associated with an IP address and the protocol type of the communication. Port, protocol, and IP together complete the destination/origination network address of a communication session. Although principally any port number can be assigned to any port, 1024 well-known port numbers are reserved by convention to identify specific service types. Prominent examples are *Secure Shell service* (22), and the *Hypertext transfer Protocol* (80). The source port number is usually of little relevance since the source computer does not have to be informed what kind of network services it is sending.

Network flow logs are originally intended as information for network dimensioning, and for *Quality-of-Service* and traffic control by the network provider. The inclusion of IP address, transmission protocol and the connection port for both ends of a connection in the flow event log allows a reconstruction of virtually all traffic inside a network. Due to the richness of network information available, network flow logs are also one of the main information sources in network intrusion detection, with both regular network users' as well as attackers' actions leaving trace evidence in it.

Models simulating the arrival times, lengths, and sizes of network flow streams have been extensively studied over the last twenty years. However, little attention was paid on the traffic originating from individual computers as the main application of such simulations is network planning and dimensioning of large-scale traffic. Many results regarding the flow arrival or duration distribution are scale-invariant and transferable to our data. We will discuss these results in more detail in Section 3.6.

2.2. Data used in this work

In this work, we look at a data set containing 16 consecutive days of network flow data from the Los Alamos National Laboratory's corporate, internal computer network (Kent, 2015). The network contains 17,684 devices and represents a typical enterprise network. Our goal is to model different user activity states on personal computers inside the network. However, the data does not contain labels to distinguish personal computers from other devices of less interest, and we do not have data about the activity conducted on the machines. Therefore, we have to apply logical and well-founded reasoning to validate our findings. To acquire some ground truth about the activity on a human controlled computer, we conducted a controlled experiment during which a user conducts different selected activities on a single computer inside the Imperial College network. The generated flow data was later collected at several key router locations.

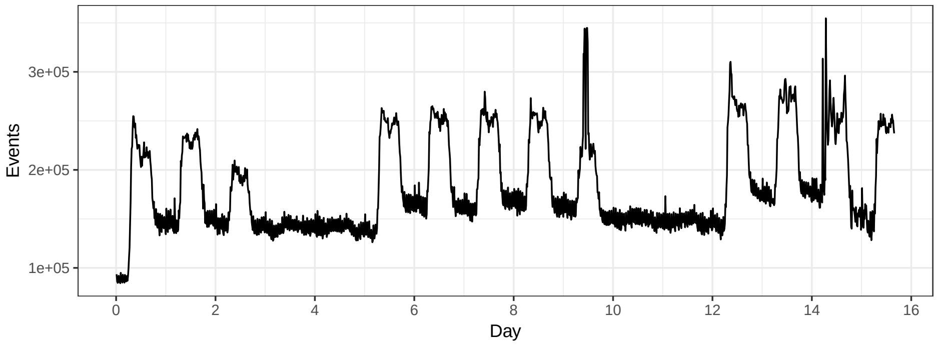

Figure 2.2.: Number of flow events in the LANL data, binned into 15 minute intervals. Clear differences day and night activity and weekday and weekend activity are visible.

We have mentioned above that flow data represents directed connections, i.e. a device can send or receive flow events. For individual personal computers, the two directions show a strong similarity in regards of the flow data distribution. We will therefore restrict ourselves to looking at flow events departing the particular computer of interest.

2.2.1. Los Alamos National Laboratory data

The main data set that we look at in this work contains 16 consecutive days of network data, collected from the *Los Alamos National Laboratory's* (LANL) corporate, *internal* computer network (Kent, 2015). The network consists of 17,684 computers which can be identified through their individual label through the data. Figure 2.2 depicts the number of flow arrivals in the network throughout time, binned into 15 minute intervals.

The LANL data contains only flows between internal devices, i.e. no connections to external hosts are included. Furthermore, the data not only contains flow events from personal computers, but from all devices connected to the network such as printers, IP phones, or network interfaces. Since we aim to model human behaviour in this work, we will restrict ourselves to devices that are subject to direct human control.

Based on observed patterns in the LANL data, in general we identify three main network flow characteristics of human controlled computers in an enterprise network:

1. A pronounced working day pattern, visible in Figure 2.4 A, and to some extent in Figure 2.2.

2. The distribution of flow events per minute in an hour has a high disorder, i.e. it is not possible to identify a specific minute in the hour that receives more flows than others. Examples of an ordered an a disordered hour distribution are shown in Figure 2.3.

3. The number of contacted destination computers and destination ports per day lies in a certain range ($\approx 10-20$ destinations and $\approx 120-140$ destination ports).

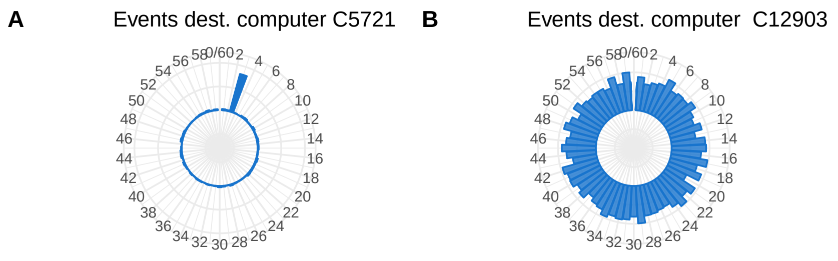

Figure 2.3.: Distribution of flow events per minute of the hour. The flow distribution of C5721 shows a high order, i.e. flows are only sent during a specific minute in an hour.

As mentioned above, there are no labels to distinguish personal computers from other devices in the LANL data, and we therefore cannot validate the correctness of these identified characteristics. Our assumptions are purely based on logical reasoning and the inspection of individual machines.

We can quantify the disorder of the hourly distribution and the order of the day/night distribution of flows from individual computers via the Shannon entropy:

$$S = -\sum_i p_i \log(p_i), \tag{2.1}$$

where $\{p_1, \ldots, p_{60}\}$ is the empirical hourly flow arrival distribution, and $\{p_{\text{day}}, p_{\text{night}}\}$ is the day/night flow distribution of a specific device. This quantification allows us to select a number of computers that exhibit human behaviour. Plots of the device distribution in the LANL data with respect to entropy and average daily contacted destinations/ports can be seen in Figure B.1. Flow plots with two supposedly human controlled devices and two other devices are depicted in Figure 2.4.

2.2.2. Imperial College data

The second data set we look at contains network flow data originating from a single source IP address inside the Imperial College network during a controlled experiment over a three-hour time span. The computer corresponding to this IP address[1] is a college computer running Microsoft Windows, and is accessible via user log-in. The purpose of the experiment was to see how different activities on a computer influence the flow arrival distribution. We therefore chose a diverse selection of activities that resemble all possible user states in an accurate way. For this, we included phases with video and social media consumption, or with surfing and information gathering, but also without any user activity or with the user logged off. Log-in and log-off processes are suspected to cause spikes in the activity, so we logged the user in and out frequently. The activity schedule looks as following:

[1]during the three-hour time interval

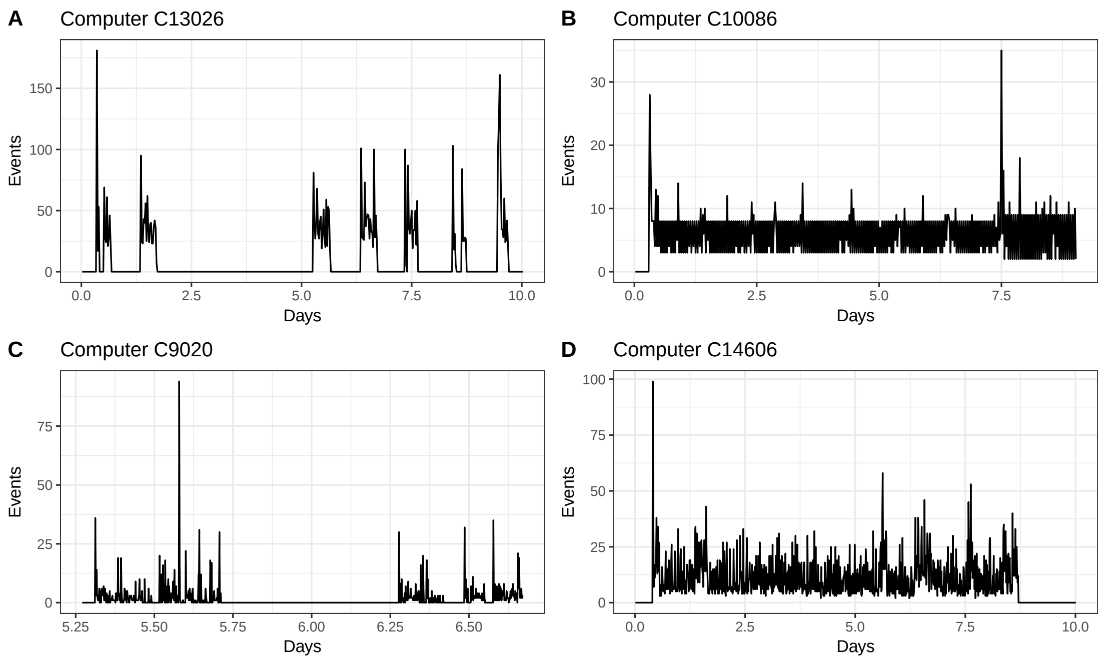

Figure 2.4.: Flow events from four different source computers. Only computer C13026 and C9020 exhibit regular human working day behaviour.

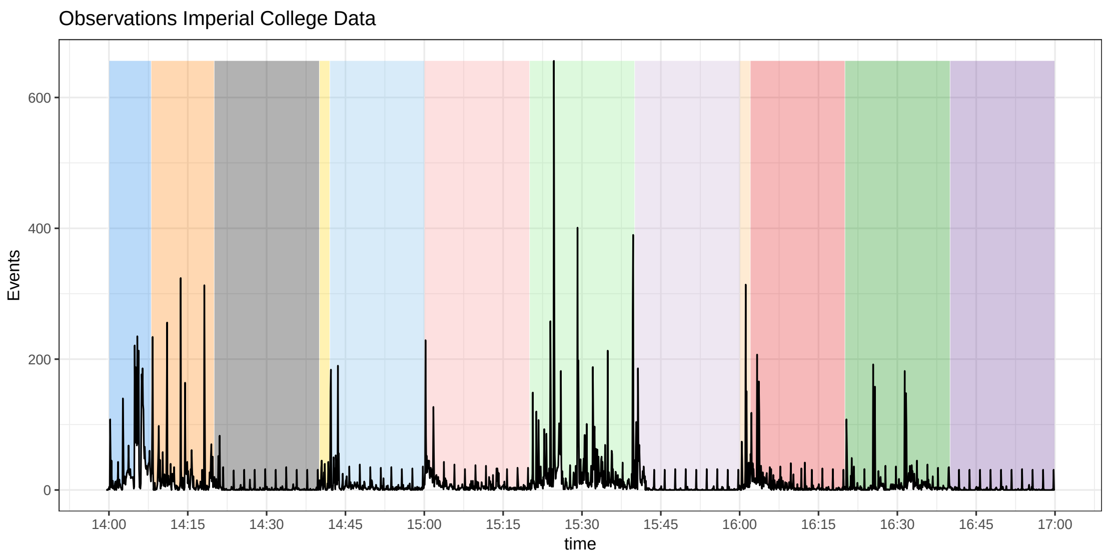

Figure 2.5.: Number of Imperial College flow events, binned into 5s intervals. The coloured intervals correspond to the different stages during the experiment.

14:00	User log-in Open Mozilla Firefox Scrolling on `www.facebook.com`, opening occasional videos on this site
14:08	Open Google Chrome, close Mozilla Firefox Scrolling on `www.9gag.com`
14:20	User log-out User log-in No further activity (no browser or other program open)
14:40	User log-out User log-in
14:42	Open Mozilla Firefox, open several sites[2] No further activity
15:00	Visiting `www.soundcloud.com`, Downloading three files with a total size of 2 GB Watching several videos on `www.youtube.com`, all of which are several minutes long.
15:20	Intense surfing, clicking from site to site Listening to music on `soundcloud.com`
15:40	User log-off
16:00	User log-in
16:02	Open Mozilla Firefox Playing multiplayer online game on `www.splix.io`
16:20	Establishing SSH-connection to `bazooka.ma.ic.ac.uk` Sending Unix-commands via ssh Occasional look-up of information using Mozilla Firefox
16:40	End of experiment

Figure 2.5 depicts the number of flow events binned into 5 second intervals during the experiment. The colours represent the activity schedule. A discussion of their relation will be done in Section 4.4.

Spikes

For both data sets, we observe several large spikes, standing out from the rest of the data. In the Imperial College data, these spikes primarily stem from new web processes being started on the computer which trigger many DNS requests to a single IP-address. For instance, the largest spike in the Imperial Data at 15:24:34 contains 656 flow events,

[2] `www.facebook.com`, `www.gmail.com`, `www.9gag.com`, `www.faz.de`, `www.outlook.com`

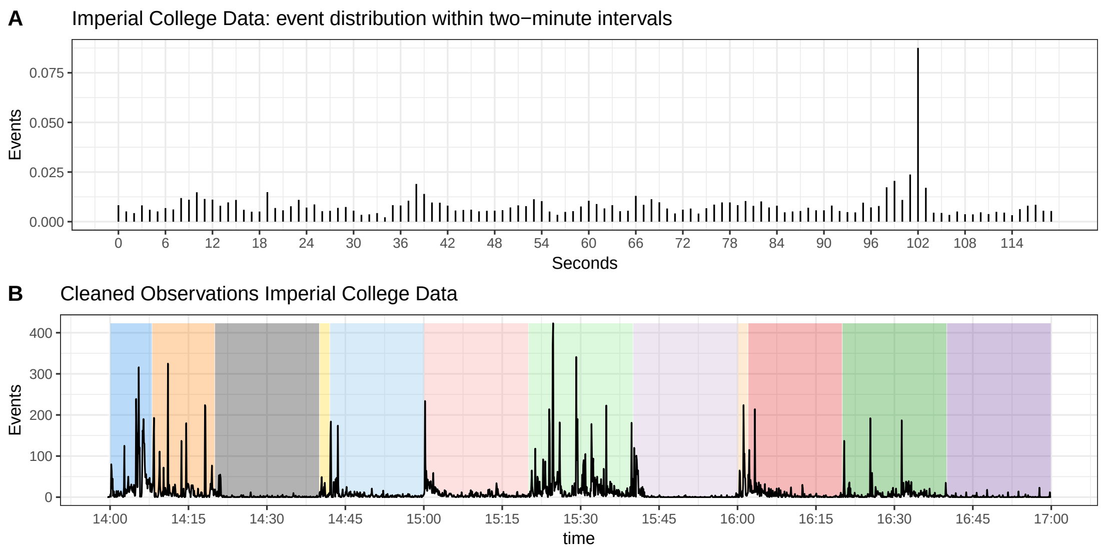

Figure 2.6.: Number of events per second distribution over two-minute intervals, and the cleaned Imperial College flow data, binned into 5s intervals.

of which 297 are directed to the IP-address `155.198.142.8`, all via UDP-port 53 (which is responsible for DNS requests). The second most contacted IP address received only 32 flows. From the Imperial College data, we can observe that these only occur during human interaction with the computer. The identification of these DNS-spikes as a new device state in our framework might be of interest for a broader intrusion detection framework.

Since the LANL data contains no external traffic, we do not observe DNS spikes. The spikes we observe correspond to *Kerberos-authentications*[3] via UDP ports 88 and 389, which explains why they are most pronounced at the start of daily computer activity.

2.3. Data cleaning

A significant amount of a computer's flow traffic is caused by strictly periodic communication. This sort of communication might manifest itself in periodic update requests or flows restarting due to exceeding specific time limits. For example, a pattern of event spikes in two-minute intervals is visible in Figure 2.5.

This periodic communication is highly deterministic and will alter the observed flow arrival distribution without giving us any information about the state of the user. We will attempt to remove this communication from our data.

Imperial College data

[3]An authentication service for open computer networks

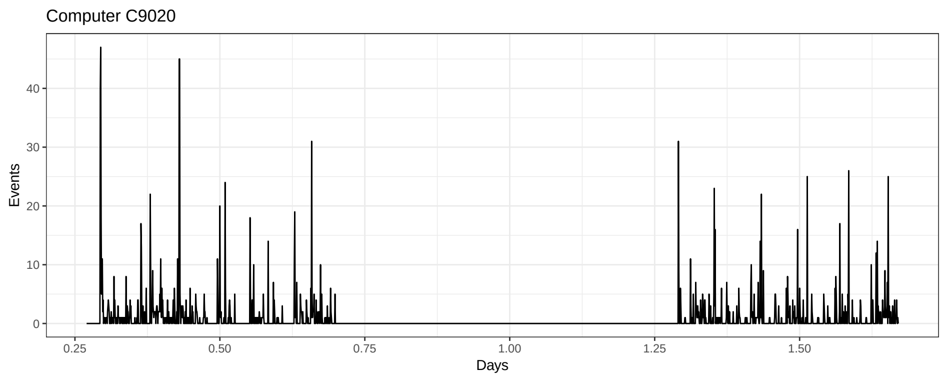

Figure 2.7.: Network flow excerpt of a LANL computer after cleaning, binned into 1 minute intervals.

In Figure 2.5 we observe spikes reoccurring every two minutes. Figure 2.6 A displays the empirical[4] distribution of flow events per second within a two-minute interval. Apparently, events in second 103 behave significantly different from those happening in during any other second. We therefore remove all events within second 103 in all two-minute intervals. The cleaned observations can then be seen in Figure 2.6.

LANL data

Since the LANL data only contains internal traffic and we are therefore looking at smaller flow arrival rates, we do not observe pronounced periodic spikes in the set of selected LANL computers. We can rather find a set of destination hosts and ports used in the network that only communicate in very periodic manners, and exclude their connections from the flow data sets we are investigating. As described in Equation 2.1, we can calculate the Shannon entropy of the minutely and hourly distribution for every destination port and computer in the network to identify ports with periodic communication. Figure 2.7 depicts an excerpt of computer C9020 after the cleaning.

Chapter summary

This chapter introduced the notion of a network flow, explained the different parts in a flow event log, and emphasized their importance for cyber-security. We discussed the characteristics of computers in the LANL network data and showed how to filter it for computers that are of interest to us. We then explained the difficulty of validating our validating our model results without the knowledge about the activity taking place on these computers, and conducted a controlled experiment to gather data that relates network traffic with human activity states. Finally, we discussed problematic points in our data and how to remove them.

[4]averaged over all two-minutes intervals of the data set

3. Bayesian framework for the Markov-modulated Poisson process

A Markov-modulated Poisson process (MMPP, also called Markovian arrival process) is a Poisson process whose intensity depends on the current state of an independently evolving continuous time Markov chain with finite state-space. An example of an MMPP is a single-molecule fluorescence experiment where the molecule alternates between two possible states according to a Markov chain, and the arrival rate of emitted photons at a receptor depends on the current state of the molecule (Burzykowski et al., 2003).

Usually, the underlying Markov process is unobserved, and MMPP frameworks are employed to infer the state of the Markov process from the observed arrival data. In a typical context, both the state of the Markov process as well as the specific parameters of the MMPP are unknown and have to be estimated while the number of states is assumed to be known. Multiple sources showed that the estimation of the parameters can be done via *Expectation-Maximization* (EM), and the chain states can be sampled afterwards (Breuer, 2002; Rydén, 1996). Here we focus on a completely Bayesian framework that follows Fearnhead and Sherlock (2006) to sample both the parameters as well as the process states directly using Gibbs sampling. Our framework will use a batch MMPP which takes binned data as input rather than raw arrival timestamps.

3.1. Likelihood for the MMPP

We begin by defining an M-state MMPP that evolves during a defined time span.

We define a time window of observation $[0, t_{\text{obs}}]$. Let $\{X_t\}$ for $t \in [0, t_{obs}]$ be a (hidden) discrete Markov process evolving inside this time window on a state space of cardinality M. Let $\mathbf{Q}$ be the infinitesimal generator of $\{X_t\}$, with $Q_{ii} = -\sum_{j \neq i} Q_{ij}$, $Q_{ii} \leq 0$. The stationary distribution of $\{X_t\}$ is given by $\nu = \ker(\mathbf{Q})$. The transition probability of the Markov process transitioning from state j to state i after time t is given by

$$P_{ij}(t) = P(X_t = i | X_0 = j) = \exp(\mathbf{Q}t)_{ij} \qquad i, j \in \{1, \ldots, M\}. \tag{3.1}$$

Let $\boldsymbol{\lambda} \triangleq \{\lambda_1, ..., \lambda_M\}$ be the Poisson process rates. Let $N(t)$ be a Poisson process with rate $\lambda(t) = \lambda_{i=X_t}$, i.e. while X stays in state i, events occur with rate λ_i.

Let $\{t'_1, ..., t'_l\}$ be the arrival times of $N(t)$ with $l \triangleq N(t_{\text{obs}})$, and define $t'_0 = 0$, $t'_{l+1} = t_{\text{obs}}$. Define the interarrival times $\Delta t'_k = t'_k - t'_{k-1}$, $k \in \{1, ..., l+1\}$.

At this point, we are interested in the evolution of the Markov process at the arrival times. For this, we need the following probability:

$$P_{ij}^{(0)}(t^*) = P\Big(\text{there are no arrivals in } (0,t^*) \text{ and } X_{t^*} = i \; | X_0 = j\Big). \tag{3.2}$$

We can easily derive this probability by defining a meta-Markov process $\{W_t\}$ on the space $\{1, ...M, M+1\}$ on $(0,t^*)$. Let $\{t_1', t_2', \dots\}$ be the arrival times during $(0,t^*)$, possibly being empty. t_1' is the time the first event occurs. $\{W_t\}$ is defined as follows:

$$W_t = \begin{cases} X_t, & 0 \le t < t_1' \\ \\ M+1, & t_1' \le t \end{cases} \qquad t \in [0,t^*], \tag{3.3}$$

i.e. W_t mirrors X_t until the first event happens, then it jumps into the added state and stays there. Let $\boldsymbol{\lambda} = \text{diag}(\boldsymbol{\lambda})$. The infinitesimal generator of this process is given by

$$\mathbf{G_w} = \begin{pmatrix} \mathbf{Q} - \boldsymbol{\Lambda} & \boldsymbol{\lambda} \\ \mathbf{0} & 0 \end{pmatrix}. \tag{3.4}$$

Note that the transition rates to leave the state $M+1$ are equal to zero, and the transition rates of jump to state $M+1$ are given by $\boldsymbol{\lambda}$. The full transition matrix of $\{W_t\}$

$$\begin{aligned} P_{ij}^{W}(t^*) &= P(W_t = i | W_0 = j) \\ &= \exp(\mathbf{G_w} t) \\ &= \begin{pmatrix} \exp\{(\mathbf{Q} - \boldsymbol{\Lambda})t\} & (\mathbf{Q} - \boldsymbol{\Lambda})^{-1}[\exp\{(\mathbf{Q} - \boldsymbol{\Lambda})t\} - \mathbf{I}]\boldsymbol{\lambda} \\ \mathbf{0} & 1 \end{pmatrix}. \end{aligned} \tag{3.5}$$

From this we see that our desired probability is given by

$$P_{ij}^{(0)}(t^*) = \exp\{(\mathbf{Q} - \boldsymbol{\Lambda})t^*\}_{ij}. \tag{3.6}$$

We can now compute likelihood of the interarrival time $\Delta t_k'$ conditional on $X_{t_k'}$, $X_{t_{k-1}'}$:

$$\begin{aligned} P(\Delta t_k' | X_{t_k'} = i, X_{t_{k-1}'} = j) &= \frac{P(\Delta t_k', X_{t_k'} = i | X_{t_{k-1}'} = j)}{P(X_{t_k'} = i | X_{t_{k-1}'} = j)} \\ &= \frac{\exp\{(\mathbf{Q} - \boldsymbol{\Lambda})\Delta t_k'\}_{ij}\lambda_j}{\exp\{(\mathbf{Q} - \boldsymbol{\Lambda})\Delta t_k'\}_{ij}} = \lambda_j \end{aligned} \tag{3.7}$$

The total likelihood of the arrival times $\{t_1', \dots, t_{l+1}'\}$ (this now includes the constraint of $t_k' = t_{\text{obs}}$) is therefore given by

$$\begin{aligned}P\big(\{t'_k\}\big) &= \sum_{\{i_{t'_k}\}} P(X_{t'_0} = i_{t'_0})P(t_1, X_{t'_1} = i_{t'_1}|X_{t'_0} = i_{t'_0})\dots \\ &\qquad \dots P(t'_{l+1}, X_{t'_{l+1}} = i_{t'_{l+1}}|X_{t'_l} = i_{t'_l}) \\ &= \nu^T \exp\{(\mathbf{Q} - \mathbf{\Lambda})\Delta t'_1\}\boldsymbol{\lambda}\dots \\ &\qquad \dots \exp\{(\mathbf{Q} - \mathbf{\Lambda})t_n\}\boldsymbol{\lambda}\exp\{(\mathbf{Q} - \mathbf{\Lambda})\Delta t'_{l+1}\}\mathbf{1}.\end{aligned} \tag{3.8}$$

3.2. Likelihood for the batch MMPP

When dealing with network flow events, two arguments speak against the use of raw arrival times:

1. The resolution with which flow arrival times are recorded is often in second intervals. Flows recorded within the same second therefore do not have a timely separation, which leads to serious deviation from a Poisson process
2. Within an enterprise network, typically between 10^7 and 10^{10} flow events are observed per day. The scalability of any operations acting on the stream of network flows is therefore of particular interest. Consequently, we are interested in an approach that bins multiple flow events together in order to reduce the necessary number of operations.

These two arguments make it particularly attractive to look at network flow arrivals in the format of *accumulation intervals* instead of the raw arrival times. These are a set of intervals $\{I_1 = [t_0 = 0, t_1), ..., I_n = [t_{n-1}, t_n = t_{\text{obs}})\}$ of equal length $t^* \triangleq t_i - t_{i-1}$. Associated with each interval is a count $z_i = N(t_i) - N(t_{i-1})$ of the number of arrival events during interval I_i.

Since we only observe the accumulated arrival events at the interval times, we are now interested in the following probability:

$$P^{(z_i)}_{jk} = P\Big(\text{there are } z_i \text{ arrival events in } (0, t^*) \text{ and } X_{t^*} = k \;|X_0 = j\Big). \tag{3.9}$$

Similar as before, we can derive this probability by defining a meta-Markov process V_t. Let $z_{\max} = \max(z_i)$. Define the new $(M \cdot z_{\max} + 1)$-dimensional state space $S = (1^{(0)}, ..., M^{(0)}, 1^{(1)}, ..., M^{(1)}, ..., 1^{(z_{\max})}, ..., M^{(z_{\max})}, 1^*)$. Let $\{t'_1, ..., t'_{z_{\max}}, t'_{z_{\max}+1}, ...\}$ be a (again possibly empty) set of arrival times inside $[0, t_{\text{obs}}]$.

$\{V_t\}$ is defined as follows:

$$V_t = \begin{cases} X_t^{(0)}, & 0 \leq t \leq t'_1 \\ \vdots & \\ X_t^{(i)}, & 0 \leq t'_i < t'_{i+1} \\ \vdots & \\ 1^*, & t'_{z_{\max}+1} \leq t \end{cases} \qquad t \in (0, t^*). \tag{3.10}$$

The state of V_t reflects both the state of X_t and the number of occurred events until this number exceeds $z_{\max}$ (which does not occur in the observations).

The infinitesimal generator for $\{V_t\}$ is given by

$$\mathbf{G_V} = \begin{pmatrix} \mathbf{Q}-\mathbf{\Lambda} & \mathbf{\Lambda} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}-\mathbf{\Lambda} & \mathbf{\Lambda} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \dots & \dots & \mathbf{0} & \mathbf{Q}-\mathbf{\Lambda} & \boldsymbol{\lambda} \\ \mathbf{0} & \dots & \dots & \mathbf{0} & \mathbf{0} & 0 \end{pmatrix} \tag{3.11}$$

The transition matrix $P_{ij}^{(z_i)}$ is now given by

$$P_{jk}^{(z_i)}(t^*) = P\Big(V_{t^*} = k^{(z_i)} \, | V_0 = j^{(0)}\Big) = \big[\exp(\mathbf{G_V} t^*)\big]_{j,k+z_i \cdot M} \tag{3.12}$$

Since the number of observed events is already encoded in the end-state of V_t, the likelihood of one accumulation interval is given by

$$P_{\mathbf{Q},\boldsymbol{\lambda}}(z_i | V_{t^*} = k^{(z_i)}, V_0 = j^{(0)}) = 1. \tag{3.13}$$

The likelihood of the observed data $\{z_1, ..., z_n\}$ is therefore given by

$$P_{\mathbf{Q},\boldsymbol{\lambda}}\Big(\{z_i\}\Big) = \nu^T \Big(\prod_{i=1}^{n} P^{(z_i)}\Big)\mathbf{1}, \tag{3.14}$$

where

$$P^{(z_i)} = \begin{pmatrix} \big[\exp(\mathbf{G_V} t^*)\big]_{1,1+z_i \cdot M} & \dots & \big[\exp(\mathbf{G_V} t^*)\big]_{1,M+z_i \cdot M} \\ \vdots & \vdots & \vdots \\ \big[\exp(\mathbf{G_V} t^*)\big]_{M,1+z_i \cdot M} & \dots & \big[\exp(\mathbf{G_V} t^*)\big]_{M,M+z_i \cdot M} \end{pmatrix}. \tag{3.15}$$

3.3. Forward-Backward Algorithm

The recursive form of the likelihood in Equation 3.8 and 3.14 shows that we can formulate the MMPP as a discrete *hidden Markov model* (HMM). Inferring the hidden Markov chain in a HMM is not trivial and subject to extensive research. For discrete HMMs, it is fortunately always possible to sample the Markov chain directly using the so called *Forward-Backward algorithm* (Baum, 1972; Devijver, 1985).

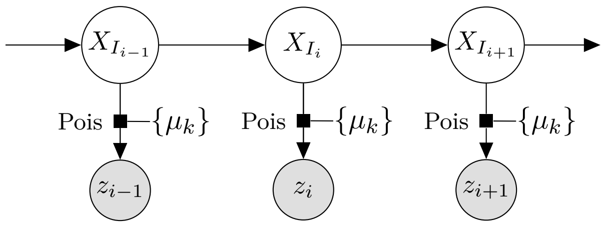

Figure 3.1.: Graphical model for the batch MMPP.

The backward-part of the Forward-Backward algorithm calculates the posterior distributions of the Markov process conditional on the previous element while the forward samples each element sequentially. We will explain the important steps in the context of the batch Markov modulated Poisson process. For a more general description see (Baum, 1972; Fearnhead and Sherlock, 2006).

Again, let $\{t_0, ..., t_n\}$ be the times defining the accumulation intervals, and let $t^* \triangleq t_i - t_{i-1}$ be the interval length and $z_i = N(t_i) - N(t_{i-1})$ be the event count of each interval.

Define

$$\mathbf{A}^{(k)} = P(\{z_k, ..., z_n\}, X_{t_n} | X_{t_{k-1}}), \qquad k \in \{1, ..., n\}. \tag{3.16}$$

For $k = n$, we have

$$\mathbf{A}^{(n)}_{ij} = P(z_n, X_{t_n} = j | X_{t_{n-1}} = i) = \big[\exp(\mathbf{G_V} t^*)\big]_{i, j+z_n \cdot M}. \tag{3.17}$$

We can then calculate $A^{(k)}$ via backward recursion:

$$\mathbf{A}^{(k)} = \big[\exp(\mathbf{G_V} t^*)\big]_{1:M, 1:M+z_k \cdot M} \mathbf{A}^{(k+1)}. \tag{3.18}$$

For large number of observations intervals, the direct calculation of $A^{(k)}$ would not be numerically stable. Since we are only interested in the relative probabilities of $\mathbf{A}^{(k)}$ in order to sample, we therefore normalize each $A^{(k)}$ with the *max norm*[1].

Forward-sampling can then be done as follows: We can sample the initial state of the Markov process using the stationary distribution of the process:

$$P\Big(X_{t_0} = s | \{z_1, ..., z_n\}\Big) = \frac{\mu_s [\mathbf{A}^{(1)} \mathbf{1}]_s}{\boldsymbol{\mu}^T \mathbf{A}^{(1)} \mathbf{1}}. \tag{3.19}$$

We can then proceed to sample $X_{t_1}, ..., X_{t_{n-1}}$:

$$P\Big(X_{t_k} = s | \{z_1, ..., z_n\}, X_{t_{k-1}} = s_{k-1}\Big) = \frac{\big[\exp(\mathbf{G_V} t^*)\big]_{s_{k-1}, s+z_k \cdot M} \big[\mathbf{A}^{(k+1)} \mathbf{1}\big]_s}{\big[\mathbf{A}^{(k)} \mathbf{1}\big]_{s_{k-1}}}. \tag{3.20}$$

[1]The maximal element of a matrix. Any other reasonable matrix norm would work too.

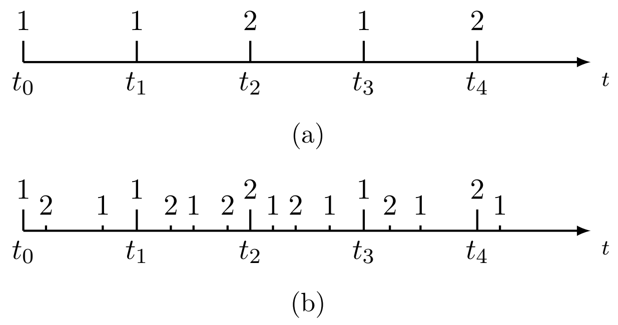

Figure 3.2.: First, simulate the chain state at the interval start and end times using the Forward-Backward algorithm (a); for each interval then simulate the evolution of $\{X_t\}$ inside the interval conditional on the endpoints (b).

Finally, we can sample $X_{t_n = t_{\text{obs}}}$:

$$P\Big(X_{t_n} = s | \{z_1, ..., z_n\}, X_{t_{n-1}} = s_n\Big) = \big[\exp(\mathbf{G_V} t^*)\big]_{i, j+z_n \cdot M}. \tag{3.21}$$

3.4. Simulate full underlying Markov process

With the Forward-Backward algorithm, we are able to sample the state of the Markov process $\{X_t\}$ at the times $\{t_0, ..., t_n\}$. We are now interested in sampling the whole Markov process, i.e. we want to sample all times at which a state transition happens. To do this, we can look at the time interval $I_k = [t_k, t_{k+1}]$ (which is of length t^*) and sample the state transitions conditional on $X_{t_k}, X_{t_{k+1}}$. We have to condition on the fact that during the interval, z_k events take place:

$$\begin{aligned} &P\Big(\{X_t\}, \text{there are } z_k \text{ Y-events in } (t_{k-1}, t_k) | X_{t_{k-1}} = i,\ X_{t_k} = j\Big) \\ &= P\Big(\{V_t, t \in (t_{k-1}, t_k)\} | V_{t_{k-1}} = i^{(0)},\ V_{t_k} = j^{(z_k)}\Big). \end{aligned} \tag{3.22}$$

where we used the definition of $\{V_t\}$ and its generator from Equation 3.10 and 3.11. We are therefore simply looking at the evolution of the Markov-process $\{V_t\}$ with specified end-points.

Drawing sample paths for endpoint-conditioned continuous time Markov processes is not trivial. Two common techniques to solve this problem are *Modified Rejection sampling* and *Uniformization sampling*. A detailed description of both algorithms and their particular advantages and disadvantages is given in Section A.1, and (Hobolth and Stone, 2009). We will use both techniques where appropriate during the sampling of $\{V_t\}$, depending on the particular endpoints $V_{t_k}, V_{t_{k+1}}$, to take full advantage of both methods. For more details, again see Section A.1.

The sampled trajectory of $\{V_t\}$ contains both a sampled trajectory of $\{X_t\}$ as well as a sample of the exact arrival times $\{t'_1, ..., t'_l\}$ of all flows[2]. Figure 3.2 resembles the procedure of sampling the full trajectory of $\{X_t\}$.

3.5. Gibbs Sampler

We have demonstrated how to sample both $\{X_t\}$ and $\{t'_1, ..., t'_l\}$ conditional on the observed counts $\{z_1, \dots, z_n\}$ and the parameters $\mathbf{Q}, \boldsymbol{\lambda}$. In other words, we are able to sample from

$$P\Big(\{X_t\}, \{t'_1, ..., t'_l\} | \{z_1, \dots, z_n\}, \mathbf{Q}, \boldsymbol{\lambda}\Big). \tag{3.23}$$

We will now follow an approach proposed by Fearnhead and Sherlock (2006) to implement an exact Gibbs sampler for

$$P\Big(\{X_t\}, \{t'_1, ..., t'_l\}, \mathbf{Q}, \boldsymbol{\lambda} | \{z_1, \dots, z_n\}\Big). \tag{3.24}$$

The likelihood of $\{X_t\}$ and $\{t'_1, .., t'_l\}$ is given by

$$L(\{X_t\}, \{t'_1, .., t'_l\} | \mathbf{Q}, \boldsymbol{\lambda}) \propto \nu_{X_{t'_0}} \prod_{i=1}^{M} \Big(\lambda_i^{n_i} \exp(-\lambda_i \tilde{t}_i) \prod_{j \neq i} q_{ij}^{r_{ij}} \exp(-q_{ij} \tilde{t}_i)\Big) \tag{3.25}$$

where

$$\tilde{t}_i \triangleq \int_0^{t_{\text{obs}}} \mathbb{1}_{X_t = i} \, \mathrm{d}t \tag{3.26}$$

is the time spent in state i,

$$n_i \triangleq \sum_{k=1}^{l} \mathbb{1}_{X_{t'_k} = i} \tag{3.27}$$

is the number of arrival events taking place while $X_t = i$, and

$$r_{ij} \triangleq |\{t_k, \quad X_{t_k} = i \text{ and } X_{t_k +} = j\}| \tag{3.28}$$

is the number of transitions from state i into state j.

If we employ a Bayesian framework, it is possible to calculate the posterior distributions of $\mathbf{Q}, \boldsymbol{\lambda}$ conditional on $\{X_t\}$ and $\{t'_1, ..., t'_l\}$:

We impose a Gamma-prior on λ_i with hyper-parameters $\alpha_{\lambda,i}$ and $\beta_{\lambda,i}$:

[2]Given the fact that we record flow events with a timestamp, the sampling of the flow arrival times might seem unnecessary and one might question the benefit of using accumulation intervals in our framework. However, despite the fact that the precision of those timestamps is not high enough for a working MMPP framework, we benefit a lot computationally by applying the Forward-Backward algorithm only on the accumulation intervals rather than all interarrival times. Furthermore, the use of a coherent interval length accelerates the sampling of $\{X_t\}$ immensely as described in Section A.1 (please keep in mind that else we would have to sample $\{X_t\}$ during each interarrival time).

$$\lambda_i \sim \Gamma(\alpha_{\lambda,i}, \beta_{\lambda,i}) \tag{3.29}$$

Likewise,

$$q_{ii} \sim \Gamma(\alpha_{q,i}, \beta_{q,i}) \tag{3.30}$$

For $q_{i \neq j}$, we impose a Dirichlet prior with hyper-parameter $\boldsymbol{\alpha}_{D,i}$:

$$\frac{(q_{1,i}, ..., q_{i-1,i}, q_{i+1,i}, ..q_{M,i})^T}{q_{ii}} \sim \text{Dir}(\boldsymbol{\alpha}_{D,i}) \tag{3.31}$$

The posterior distributions are then given by

$$\lambda_i | \{X_t\}, \{t'_1, ..., t'_l\} \sim \Gamma(\alpha_{\lambda,i} + n_i, \beta_{\lambda,i} + \tilde{t}_i) \tag{3.32}$$

Algorithm 1: Gibbs sampler for batch MMPP

```
Data: ${z_1, \ldots, z_n}$
Initialize:
    1. Sample:
        $\lambda_i^{(0)} \sim \Gamma(\alpha_{\lambda,i}, \beta_{\lambda,i})$
        $q_{ii}^{(0)} \sim \Gamma(\alpha_{q,i}, \beta_{q,i})$
        $(q_{1,i}, ..., q_{i-1,i}, q_{i+1,i}, ..q_{M,i})^{T(0)} \sim \text{Dir}(\boldsymbol{\alpha}_{D,i}) \cdot q_{ii}$

for $a \in \{1, \ldots,$ number of desired samples$\}$ do
    1. Sample $\{X_{t_1}, \ldots, X_{t_n}\}^{(a)}$ recursively using the Forward-Backward algorithm
       conditional on $\{z_1, \ldots, z_n\}^{(a)}$, $\boldsymbol{\lambda}^{(a-1)}$, and $\mathbf{Q}^{(a-1)}$ as described in Equation 3.20
    2. Sample $\{X_t\}^{(a)}$ and $\{t'_1, \ldots, t'_l\}^{(a)}$ conditional on $\{X_{t_1}, \ldots, X_{t_n}\}^{(a)}$ using
       Equation 3.22
    3. Calculate $\tilde{t}_i^{(a)}$, $n_i^{(a)}$, and $r_{ij}^{(a)}$ with equations 3.26-3.28 from $\{X_t\}^{(a)}$ and
       $\{t'_1, \ldots, t'_l\}^{(a)}$
    4. Sample:
        $\lambda_i^{(a)} \sim\sim \Gamma(\alpha_{\lambda,i} + n_i,^{(a)} \beta_{\lambda,i} + \tilde{t}_i^{(k)})$
        $q_{ii}^{(a)} \sim \Gamma(\alpha_{q,i} + \sum_{j \neq i} r_{ij}^{(a)}, \beta_{q,i} + \tilde{t}_i^{(a)})$
        $(q_{1,i}, ..., q_{i-1,i}, q_{i+1,i}, ..q_{M,i})^{T(a)} \sim \text{Dir}(\boldsymbol{\alpha}_{D,i} + \mathbf{r}_i)^{(a)} \cdot q_{ii}$
```

$$q_{ii}|\{X_t\},\{t'_1,...,t'_l\} \sim \Gamma(\alpha_{q,i} + \sum_{j\neq i} r_{ij}, \beta_{q,i} + \tilde{t}_i) \tag{3.33}$$

$$\frac{(q_{1,i},...,q_{i-1,i},q_{i+1,i},..q_{M,i})}{q_{ii}}|\{X_t\},\{t'_1,...,t'_l\} \sim \mathrm{Dir}(\boldsymbol{\alpha}_{D,i} + \mathbf{r}_i) \tag{3.34}$$

where $\mathbf{r}_i = (r_{1,i},...,r_{i-1,i},r_{i+1,i},..r_{M,i})^T$.
We can now sample from

$$P\Big(\{X_t\},\{t'_1,...,t'_l\}|\{z_1,\ldots,z_n\},\mathbf{Q},\boldsymbol{\lambda}\Big) \tag{3.35}$$

as well as from

$$P\Big(\mathbf{Q},\boldsymbol{\lambda}|\{X_t\},\{t'_1,...,t'_l\},\{z_1,\ldots,z_n\}\Big). \tag{3.36}$$

This now allows the implementation of a direct Gibbs sampler in order to explore

$$P\Big(\mathbf{Q},\boldsymbol{\lambda},\{X_t\},\{t'_1,...,t'_l\}|\{z_1,\ldots,z_n\}\Big). \tag{3.37}$$

An implementation of this Gibbs sampler is described in Algorithm 1.

3.6. A naive fitting attempt and analysis

The above described Gibbs sampler principally allows us a direct attempt to identify different machine states: We can define appropriate hyperparameters for $\mathbf{Q}$ and $\boldsymbol{\lambda}$, choose an appropriate number of states, and employ Algorithm 1 on the binned data $\{z_1,\ldots,z_n\}$ to sample $\Big\{\{X_t\},\mathbf{Q},\boldsymbol{\lambda}\Big\}|\{z_1,\ldots,z_n\}$. As this is only a demonstration of the flaws a simple MMPP model has, we will discuss the choice of prior hyperparameters and number of states in Section 4.3.

We run Algorithm 1 to generate 500 samples of $\Big\{\{X_t\},\mathbf{Q},\lambda\Big\}|\{z_1,\ldots,z_n\}$ for both the Imperial College data as well for a selected machine from the LANL data. Figure 3.3 shows excerpts of the sampled $\{X_t\}$-distributions, with a detailed plot description in the figure caption. Figure 3.4 depicts the corresponding samples of the process rates Q_{ii}.

As visible in the plots, the sampler is able to identify a rough structure in the data. Periods with no or little activity can be distinguished from very active periods, and the isolated peaks described in Section 2.2 are accurately identified as state 4 resp. state 5 periods.

However, the plots indicate that the predictions of the underlying Markov process are very unstable: The overall uncertainty of the sampled states is immense during the majority of the observed time periods, making the identification of one distinct state at each point in time impossible. Furthermore, with the exception of Q_1 and Q_2 for the LANL data, the sampled rates for Q_i are unreasonably high for both data sets, corresponding to an average lifetime of each state in the range of seconds. An accurate

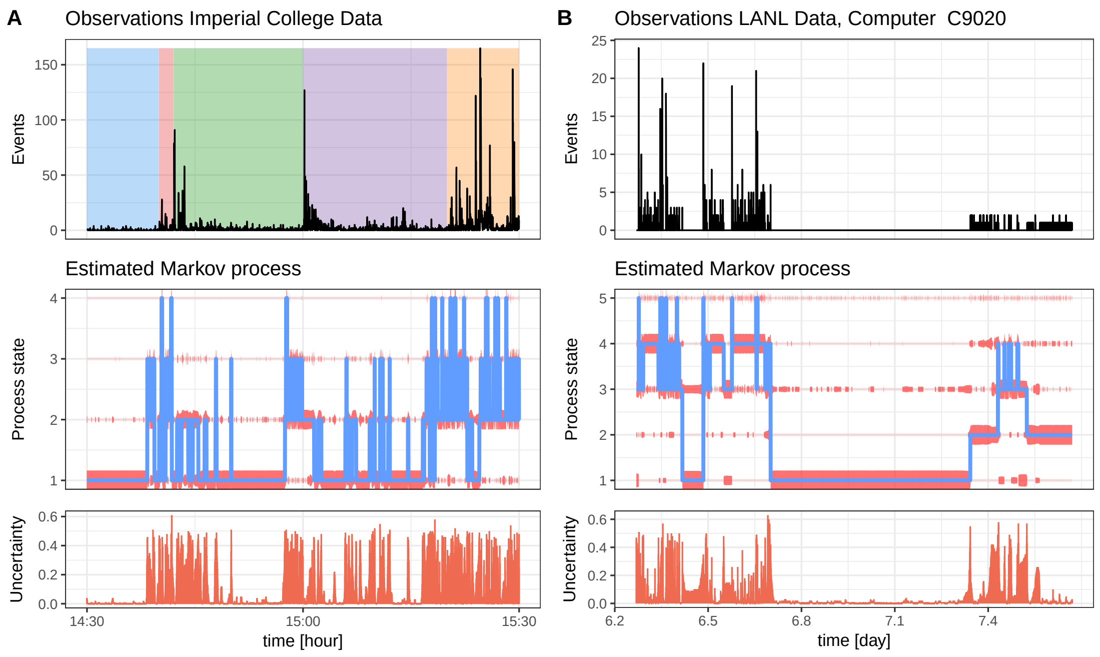

Figure 3.3.: Binned flow data (top), 500 samples of $\{X_t\}$ (using Algorithm 1) (middle), and uncertainty of the inferred state. Plot A depicts an excerpt of the Imperial College Data, Plot B and excerpt of a machine from the LANL data. The blue line indicates the sample mode at each point in time while the thickness of the red lines indicates the amount of samples in each state. The uncertainty is calculated by $1 - \alpha_{t_i}$ where α_{t_i} is the fraction of the sample mode of all samples at time t_i.

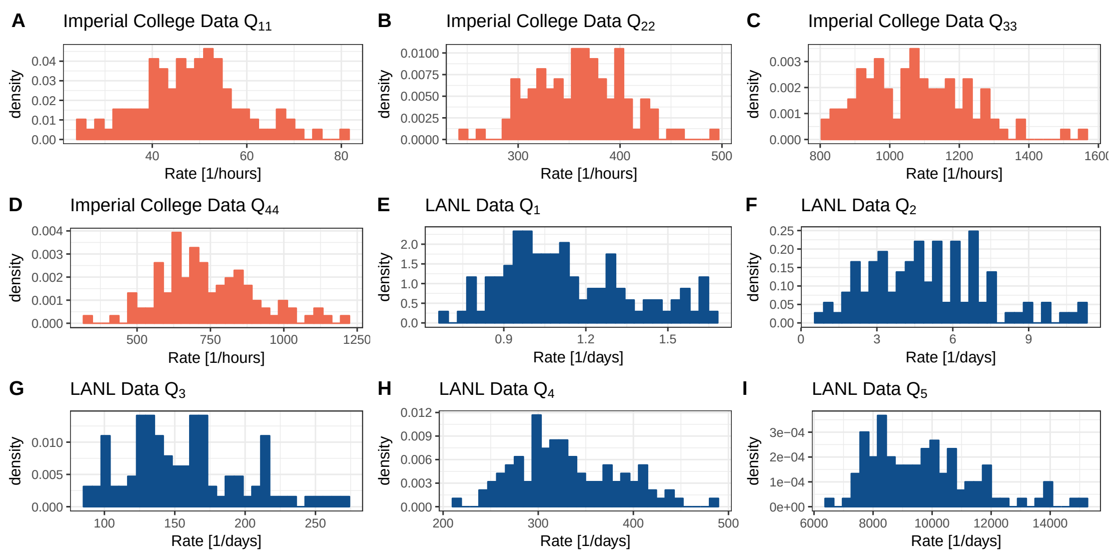

Figure 3.4.: Sampled Markov process decay rates for both the Imperial College and the LANL data.

model of human computer usage cannot agree with such predictions. Therefore, these results indicate that a simple MMPP model is not capturing the nature of network traffic properly.

Network arrivals are often modeled as Poisson processes for analytic simplicity, even though a number of traffic studies have shown that packet or flow interarrivals are not exponentially distributed. The fact that network traffic shows heavy tail behaviour, Long Range Dependence[3] (LRD), and self-similarity[4] is widely accepted, and has been discussed in a vast literature (Leland et al., 1993; Muscariello et al., 2005; Paxson and Floyd, 1995). Recent work argues convincingly that network traffic is much better modeled using statistically self-similar processes. However, such processes have much different theoretical properties than a Poisson process, and currently none of the mathematical models that present LRD or self-similarity allow an analytical solution when used for network traffic generation.

In this work, we are attempting to model behaviour from network flows for individual computers. Scale-invariant concepts such as self-similarity and LRD itself are therefore not essential to our modelling process as traffic from single machines has natural limits. However, what we should be concerned with are the large tails observed in the flow arrival distributions (Figure 4.2 depicting the tail behaviour will be presented in Section 4). These tails cannot be accurately modelled by a Poisson distribution whose variance is equal to its mean. Consequently, in a simple MMPP model tail observations receive

[3]A process is considered to have long-range dependence if its temporal dependence decays more slowly than an exponential, typically a power-like decay.

[4]A self-similar process behaves the same when viewed at different scales of time.

a disproportionately low likelihood, which in turn causes state changes in the sampled MMPP for observed tail event. Therefore, it is crucial to take the shape of the observed event arrival distribution into account.

Chapter summary

In this chapter, we discussed the advantages of accumulation intervals with regards to flow data and computational scalability. We introduced the concept of the MMPP and the batch MMPP, and developed a framework to sample the latent Markov process with given parameters $\mathbf{Q}$ and $\boldsymbol{\lambda}$ using the Forward-Backward algorithm and an appropriate mixture of Uniformization sampling and Rejection sampling. We showed how to build a Gibbs sampler for $P\Big(\{X_t\}, \mathbf{Q}, \boldsymbol{\lambda} | \{z_1, \ldots, z_n\}\Big)$. Finally, we demonstrated the shortcomings of a conventional MMPP approach on our data, and argued that the reason for this is a deviating tail behaviour of the flow arrival distribution. In the next chapter, we will develop a model that models these tails in an accurate way.

4. A hierarchical modification of the MMPP model

The advantage of Markovian models like MMPPs lies in the possibility of exploiting powerful analytical techniques to predict the network performance. As mentioned above, network traffic is more accurately modelled by statistically self-similar processes, which in turn do not allow for an analytical solution, even in simple traffic simulations.

Muscariello et al. (2005) suggest the use of a hierarchical Poisson model in order generate pseudo-LRD characteristics that match those measured on the Internet while still maintaining analytical tractability. The principle idea of their approach is to introduce a latent variable called a *session* which can be seen as an individual interaction of the person with the computer such as the opening of a specific website. Sessions are generated by an MMPP and trigger time-limited Poisson processes with events that are observed as flows. Muscariello et al. show that simulations generated by this model approximate the tail behaviour of flow data coherently. However, in their work they neither provide a general approach to fit the involved parameters nor a way to calculate or sample from $P\Big(\{X_t\}|\{z_1, \ldots, z_n\}\Big)$, which eventually is our ultimate interest in this work.

We now propose a model that adopts the idea of sessions acting as a latent variable, and that benefits from the use of accumulation intervals rather than raw arrival times. For this, we are drawing heavily from the batch MMPP framework discussed in Section 3.2. We will then embed our model in a simple, yet effective Bayesian framework that is capable of inferring the interaction state at a particular computer.

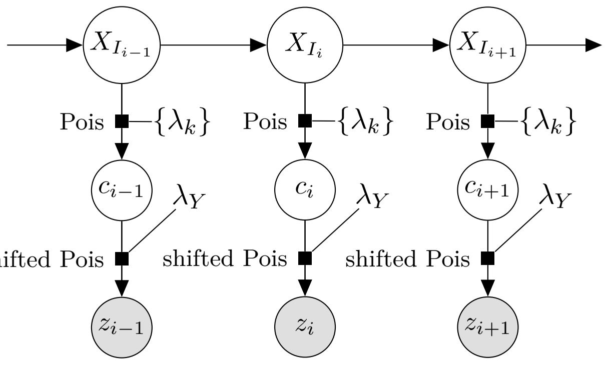

Figure 4.1.: Graphical model for our proposed hierarchical MMPP.

4.1. Model

As before, let $\{X_t\}$ for $t \in [0, t_{obs}]$ be the unobserved discrete Markov process with infinitesimal generator $\mathbf{Q}$, and let $N(t)$ be a Poisson process with rate $\lambda(t) = \lambda_{i=X_t}$. In contrast to the batch MMPP framework that we discuss in Section 3.2, $N(t)$ does not generate flow events, but unobserved events called *sessions* in reference to Muscariello et al.. Let $\{I_1 = [t_0 = 0, t_1), ..., I_n = [t_{n-1}, t_n = t_{\text{obs}})\}$ be our accumulation intervals, and let $c_k \triangleq N(t_k) - N(t_{k-1})$ be the number of sessions in I_k.

Upon generation, session s_i instantaneously generates y_i events that are observed. $y_i \in \mathbb{N}_{>0}$ is a discrete random variable with probability distribution $P(y_i|\mathbf{\Theta})$ with $\mathbf{\Theta}$ being the specific parameters. The generation of multiple flows at once rather than at staggered times is obviously a false assumption, but will be essential for analytical computability in our framework. However, since we are observing flow events in accumulation intervals instead of their raw timestamps, we avoid potential problems caused by this assumption.

$P(y_i|\mathbf{\Theta})$ can in principle be any discrete probability distribution. In this work we will assume that y_i follows a shifted Poisson distribution:

$$y_i - 1|\mathbf{\Theta} \sim \text{Poiss}(\lambda_Y). \tag{4.1}$$

The choice of a shifted Poisson distribution is beneficial since it allows for conjugate priors on the parameter λ_Y. Alternatives like geometric distributions were considered as well, but the shifted Poisson distribution provided the most promising results. The shift introduced ensures that every session generates at least one flow event.

The flow events z_k observed during interval I_k are now defined as

$$z_k \triangleq \sum_{i=1}^{l} y_i \mathbb{1}_{t_{k-1} < t'_i \leq t_k}. \tag{4.2}$$

Since the sum of Poisson-distributed variables is also Poisson-distributed, we can easily derive $P(z_k|c_k, \lambda_Y)$ to be

$$P(z_k|c_k, \lambda_Y) = \frac{(c_k \lambda_Y)^{z_k - c_k} \exp(-c_k \lambda_Y)}{(z_k - c_k)!}. \tag{4.3}$$

Figure 4.2 A depicts an excerpt of the Imperial College data during which the flow arrival rate supposedly stays constant, while 4.2 B shows the distribution of the observed counts during each interval. Plots C/D and E/F show a comparison with events generated from a Poisson process and from our proposed hierarchical Poisson model. λ for the Poisson process was chosen such that the data generated has the same mean as the Imperial College interval, while λ and λ_Y for the hierarchical model were chosen to adjust both the mean and the variance.

Table 4.1 shows the first three moments of the the data depicted in Figure 4.2. It is clear that our hierarchical Poisson model is better suited to imitate the tail behaviour of the observed data distribution.

Moment	IC data	Poisson	hierarchical Poisson
mean	0.73	0.73	0.71
variance	1.92	0.74	1.92
skewness	3.11	1.33	2.91

Table 4.1.: Comparison of the first, second, and third moment for the data depicted Figure 4.2.

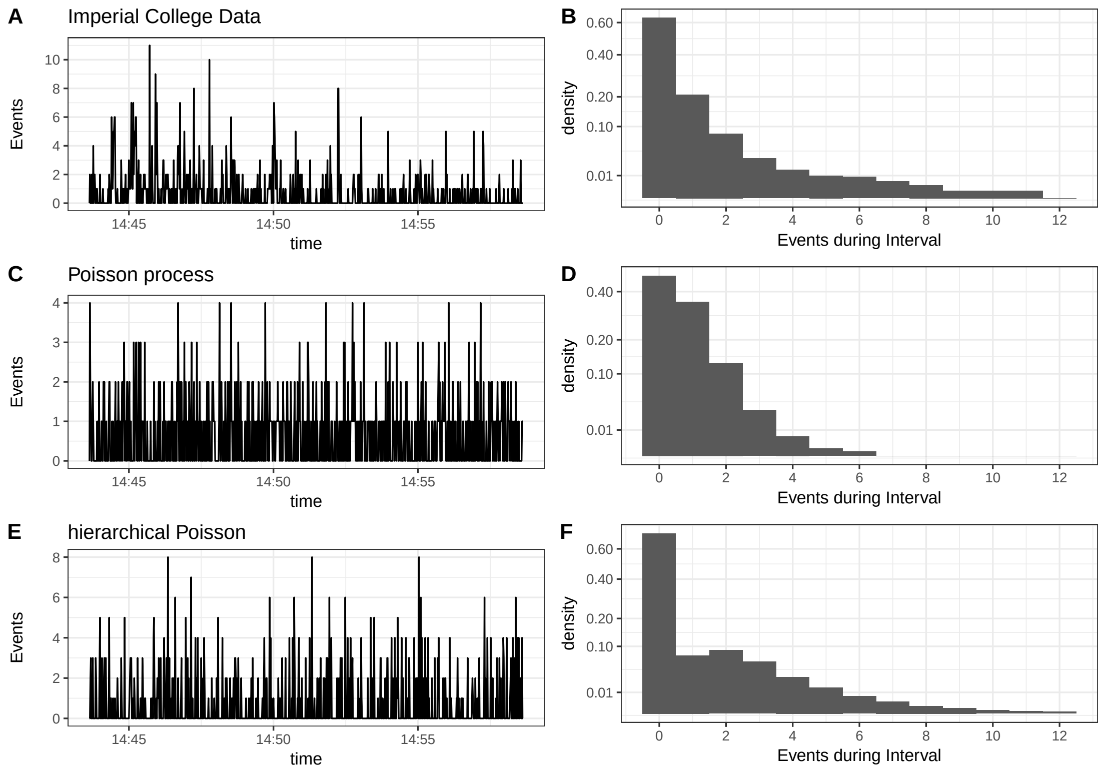

Figure 4.2.: Plot A shows an interval of the Imperial College flow data during which the flow arrival rate was supposedly constant (bin size 2 seconds). Plot B shows the corresponding empirical flow count distribution. Plot C and D show simulated arrivals from a Poisson processes with the same mean as the data in A. Plot E and F show simulated arrivals from the hierarchical Poisson model discussed in section 4.

4.2. Modified Gibbs sampler

To implement a Gibbs sampler for our model, we have to be capable of sampling from

$$\begin{aligned}
&1.\ P(\{X_t\}|\{z_1,\dots,z_n\},\{c_1,\dots,c_n\},\mathbf{Q},\boldsymbol{\lambda},\lambda_Y)\\
&2.\ P(\{c_1,\dots,c_n\}|\{z_1,\dots,z_n\},\{X_t\},\mathbf{Q},\boldsymbol{\lambda},\lambda_Y)\\
&3.\ P(\mathbf{Q},\boldsymbol{\lambda},\lambda_Y|\{z_1,\dots,z_n\},\{c_1,\dots,c_n\},\{X_t\})
\end{aligned} \tag{4.4}$$

1. Since we $\{z_1,\dots,z_n\}$ and $\{X_t\}$ are conditionally independent, we can rewrite

$$\begin{aligned}
1.\ &P(\{X_t\}|\{z_1,\dots,z_n\},\{c_1,\dots,c_n\},\mathbf{Q},\boldsymbol{\lambda},\lambda_Y)\\
&\qquad = P(\{X_t\}|\{c_1,\dots,c_n\},\mathbf{Q},\boldsymbol{\lambda})
\end{aligned} \tag{4.5}$$

from which we can sample in the same manner as described in Section 3.22.

2. We define

$$\lambda_{I_k} \triangleq \Big(\sum_{i=1}^{M}\int_{t_{k-1}}^{t_k} \mathbb{1}_{X_t=i}\cdot\lambda_i\,\mathrm{d}t\Big). \tag{4.6}$$

Since the increments of a Poisson process are independently Poisson-distributed, the number of sessions in I_k conditional on $\{X_t\}$ is Poisson-distributed with rate λ_{I_k}:

$$P(c_k|\{X_t,\ t\in I_k\},\boldsymbol{\lambda}) = \frac{\lambda_{I_k}^{c_k}\exp(-\lambda_{I_k})}{(c_k)!} \tag{4.7}$$

This allows us to calculate the posterior distribution of c_k:

$$\begin{aligned}
2.\ &P(c_k|z_k,\{X_t,\ t\in I_k\},\boldsymbol{\lambda},\lambda_Y)\\
&\propto P(c_k|\{X_t,\ t\in I_k\},\boldsymbol{\lambda})P(z_k|c_k,\lambda_Y)\\
&= \frac{\lambda_{I_k}^{c_k}\exp(-\lambda_{I_k})}{(c_k)!}\frac{(c_k\lambda_Y)^{z_k-c_k}\exp(-c_k\lambda_Y)}{(z_k-c_k)!}
\end{aligned} \tag{4.8}$$

Since $P(z_k < c_k|c_k) = 0$, this expression is easily normalizable. We can therefore also sample the sessions counts $\{c_1,\dots,c_n\}$ conditional on $\{Z_1,\dots,Z_n\}$, $\{X_t\}$, $\boldsymbol{\lambda}$ and λ_Y.

3. We saw that $z_k - c_k$ is Poisson-distributed with rate $c_k\lambda_Y$. If we impose a Gamma-prior on λ_Y, its posterior is given $\{z_k\}$ and $\{c_k\}$ is again Gamma-distributed:

$$\lambda_Y|\{z_k\},\{c_k\} \sim \Gamma(\alpha_{\lambda_Y} + z_k - c_k, \beta_{\lambda_Y} + z_k), \tag{4.9}$$

where α_{λ_Y} and β_{λ_Y} are the hyperparameters.
The prior and posterior distributions of $\mathbf{Q}$ and $\boldsymbol{\lambda}$ remain unchanged.

An description of the exact Gibbs-sampler used in this work is given in Algorithm 2.

```
**Algorithm 2:** Gibbs sampler for hierarchical MMPP model
Data: $\{Z_1, \dots, Z_n\}$
Initialize:
    1. Sample:
        $\lambda_i^{(0)} \sim \Gamma(\alpha_{\lambda,i}, \beta_{\lambda,i})$
        $q_{ii}^{(0)} \sim \Gamma(\alpha_{q,i}, \beta_{q,i})$
        $(q_{1,i}, ..., q_{i-1,i}, q_{i+1,i}, ..q_{M,i})^{T^{(0)}} \sim \mathrm{Dir}(\boldsymbol{\alpha}_{D,i}) \cdot q_{ii}$
        $\lambda_Y^{(0)} \sim \Gamma(\alpha_{\lambda_Y}, \beta_{\lambda_Y})$
    2. Sample $\{c_1, ..., c_n\}^{(0)}$ from $P(c_i|z_i, \lambda_Y^{(0)})$
    3. Sample $\{X_{t_1}, \dots, X_{t_n}\}^{(0)}$ recursively using the Forward-Backward algorithm
       conditional on $\{c_1, \dots, c_n\}^{(0)}$, $\boldsymbol{\lambda}^{(0)}$, and $\mathbf{Q}^{(0)}$ as described in Equation 3.20
    4. Sample $\{X_t\}^{(0)}$ conditional on $\{X_{t_1}, \dots, X_{t_n}\}^{(0)}$ using Equation 3.22
for $a \in \{1, \dots,$ number of desired samples$\}$ do
    1. Sample $\{c_1, \dots, c_n\}^{(a)}$ from $P\Big(c_k^{(a)}|z_k, \{X_t,\ t \in I_k\}^{(a-1)}, \lambda_Y^{(a-1)}\Big)$ as defined in
       Equation 4.8
    2. Sample $\{X_{t_1}, \dots, X_{t_n}\}^{(a)}$ recursively using the Forward-Backward algorithm
       conditional on $\{c_1, \dots, c_n\}^{(a)}$, $\boldsymbol{\lambda}^{(a-1)}$, and $\mathbf{Q}^{(a-1)}$ as described in Equation 3.20
    3. Sample $\{X_t\}^{(a)}$ and $\{t'_1, \dots, t'_l\}^{(a)}$ conditional on $\{X_{t_1}, \dots, X_{t_n}\}^{(a)}$ using
       Equation 3.22
    4. Calculate $\tilde{t}_i^{(a)}$, $n_i^{(a)}$, and $r_{ij}^{(a)}$ via equations 3.26-3.28 from $\{X_t\}^{(a)}$ and
       $\{t'_1, \dots, t'_l\}^{(a)}$
    5. Sample:
        $\lambda_i^{(a)} \sim\sim \Gamma(\alpha_{\lambda,i} + n_i,^{(a)} \beta_{\lambda,i} + \tilde{t}_i^{(k)})$
        $q_{ii}^{(a)} \sim \Gamma(\alpha_{q,i} + \sum_{j \neq i} r_{ij}^{(a)}, \beta_{q,i} + \tilde{t}_i^{(a)})$
        $(q_{1,i}, ..., q_{i-1,i}, q_{i+1,i}, ..q_{M,i})^{T^{(a)}} \sim \mathrm{Dir}(\boldsymbol{\alpha}_{D,i} + \mathbf{r}_i)^{(a)} \cdot q_{ii}$
        $\lambda_Y^{(a)} \sim \Gamma(\alpha_{\lambda_Y} + z_k^{(a)} - c_k^{(a)}, \beta_{\lambda_Y} + z_k^{(a)})$
```

4.3. Results and model comparison

We can now proceed to apply Algorithm 2 to the given data sets. We will discuss the choice of prior hyperparameters, number of states, and convergence explicitly for the Imperial College data before presenting the calculated sample distribution of the underlying Markov process for both data sets.

Hyperparameter selection

The first thing we have to do is choose a number of an appropriate number of states and a set of hyperparameters for our priors. We will start here with fitting a four-state model before discussing the eventually appropriate numbers in section 4.3.

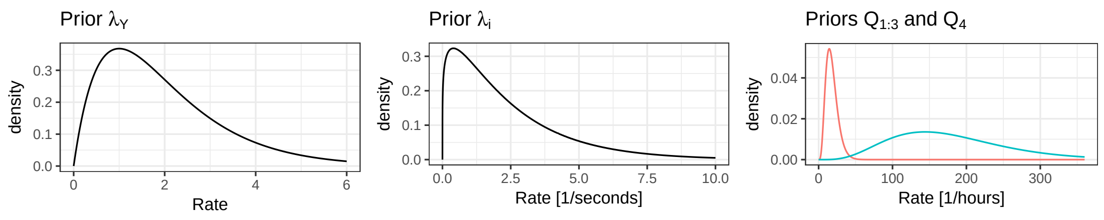

Figure 4.3.: Prior distributions of λ_Y, λ_i, and Q_{ii} for $i \in \{1,2,3\}$ (red) and $i = 4$ (blue).

For all our parameters, we have to make sure that we choose our priors wide enough to allow for proper convergence of the posterior, but narrow enough to initialize the parameters correctly. It is therefore helpful to look at the given data in order to incorporate some information into the design of our priors.

λ_Y:
We are not expecting the number of flows generated by each session to be excessively high. From experience, we know that almost in general $\lambda_Y \in [0.5, 10]$. An appropriate prior on λ_Y would therefore be given by

$$\alpha_{\lambda_Y} = 2, \qquad \beta_{\lambda_Y} = 1. \tag{4.10}$$

λ_i:
To allow for a wide state space exploration, we want to employ a prior that does not impose any restrictions on the state's individual Poisson rates. However, we do observe an upper and lower limit in the possible Poisson rates. The highest rate of flow events observed in the data set can be seen in the large spikes (around 50 events per second) while the lowest rate lowest rate of arrival events observed in the data set (seen at 15:50 with around 250 events per hour, 0.07 events per second). Please keep in mind that since λ_i represents the rate of unobserved session events, it will be considerably lower (as mentioned above between 1/2 and 1/10) than the observed rates. However, we can still infer an approximate region for λ_i with this information.

We will employ the same prior on all λ_i. Since there lie several orders of magnitude between the upper and lower limit of the rates, we would like a prior that favours smaller values while not suppressing larger ones in order to have enough resolution for values of smaller magnitude during initialization. We therefore choose the following hyperparameters for our Gamma-prior:

$$\alpha_{\lambda,i} = 1.2\frac{1}{\text{sec}}, \qquad \beta_{\lambda,i} = 1. \tag{4.11}$$

In order to impose an ascending order on the state rates, we will sort the rates sampled during initialization (explained in Section 4.3 for more details).

q_{ii}:

Since the Imperial data set is limited to three hours, we can assume that no state has a half life exceeding this number. Furthermore, we dismiss any unreasonably high decay rates leading to half lifes under 1 minutes with the exception of state 4 in order to allow it to catch the faster decaying spikes observed in our data:

$$\begin{aligned} &\alpha_{q,1} = \alpha_{q,2} = \alpha_{q,3} = 5\frac{1}{\text{sec}}, \\ &\beta_{q,1} = \beta_{q,2} = \beta_{q,3} = 1000, \\ &\alpha_{q,4} = 5\frac{1}{\text{sec}}, \qquad \beta_{q,4} = 100. \end{aligned} \tag{4.12}$$

$q_{i\neq j}$:

Since we have no knowledge about the relative transition probability of state i into state j, we employ an uninformative Dirichlet prior with $\boldsymbol{\alpha}_{D,i} = (1, 1, 1)^T$.

Interval length

Increasing the accumulation interval length usually leads to a decrease of computational operations and computation time when sampling $\{X_t\}, \mathbf{Q}, \boldsymbol{\lambda}, \lambda_Y | \{z_1, ..., z_n\}$. However, increasing the interval length heavily above the half life of any of the Markov states distorts the estimation of the decay rates q_i and should be avoided.

To fit our algorithm to the Imperial College data set, we chose an interval length of 5 seconds.

Results

Using the above described settings, we generate 500 samples of $\{X_t\}, \mathbf{Q}, \boldsymbol{\lambda}, \lambda_Y | \{z_1, ..., z_n\}$ using Algorithm 2. Figure 4.4 depicts an excerpt of the sampled $\{X_t\}$ for the Imperial College data. The whole sampled trajectory can be seen in Figure B.2.

In comparison to Figure 3.3, the improvements of our model are apparent: While we are not able to identify any activity state coherently with the simple MMPP framework, our new model is able to distinguish several states of user activity throughout

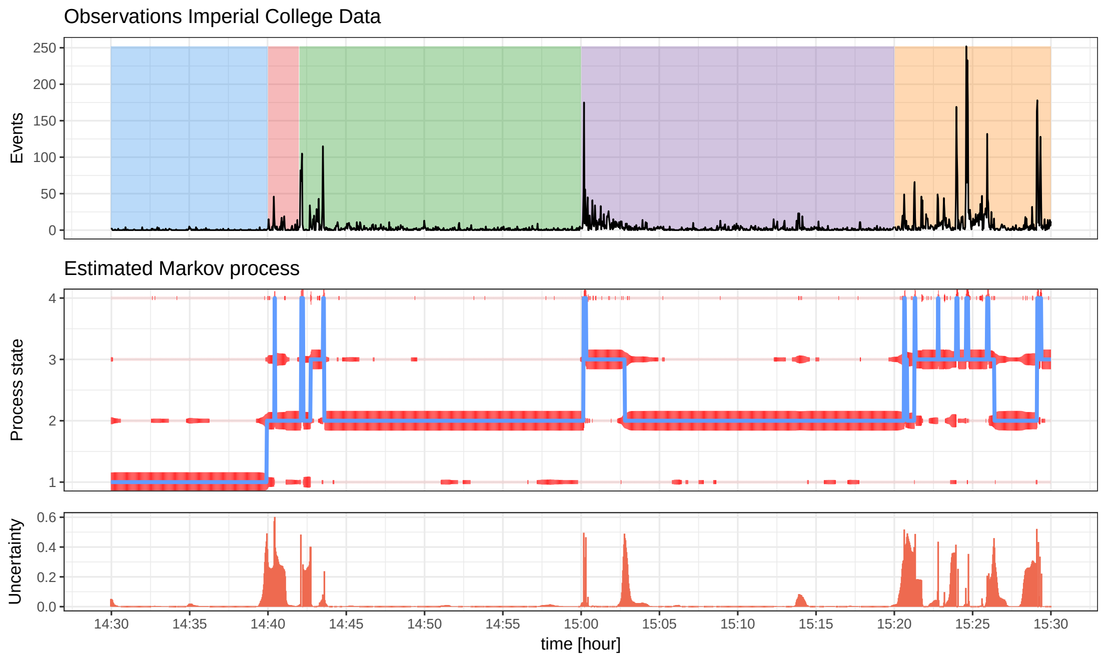

Figure 4.4.: Excerpt of sampled $\{X_t\}$ for the Imperial College data. Binned flow data (top), 500 samples of $\{X_t\}$ (using Algorithm 2) (middle), and uncertainty of the inferred state. The blue line indicates the sample mode at each point in time while the thickness of the red lines indicates the amount of samples in each state. The uncertainty is calculated by $1 - \alpha_{t_i}$ where α_{t_i} is the fraction of the sample mode of all samples at time t_i.

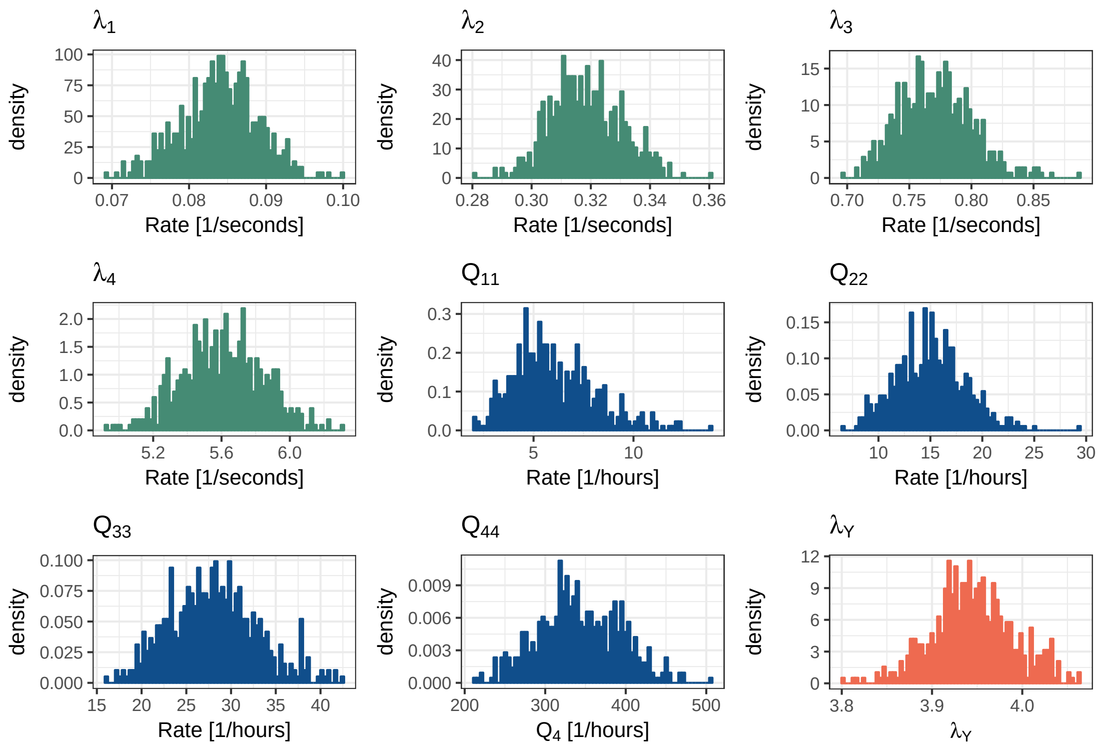

Figure 4.5.: Sampled posterior distributions of Q_{ii} (blue), λ_i (green), and λ_Y (red).

the whole data. The identified states are in good correspondance to the actual activity on the computer, i.e. similar experiment phases are identified by the same state. The state estimates are consistent and stable during each activity phase. As expected, the described DNS are identified as a separate state. Furthermore, the overall uncertainty of the $\{X_t\}$ samples is small. An exception are points where the user activity changes, since an exact estimation of the state change is difficult. A detailed interpretation of the individual states will be presented in Section 4.3.

Figure 4.5 shows the sampled marginal posterior distributions of $\mathbf{Q}$, $\boldsymbol{\lambda}$, and λ_Y. All λ_i are sampled well within our previously estimated range, as is λ_Y. Furthermore, the decay rates Q_i for state 1 to 3 are sampled in a reasonable range, corresponding to state half lifes of 2-15 minutes, which is a good reflection of the activity phases in the experiment. Since state 4 is capturing the above described DNS-spikes, the values for Q_4 are much higher, corresponding to a half-life of around 9 seconds.

Plots for the sampled posterior distributions of the parameters Q_{ij}/Q_{ii}, which indicate the probability to move from state i to state j once a state change occurs, are given in Figure B.3. As visible, the posterior distribution for all possible state changes is non-vanishing. Since the number of activity state changes is comparably small, we do not expect the posterior distributions to deviate strongly from our prior distribution. The sampled values of Q_{ij}/Q_{ii} are therefore of little interest.

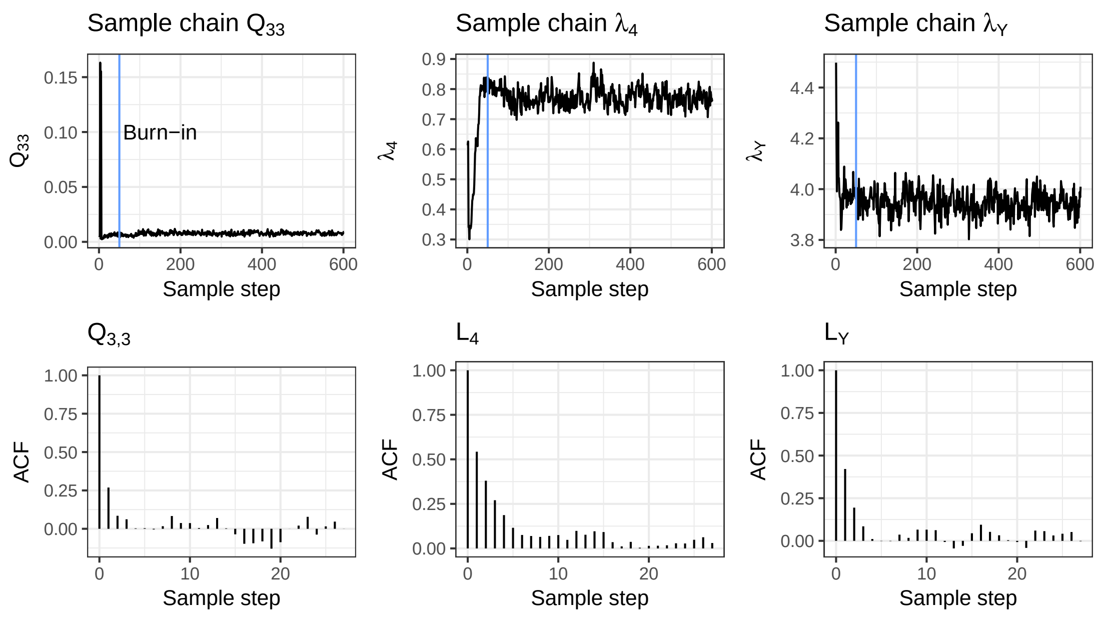

Figure 4.6.: Trace and autocorrelation plots for selected parameters. The blue line indicates the discarded burn-in.

Convergence

The posterior distributions of the generated sample chain of $\mathbf{Q}$, $\boldsymbol{\lambda}$, and λ_Y, depicted in Figure 4.5, all appear to be well explored with a single pronounced mode. There is an important aspect to be considered however: Since we impose the same priors on the state parameters $\mathbf{Q}_i$ and λ_i for every i, the posterior distribution is invariant under state label permutation, i.e.

$$\begin{aligned} P(\lambda_i = a, \lambda_j = b, \mathbf{Q}_i = c, \mathbf{Q}_j = d|\{z_1, \ldots, z_n\}) = \\ P(\lambda_i = b, \lambda_j = a, \mathbf{Q}_i = d, \mathbf{Q}_j = c|\{z_1, \ldots, z_n\}). \end{aligned} \tag{4.13}$$

The interchangeability of each state causes the posterior distribution $P(\mathbf{Q}, \boldsymbol{\lambda}|\{z_1, \ldots, z_n\})$ to have multiple equivalent modes, which makes a complete exploration of the posterior with MCMC techniques difficult to achieve. The non-identifiability of the components under symmetric priors is known as the *label switching* problem (Jasra et al., 2005), and is frequently observed in Bayesian mixture models. Due to the equivalence of the posterior distribution under state label permutation, it is sufficient to explore one of the posterior modes to describe the posterior distribution fully.

In order to have a consistent sample output, it often desired to suppress label switching, i.e. to constrain the sample chain to one mode. Since the marginal posterior distributions of our sampled parameters are well separated, we do not observe mode mixing. However, if the marginals would show overlap, additional post-processing to re-label the states would be an appropriate method to keep the output consistent.

Figure 4.6 depicts trace plots and autocorrelation plots of our sampled chain for selected parameters. All parameters show a fast convergence towards a stationary distribution, and a low autocorrelation, indicating a great efficiency of our sampler. The first 50 samples were discarded as burn-in.

Model comparison

An important question that arises in the context of finite mixture models for example is the choice of an appropriate number of mixture components. Although likelihood ratio tests and the BIC criterion are often appropriate for independent mixture models, the sequential nature of finite state hidden Markov models leads to a stochastically unbounded likelihood ratio and to an underpenalisation of the likelihood by the BIC (Zhang and Siegmund, 2007).

Another promising criterion is the cross-validated likelihood criterion, which has been proposed by Smyth (2000) in the case of independent mixtures. An advantage of this criterion is that it seems to avoid some of the theoretical difficulties occurring with penalised likelihood criteria and unrealistic assumptions regarding the distribution of the data. If the data of the HMM consists of several independent sequences, cross-validated likelihood can be applied to with the different sequences acting as individual training and validation sets. If the data is given by a single sequence, the choice of independent training and validation sets is more difficult due to the sequential dependence of the data.

We will employ a deterministic half-sampling procedure for finite state-space HMMs proposed by Celeux and Durand (2008), which splits the data into a set with odd indices and a set with even indices:

We assume that we have an even number $n = 2m$ of observations $\{z_1, \dots, z_n\}$. Define the index set $U_1 \triangleq \{1, 3, \dots, n-1\}$. Define $U_2 \triangleq \{2, 4, \dots, n\}$. We define the two training and validation sets $T_1 \triangleq \{z_{U_1}\}$, $T_2 \triangleq \{z_{U_2}\}$.

We then proceed to calculate validation of T_2 under T_1 (and vice versa) by estimating the parameter sample posterior distribution $P_{T_1} = P(\mathbf{Q}, \boldsymbol{\lambda}, \lambda_Y | T_1)$ using Algorithm 2, and afterwards calculating the likelihood of T_2 under P_{T_1} via integration:

$$\begin{aligned} P(T_2|P_{T_1}) &= \int P(T_2|\mathbf{Q}, \boldsymbol{\lambda}, \lambda_Y)\, \mathrm{d}P_{T_1} \\ &\approx \frac{1}{N} \sum_{k=1}^{N} P(T_2|\mathbf{Q}_k, \boldsymbol{\lambda}_k, \lambda_{Yk}) \end{aligned} \tag{4.14}$$

where N is the number of generated samples of P_{T_1}. $P(T_2|\mathbf{Q}_k, \boldsymbol{\lambda}_k, \lambda_{Yk})$ can be calculated via

$$\begin{aligned} P(T_2|\mathbf{Q}_k, \boldsymbol{\lambda}_k, \lambda_{Yk}) = \int P(T_2|\{c_1, \dots, c_n\}) \cdot \\ P(\{c_1, \dots, c_n\}|T_2, \mathbf{Q}_k, \boldsymbol{\lambda}_k, \lambda_{Yk})\, \mathrm{d}\{c_1, \dots, c_n\} \end{aligned} \tag{4.15}$$

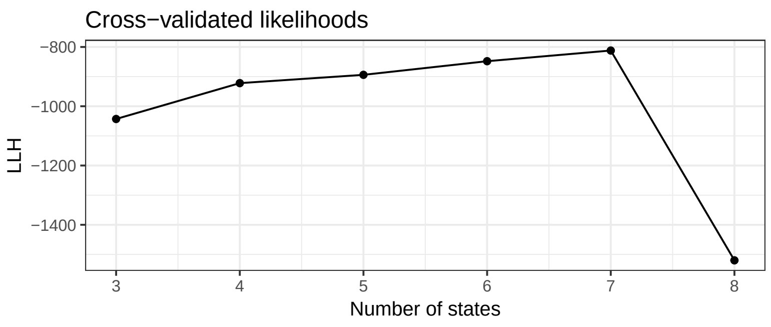

Figure 4.7.: Cross-validation log-likelihoods for different state numbers.

with $P(\{c_1, \ldots, c_n\}|T_2, \mathbf{Q}_k, \boldsymbol{\lambda}_k, \lambda_{Yk})$ being calculated via 2 with fixed parameters, and $P(T_2|\{c_1, \ldots, c_n\})$ being given by Equation 3.8.

The splitting of the data with the described method changes the posterior distribution of the decay rates Q_{ii} for the training and the validation set in the same way, in our model it doubles them since we on average leave out half the time until the next state change appears. The likelihood $P(T_2|\{c_1, \ldots, c_n\})$ (resp. T_2) does not directly depend on Q_{ii}, and the remaining parameter distribution remain unchanged. Therefore this approach does not alter the validation likelihood.

Figure 4.7 depicts the cross-validation log-likelihoods for state numbers from 3 to 8, with 7 states having the highest cross-validated likelihood. Figure 4.8 and B.4 depict the sampled posterior of $\{X_t\}$ for 7 states.

4.4. Result interpretation

The interpretation of the **4-state model** in terms of human activity results are straightforward:

- State 1 corresponds to pure standby activity, no human actions are taken nor are any programs open.
- State 2 corresponds to the presence of an inactive user, i.e. the internet browser or a similar program is open, but no actions are taken. It is remarkable that the watching of internet videos or ssh-communcation create flow events at a similar rate and is therefore identified as state 2. Differences between these two activities are however visible when looking at the length and the size of the transmitted flows.
- State 3 identifies surfing activity of the user.
- State 4 captures the above described DNS spikes. Since these spikes are very short, they do not correspond to a change in user activity, but are more or less a nuisance. State 4 should therefore also be seen as a nuisance state that ensures

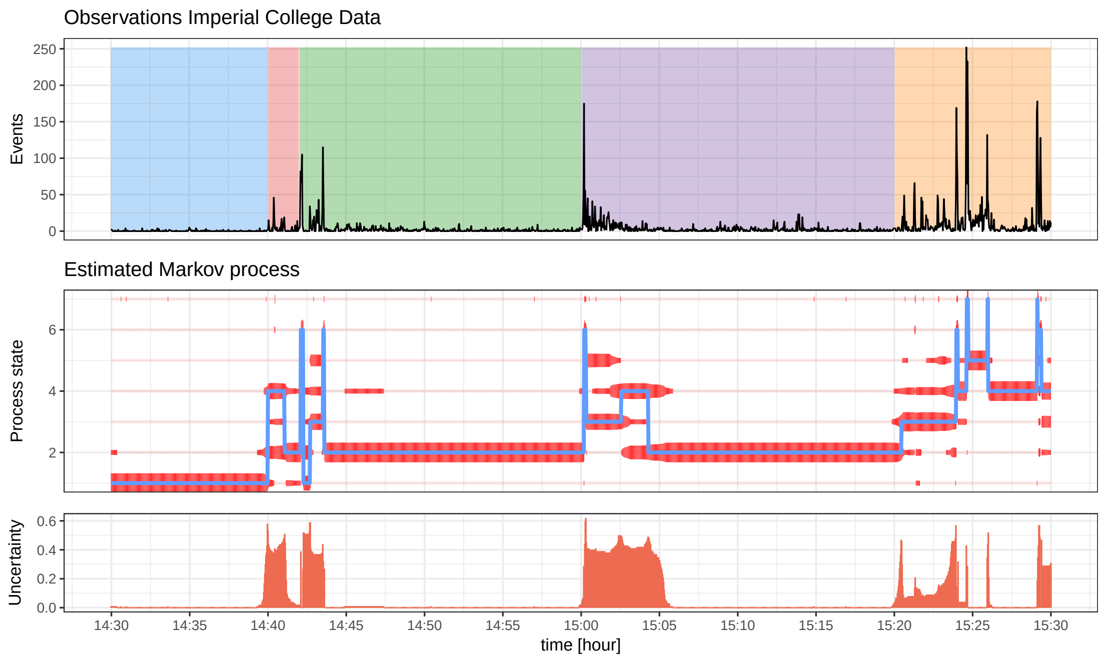

Figure 4.8.: Excerpt of sampled $\{X_t\}$ for the Imperial College data with a 7 state model.

the model stability without indicating a user state. In a cyber-security framework, the identification of the spikes through state 4 might also be of other interest.

Obviously, our framework models the state of the user only as perceived through the machine. For instance, we observe multiple short drops of the perceived activity from state 3 to state 2, indicating user inactivity, while it is most likely that the user just stopped his actions shortly to read something of interest or similar. To make the infer from activity on the computer to a general state of the person sitting behind the computer, we need additional assumptions and post-process modeling. All in all, there is a great correspondence of these four activity states that are sampled with our model, and the actual user activity.

The interpretation of the **7-state model** is a bit more difficult. State changes occur while the activity state of the user does not change, and therefore do not model user activity in a one-to-one correspondance. The uncertainty of our predictions however only increases slightly. It can therefore rather be concluded that an individual user activity state does not correspond to a single flow arrival rate, but possesses a more nuanced spectrum of arrival rates. This nuanced spectrum can be captured more accurately by an increase of the Markov process state number. A direct identification of the user state is however is made more complicated.

We can conclude that a model with a small number of Markov states is able to distinguish between user absence, presence, and activity very well. However, different user activities with similar flow generation rates are identified by the same state, such as ssh-

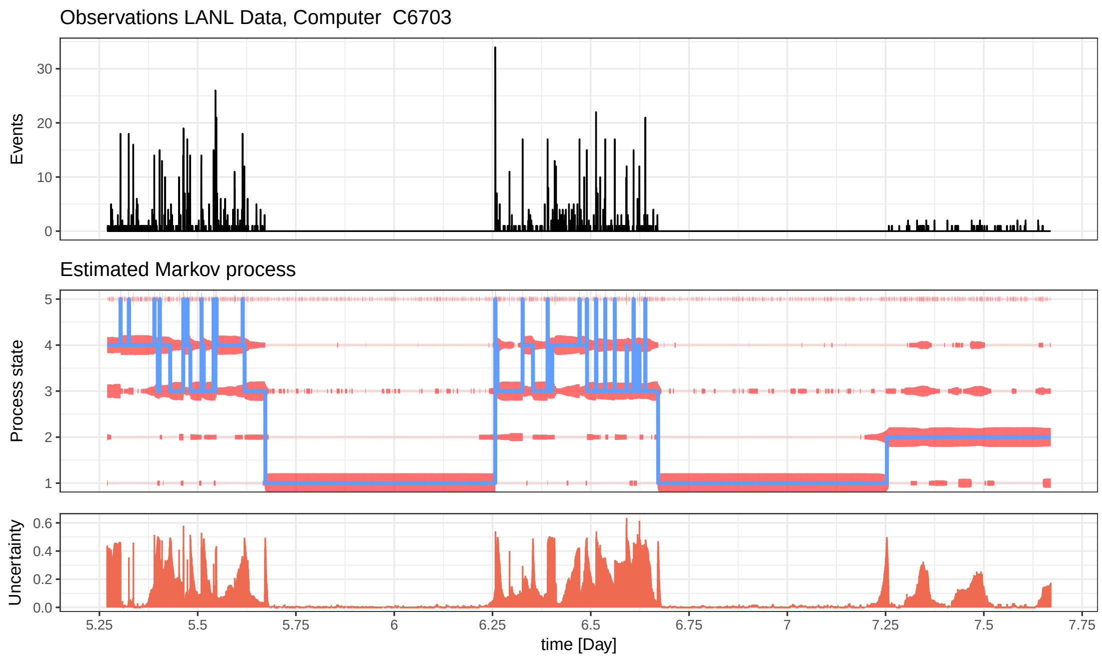

Figure 4.9.: Excerpt of sampled $\{X_t\}$ for a selected computer in the LANL data.

activity and an open browser. It might therefore be of interest to have a state space with a higher resolution to model user activity. Increasing the state number of the Markov process does increase the resolution with which we observe different flow arrival states in our data, but it does not aid us in the discrimination of different activity states. Since different user activities often have similar flow arrival rates, it is difficult to distinguish them through the number just through the observed flow arrivals. In Section A.2, we suggest a possible extension of our model that also incorporates other information such as the length of the generated flows in order to build a more complete model of user activity

4.5. Results for LANL data

Inference for the LANL data can be done in the same way as for the Imperial data. All we have to do is to update the prior hyperparameters:

$$\begin{aligned}
\alpha_{\lambda,i} &= \frac{1}{\text{day}}, \qquad \beta_{\lambda,i} = 0.0006 \ \forall i \in \{2,3,4,5\}, \\
\alpha_{\lambda,1} &= \frac{0.00001}{\text{day}}, \qquad \beta_{\lambda,1} = 1, \\
\alpha_{q,i} &= \frac{3}{\text{day}}, \qquad \beta_{q,i} = 0.6 \ \forall i \in \{1,2,3,4,\}, \\
\alpha_{q,5} &= \frac{5}{\text{day}}, \qquad \beta_{q,5} = 0.002.
\end{aligned} \tag{4.16}$$

where we considered the fact we observe large periods during which the computer is off and no flows are generated (i.e. $\lambda_1 \approx 0$), and that the decay rate for the state that is supposed to capture the observed spikes is larger than for the other states. We chose an accumulation interval length of 30 seconds.

Figure 4.9 depicts the sample posterior distribution of the Markov process $\{X_t\}$ for a selected computer. The corresponding parameter posteriors can be found in Figure B.5, and further plots for selected computers can be found in the appendix chapter C. Cross-validation of the likelihood as described in Section 4.3 suggests that the optimal number of states is either 5 or 6 with similar cross-validated likelihoods for both numbers.

Since we have no absolute knowledge about the activity state of the computer users for the LANL data, the evaluation and interpretation of our results is more difficult than for the Imperial College data. Still, phases during which the computer is off are safely identified, and our model is able to distinguish days during which the user is obviously active (day five and six) from days during which the user is most likely not logged in (day 7). Furthermore, phases with different flow arrivals distinguished on day five and day six that are imminent in the data. The question remains if these phases truly correspond to different states of user activity.

It is noticeable that the uncertainty of $\{X_t\}$ is higher than for the Imperial College data. This can be explained by the fact that the LANL data only internal traffic. For this reason, the number of flows generated by a computer (1303 events in three days for machine C6703) is much lower than for the Imperial College data (28488 events). The posterior distributions of the parameters and the Markov process are thus estimated with much less precision (also visible in Figure B.5). To achieve greater precision, more data is necessary.

Chapter summary

In this chapter, we introduced a hierarchical model that is based on a batch MMPP, and we showed how to inference and parameter estimation can be done in our new model by extending the Gibbs sampler of Fernhead and Sherlock. We demonstrated that our model addresses the issues of a conventional MMPP in regards of the tail distribution of network traffic. We then demonstrated how choose appropriate prior hyperparameters and accumulation interval lengths, and applied our developed framework to the Imperial College data and a selected computer from the LANL data. The results prove that our framework is capable of inferring a reasonable Markov process from the given data, with convergent parameter posterior distributions in a reasonable and consistent range. We discussed the interpretation of our results in regards of human activity and linked them to the actual activity that was taking place during our experiment, concluding that our framework captures general activity states in an accurate way. Finally, we discussed some difficulties regarding the incapability of distinguishing activities with similar flow arrival rates, and the existing of a more nuanced rate spectrum for individual activities.

5. Conclusion and outlook

In this thesis, we have established a novel Bayesian framework that is able to identify temporal patterns in the network flow generation of individual personal computers. The consistency of our framework was demonstrated by applying it on 10 computers from the LANL enterprise network. A controlled experiment conducted during this project verified that the identified patterns can be linked quite closely to states of human activity.

As we have shown, flow arrivals cannot be modeled accurately by the well-established *Markov modulated Poisson process* due to strong deviations in the arrival distribution. We proposed a new hierarchical model, based on the MMPP, that addresses this problem sufficiently while retaining the computational simplicity and scalability of a conventional MMPP-based model. Our model extends the MMPP model adding a latent layer that separates the observations from the observations. This approach is inspired by a network traffic simulation model by Muscariello et al. (2005) and reflects the physical process of flow generation to some extend. We then adopted the concept of the exact Gibbs sampler by Fearnhead and Sherlock (2006) and extended it to incorporate our proposed model. Our model is fully Bayesian, and samples the posterior distribution of the Markov process that represents the user's activity state. Our implementation samples 500 process trajectories in less than a minute for approximately 30,000 observed flow events.

Our model has identified different activity states on multiple computers in the LANL data set, and we validated our results with data from a controlled experiment within the Imperial College network. However, we have seen that the interpretation of the results is not straightforward as the correspondence between user activity states and flow arrival rates is not always one-to-one, i.e. individual user activity states can correspond to more than one arrival rate. Models with fewer latent states of the Markov process therefore yield, but clearer distinction of individual states. Models with a larger state number however might provide a good resolution in the distinction of individual activities when combined with other quantities in the flow log. In order to achieve this, it is possible to extend our model to incorporate additional values collected in flow logs to build a finer distinction of user activities. A concrete plan for a possible model extension incorporating cumulated connection lengths that would retain the analytical and computational benefits of our current model is described in Section A.2.

The estimation of human activity states has direct applications in modelling human behaviour in order to identify intruders operating inside enterprise computer networks, and is intended as a building block in a larger Bayesian cyber-security model. Several other potential applications such as Internet traffic modelling for engineering and performance evaluations, or user monitoring in online marketing, might benefit from both its advantages over conventional MMPP models and its computational scalability. Furthermore, our model can be used for multiple applications outside of network modelling in which events arrivals are not exactly Poisson distributed. The release of heavily optimised computational routines is planned in the form of an R-C++-package.

Bibliography

L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov process. *Inequalities*, 3:1–8, 1972.

L. Breuer. An EM algorithm for batch Markovian arrival processes and its comparison to a simpler estimation procedure. *Annals of Operations Research*, 112(1):123–138, 2002.

T. Burzykowski, J. Szubiakowski, and T. Rydén. Analysis of photon count data from single-molecule fluorescence experiments. *Chemical Physics*, 288(2):291–307, 2003.

G. Celeux and J.-B. Durand. Selecting hidden Markov model state number with cross-validated likelihood. *Computational Statistics*, 23(4):541–564, 2008.

N. Y. Conteh and M. D. Royer. The rise in cybercrime and the dynamics of exploiting the human vulnerability factor. *International Journal of Computer (IJC)*, 20(1):1–12, 2016.

P. A. Devijver. Baum's Forward-Backward algorithm revisited. *Pattern Recognition Letters*, 3(6):369–373, 1985.

P. Fearnhead and C. Sherlock. An exact Gibbs sampler for the Markov-modulated Poisson process. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(5):767–784, 2006.

A. Hobolth and E. A. Stone. Simulation from endpoint-conditioned, continuous-time Markov chains on a finite state space, with applications to molecular evolution. *The Annals of Applied Statistics*, 3(3):1204âĂŤ1233, 2009.

A. Jasra, C. C. Holmes, and D. A. Stephens. Markov chain Monte Carlo methods and the label switching problem in Bayesian mixture modeling. *Statistical Science*, pages 50–67, 2005.

A. D. Kent. Comprehensive, multi-source cyber-security events data set. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2015.

W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of Ethernet traffic. In *ACM SIGCOMM Computer Communication Review*, volume 23, pages 183–193. ACM, 1993.

L. Muscariello, M. Mellia, M. Meo, M. A. Marsan, and R. L. Cigno. Markov models of internet traffic and a new hierarchical MMPP model. *Computer Communications*, 28 (16):1835–1851, 2005.

J. Neil, C. Hash, A. Brugh, M. Fisk, and C. B. Storlie. Scan statistics for the online detection of locally anomalous subgraphs. *Technometrics*, 55(4):403–414, 2013.

V. Paxson and S. Floyd. Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking (ToN)*, 3(3):226–244, 1995.

T. Rydén. An EM algorithm for estimation in Markov-modulated Poisson processes. *Computational Statistics & Data Analysis*, 21(4):431–447, 1996.

P. Smyth. Model selection for probabilistic clustering using cross-validated likelihood. *Statistics and computing*, 10(1):63–72, 2000.

C. Tankard. Advanced persistent threats and how to monitor and deter them. *Network security*, 2011(8):16–19, 2011.

R. Walters. Cyber attacks on US companies in 2014. *The Heritage Foundation*, 4289: 1–5, 2014.

N. R. Zhang and D. O. Siegmund. A modified Bayes information criterion with applications to the analysis of comparative genomic hybridization data. *Biometrics*, 63(1): 22–32, 2007.

A. Additional theory

A.1. Simulation from an end-point constraint CTMP on finite state space

Let $\{Z_t,\ t \in [0,T]\}$ be a CTMP with generator matrix Q. Let $Z_{t_k} = a,\ Z_{t_{k+1}} = b$.

Modified Rejection Sampling

If $a = b$:

1. Sample $\{Z_t,\ \ 0 \leq t \leq T\}$ using simple forward sampling with $Z_0 = a$.
2. If $Z_T = b$, accept the sample, otherwise return to 1.

If $a \neq b$:

1. Sample τ from
$$f(\tau) = \frac{Q_{aa}e^{-\tau Q_{aa}}}{1 - e^{-TQ_{aa}}}, \qquad 0 \leq \tau \leq T, \tag{A.1}$$
and choose a new state $c \neq a$ from a discrete probability distribution with masses $-Q_{ac}/Q_{aa}$
2. Sample $\{Z_t,\ \ t_k + \tau \leq t \leq t_{k+1}\}$ using simple forward sampling with $Z_\tau = a$.
3. If $Z_T = b$, accept the sample, otherwise return to 1.

Modified rejection sampling is particularly fast if the probability mass $-Q_{ac}/Q_{aa}$ for all states Z_t can transition to from a is concentrated on a small number of states. However, if Z_t requires a lot of state transitions to get from a to b, modified rejection sampling will almost certainly fail, and is therefore inappropriate for us to use to sample V_t during intervals with a high observed c_k.

Uniformization

Define $\mu = \max_c -Q_{cc}$. Define
$$R = I + \frac{1}{\mu}Q, \tag{A.2}$$
and define the probability density of the number of state changes N as
$$P(N = n|Z_0 = a, Z_T = b) = e^{-\mu T}\frac{(\mu T)^n}{n!}\frac{R^n_{ab}}{e^{-QT}_{ab}}. \tag{A.3}$$

1. Sample n from A.3.
2. If $n = 0$, we are done: $Z_t = a,\ 0 \leq t \leq T$.
3. If $n = 1$ and $a = b$, we are done: $Z_t = a,\ 0 \leq t \leq T$.
4. If the $n = 1$ and $a \neq b$, sample t_1 from a uniform distribution in $[0, T]$, and set $Z_t = a,\ 0 \leq t \leq t_1$, and $Z_t = b,\ t_1 \leq t \leq T$.
5. If $n \geq 2$, draw n independent samples from a uniform distribution in $[0, T]$, and sort them in order as $\{t_1 \leq \dots \leq t_n\}$. Simulate $Z_{t_1}, \dots, Z_{t_n}$ from a discrete-time Markov chain with transition matrix R conditional on $X_0 = a$, and $Z_{t_n} = b$.

In step 1 above, we sample n by drawing $u \sim \text{Unif}(0, 1)$, and letting n be the first time the cumulative sum of A.3 exceeds u.

Note that we allow virtual state changes, in which a jump occurs but the state does change. This makes Uniformization sampling comparably inefficient if the number of actual state changes is much smaller than $\mu \cdot T$. However, we are guaranteed to find a sample path in one iteration.

For large matrices transition matrices Q, the biggest computational load during Uniformization sampling is the calculation of R^n for each n. When simulating $\{V_t\}$ for our MMPP, R stays constant for each interval I_k. We can therefore calculate R^n up to a chosen limit n_l, and store all R^n before starting the simulation of $\{X_t\}$ to greatly decrease the computation load.

To simulate the whole trajectory of $\{X_t\}$, we need to simulate $\{V_t\}$ for each accumulation interval I_k. If $c_k \leq 10$ and $X_{t_k} = X_{t_{k+1}}$ (sampled through Forward-Backward algorithm), we will use Modified Rejection sampling. Else, we will use Uniformization using the prior calculated $R^n,\ n \in \{0, \dots, n_l\}$.

The sparse block-structure of R principally enables the use of faster block multiplication techniques if needed. This was however not investigated in this work.

A.2. Possible extension incorporating flow durations

Our current model is based solely on the arrival times of flow events. A possible and promising addition that might help to build a finer distinction of user activities is the incorporation of connection durations. Concretely, we propose the use of the cumulated time l_k of all connections inside of each accumulation interval I_k. Following the proposal of Muscariello et. al, l_k can be modelled as being exponentially distributed with a rate governed by another latent variable c'_k, similar to the earlier introduced sessions.

Modelling connection lengths is promising since they are limited by the user's presence, in contrast to transmitted traffic size. However, an overwhelming amount of transmitted flows is recorded with duration zero. This makes it difficult to model the total flow lengths in an interval in dependence of the number of flows.

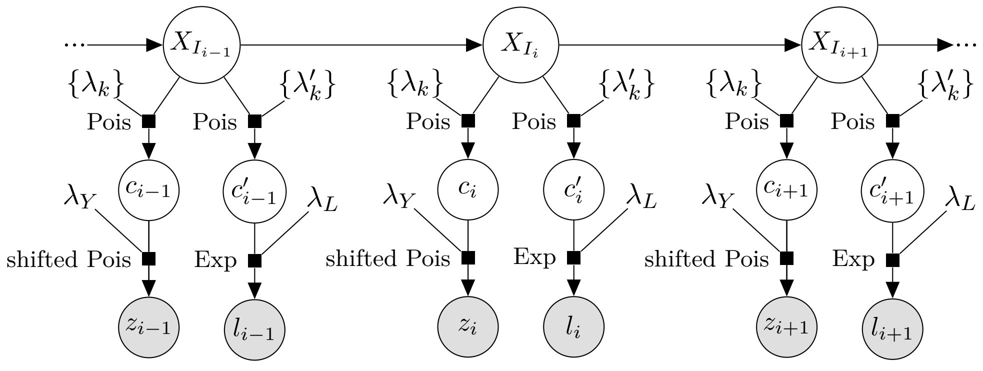

Figure A.1.: Graphical model for a possible extension of our proposed model that incorporates observed flow lengths.

We add another set of Markov state dependent arrival rates $\boldsymbol{\lambda}' \triangleq \{\lambda'_1, \dots, \lambda'_M\}$ for a second Poisson process $N\prime(t)$ with rate $\lambda'(t) = \lambda'_{i=X_t}$. Let $c'_k \triangleq N'(t_k) - N'(t_{k-1})$ be the number of sessions in I_k. The MMPP now generates two different types of events c_k and c'_k, i.e. the MMPP acts now on a two-dimensional space. The estimation of of $\{X_t\}$ in the two-dimensional case can be done in a similar manner as for the one-dimensional case.

Let d_i be the duration of the flow event t'_i. The total flow duration l_k during the interval I_k is given by

$$l_k \triangleq \sum_{i=1}^{l} d_i \mathbb{1}_{t_{k-1} < t'_i \leq t_k}. \tag{A.4}$$

We propose to relate l_k to c'_k through an exponential distribution:

$$P(l_k) = c'_k \lambda_D \exp(-c'_k \lambda_D l_k) \tag{A.5}$$

where λ_D is the ground rate of the exponential distribution. A graphical model of our proposed model can be found in figure A.1. This model will hopefully capture the piecewise distribution of flow lengths through the parameters $\boldsymbol{\lambda}'$. Furthermore, it will be straightforward to incorporate this model into our Gibbs sampling framework by sampling $\{c'_1, \dots, c'_n\}$ similarly to $\{c_1, \dots, c_n\}$, and then sampling $\{X_t\}$ conditional on $\Big\{\{c'_1, \dots, c'_n\}, \{c'_1, \dots, c'_n\}, \mathbf{Q}, \boldsymbol{\lambda}, \boldsymbol{\lambda}'\Big\}$. This is however just a proposal, and the exact relation of l_k with c_k might be better modelled in another way.

B. Additional plots

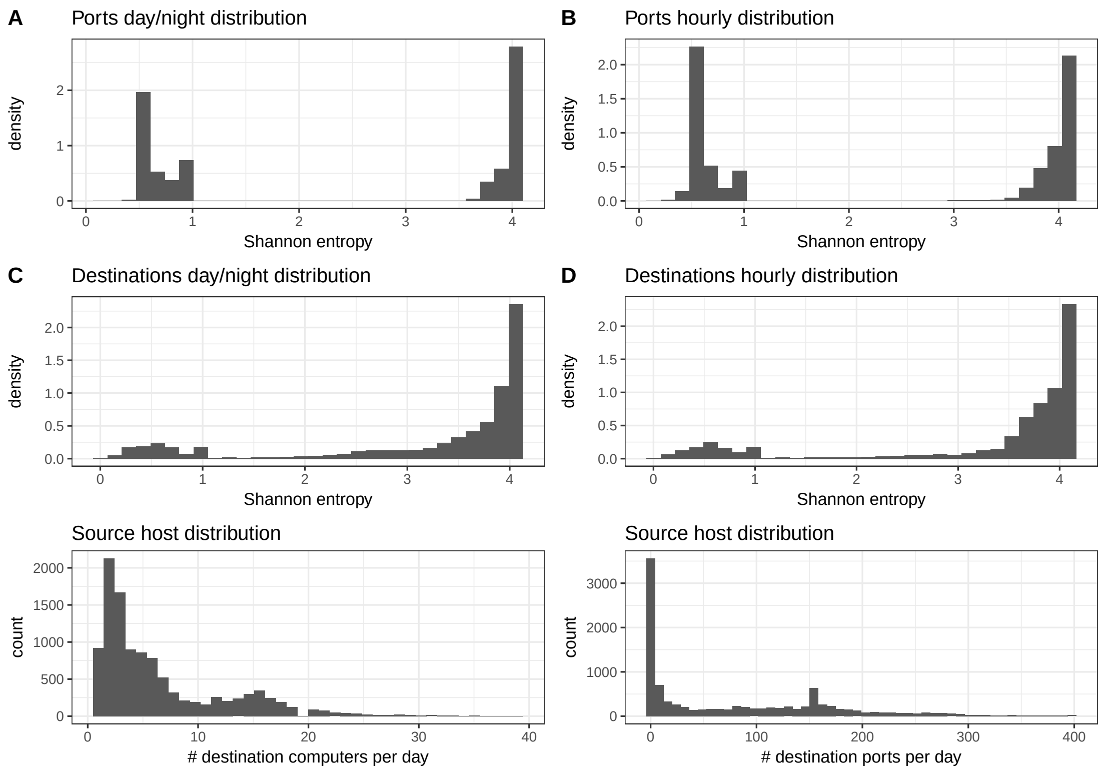

Figure B.1.: Entropies of the hourly and day/night distribution of source hosts in the network, and marginal distributions depicting the number of contacted destinations and ports.

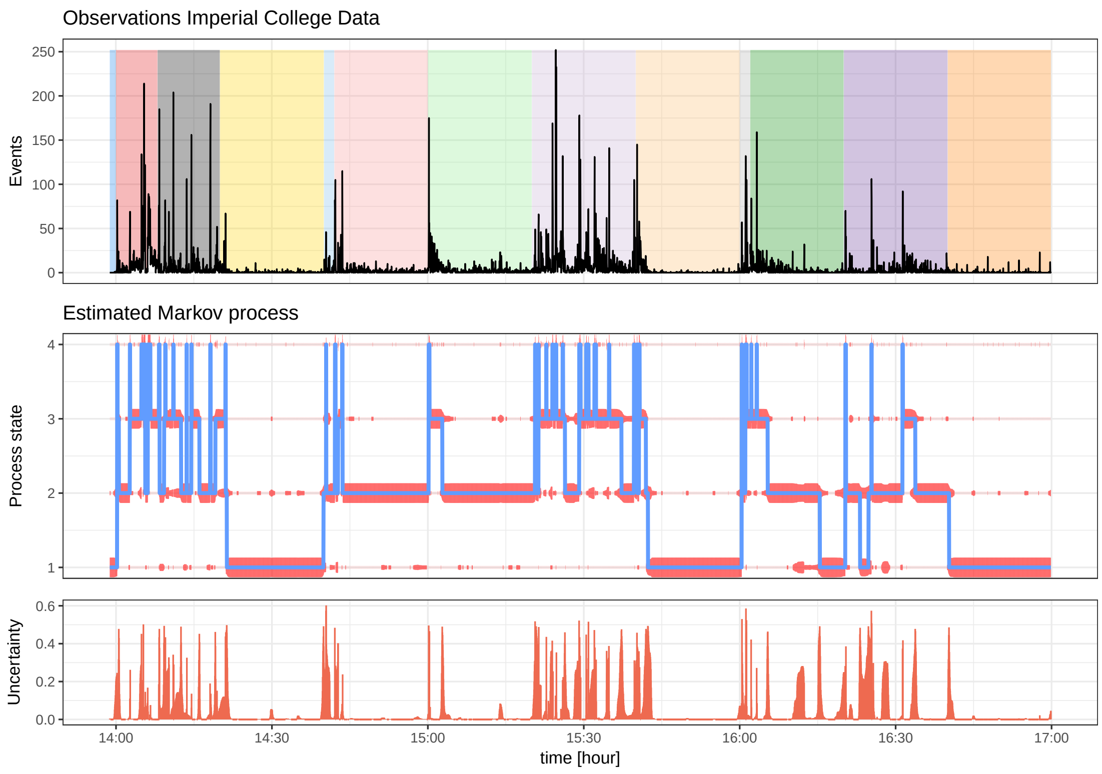

Figure B.2.: Total trajectory of sampled $\{X_t\}$ for the Imperial College data for a 4 state model.

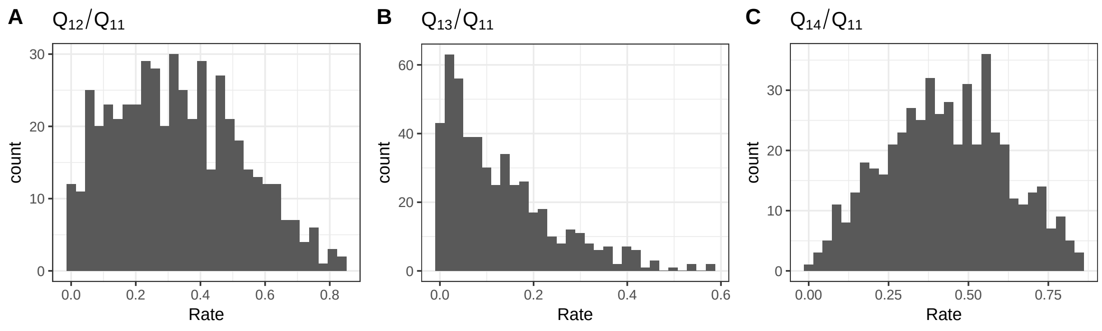

Figure B.3.: Posterior distribution for relative state change probabilities. Since there are in total 12 parameters, we choose to only plot the relative state changes starting from state 1.

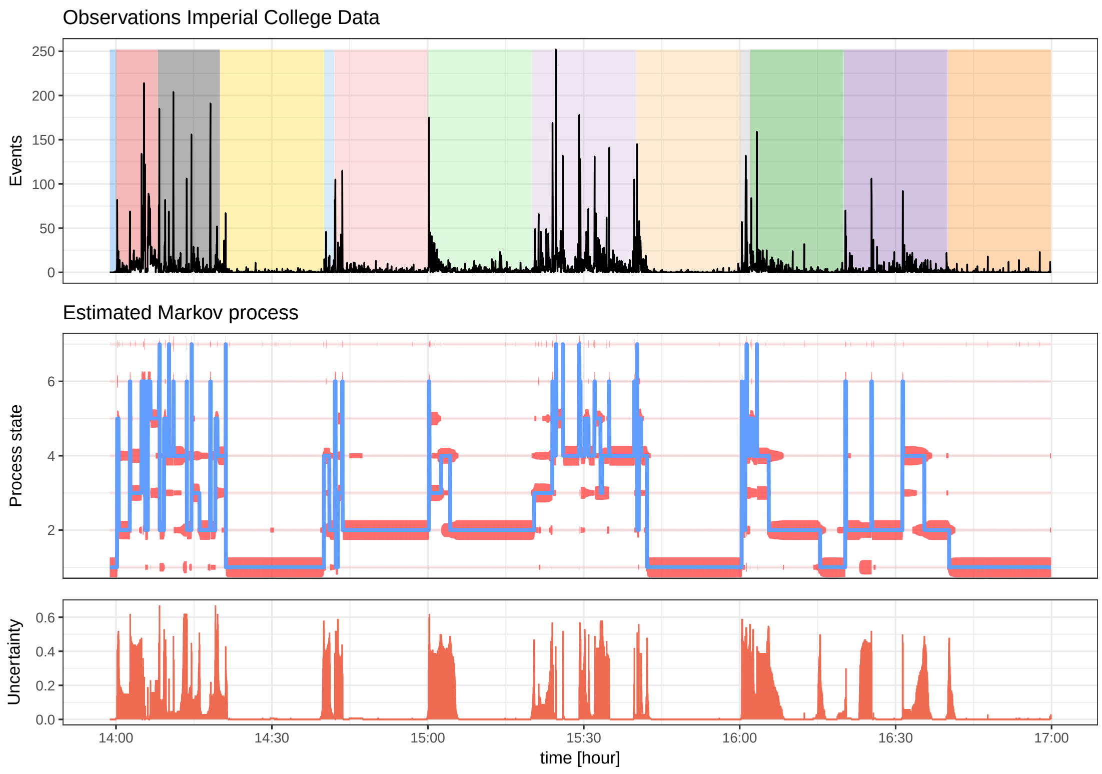

Figure B.4.: Total trajectory of sampled $\{X_t\}$ for the Imperial College data for a 7 state model.

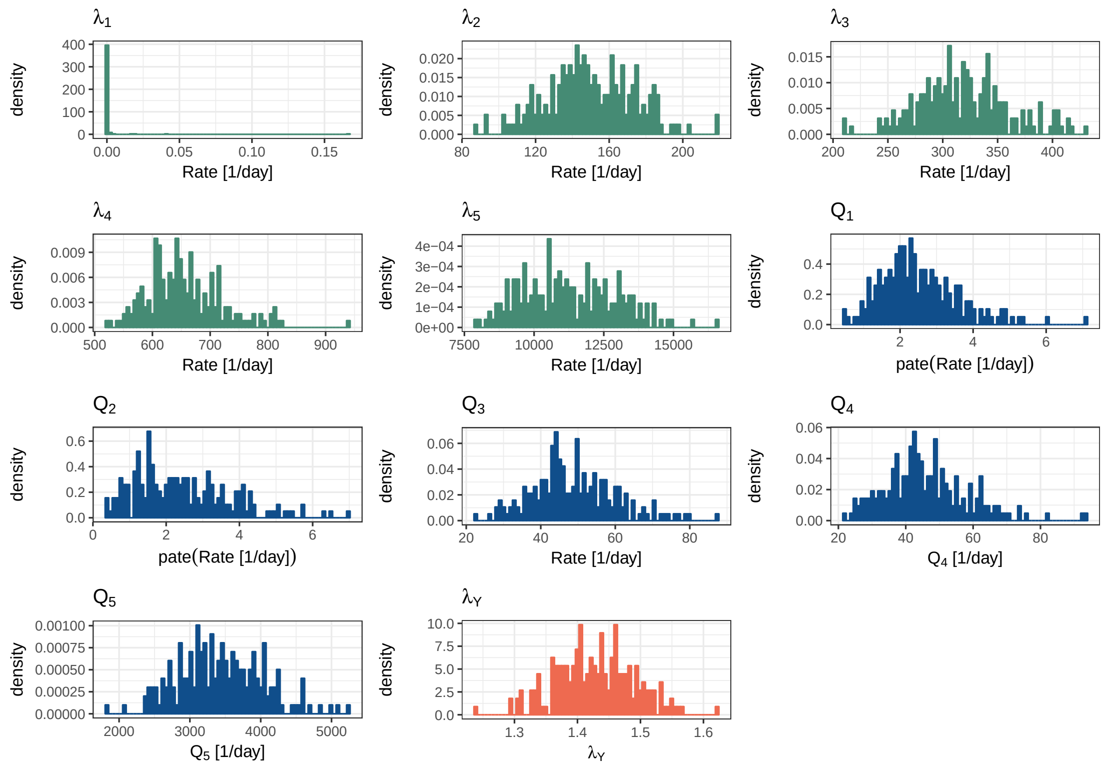

Figure B.5.: Sampled posterior distributions of Q_i, λ_i and λ_Y for a computer in the LANL network (corresponding to figure 4.8).

C. Plots for additional LANL computers

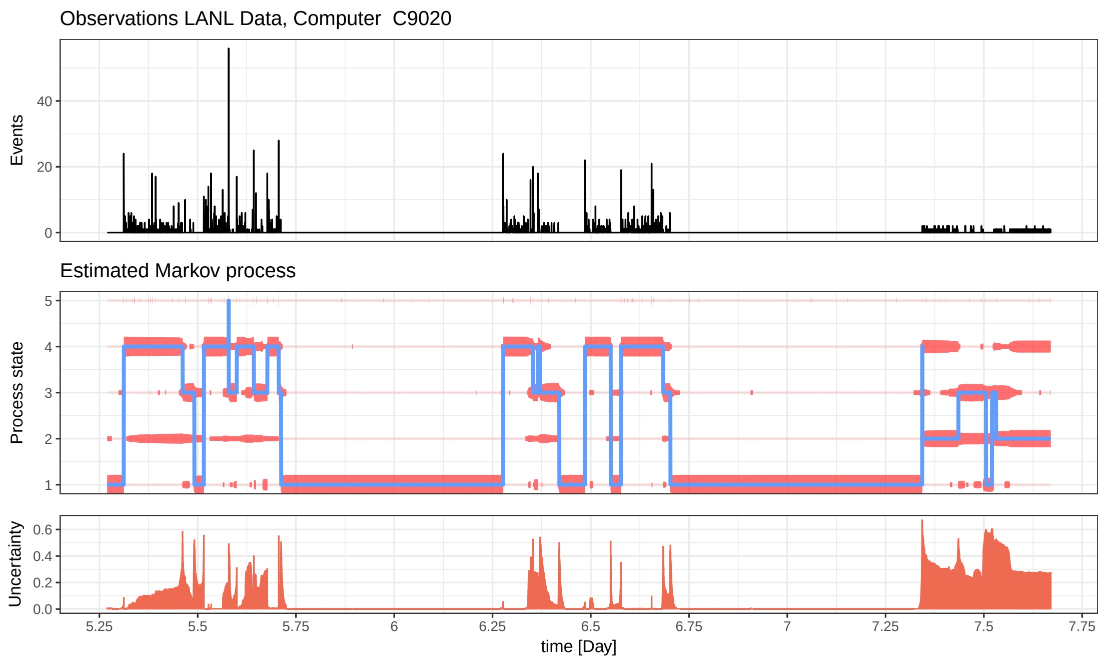

Figure C.1.: Excerpt of sampled $\{X_t\}$ for a selected computer in the LANL data.

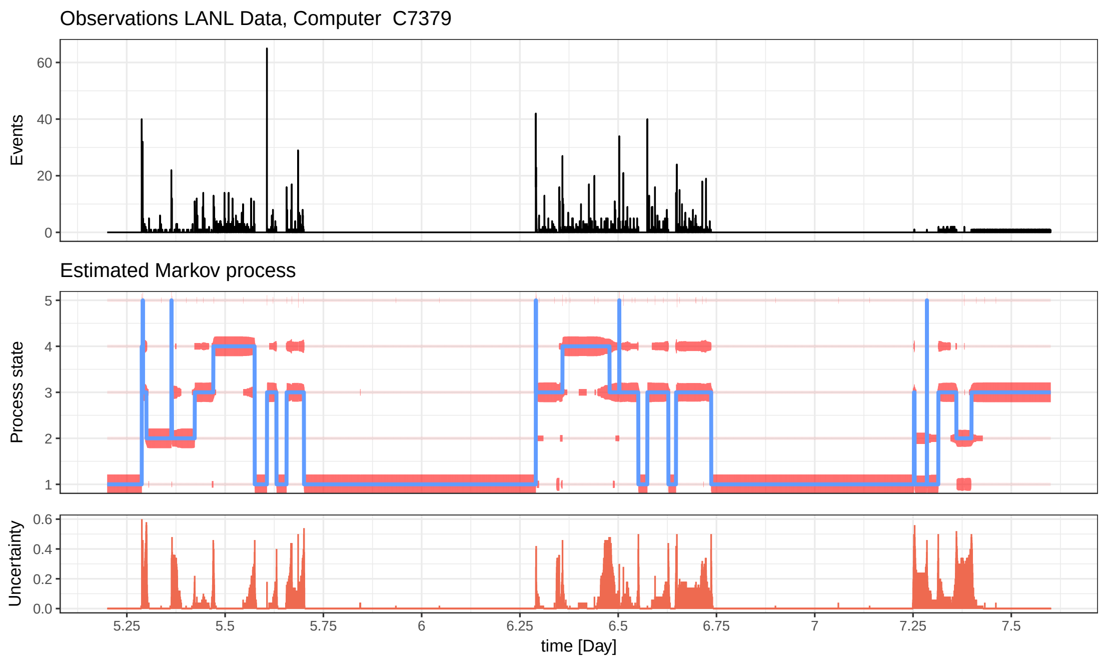

Figure C.2.: Excerpt of sampled $\{X_t\}$ for a selected computer in the LANL data.

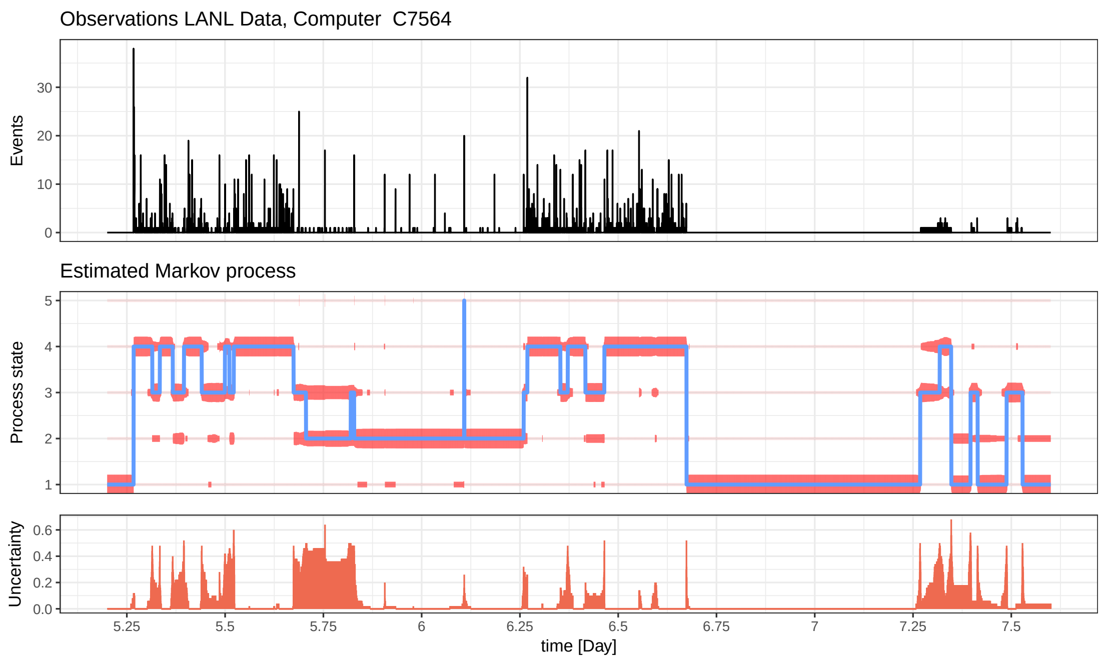

Figure C.3.: Excerpt of sampled $\{X_t\}$ for a selected computer in the LANL data.

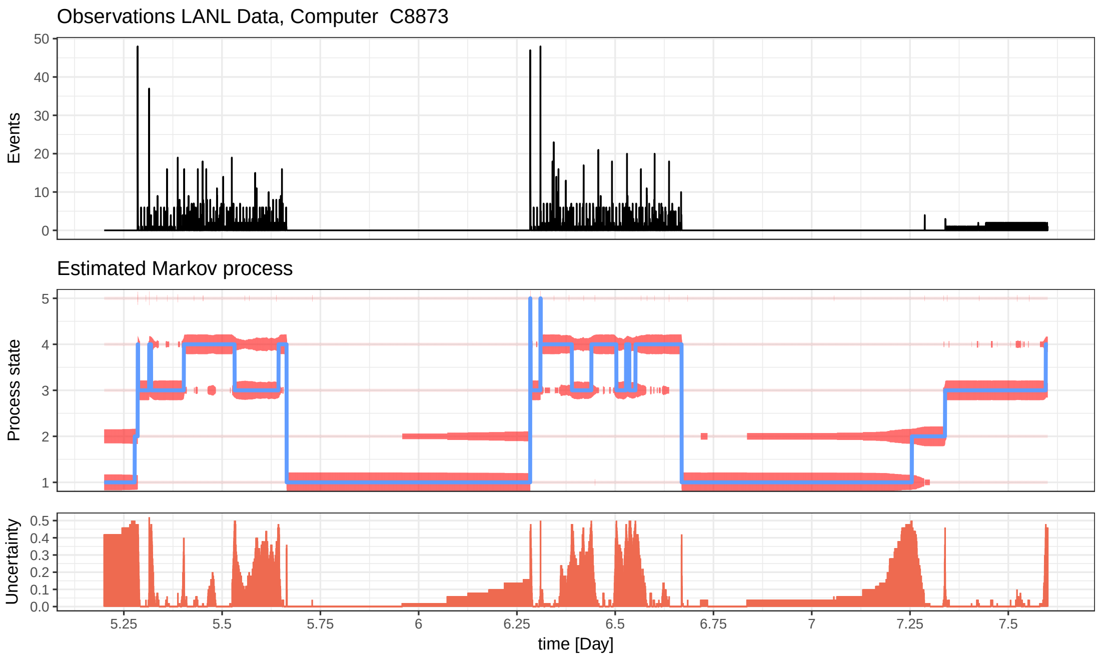

Figure C.4.: Excerpt of sampled $\{X_t\}$ for a selected computer in the LANL data.

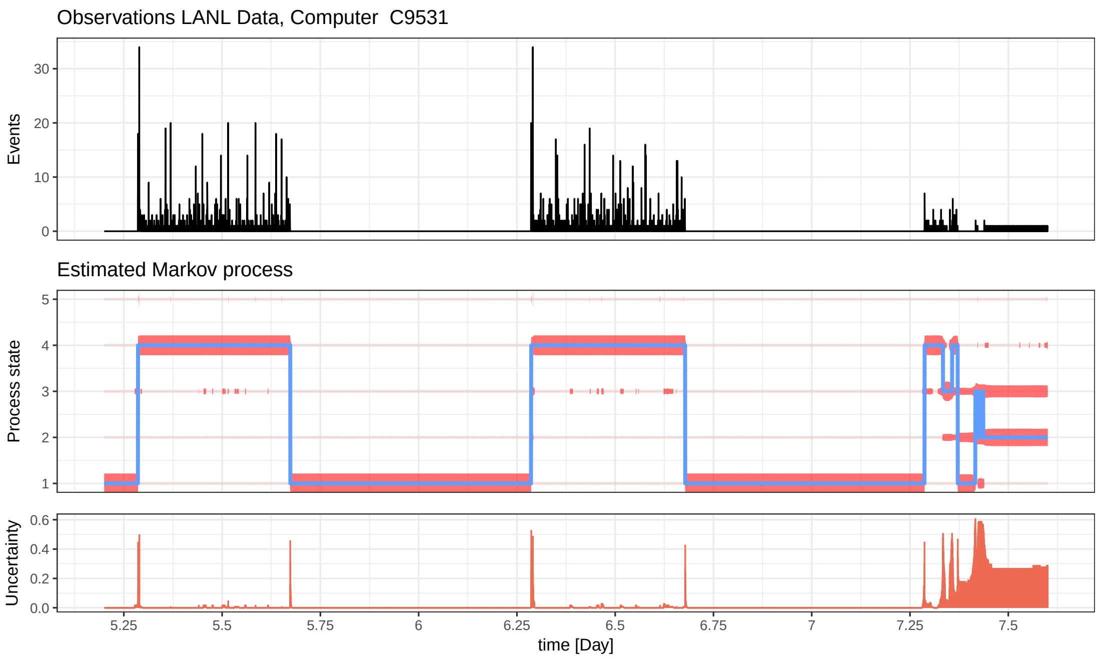

Figure C.5.: Excerpt of sampled $\{X_t\}$ for a selected computer in the LANL data.

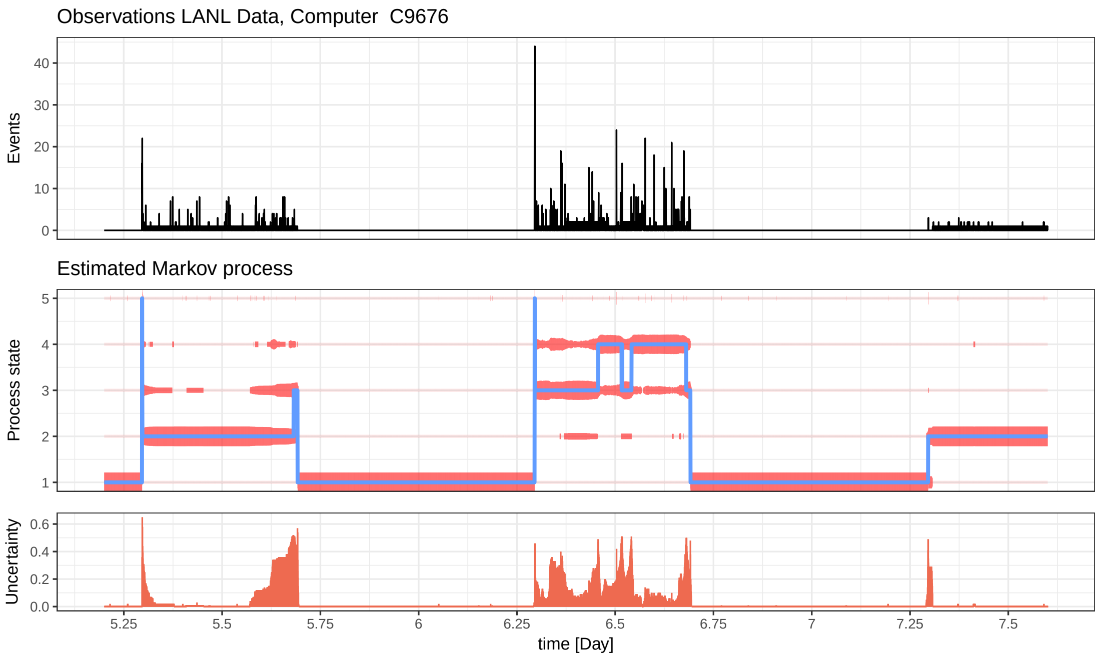

Figure C.6.: Excerpt of sampled $\{X_t\}$ for a selected computer in the LANL data.

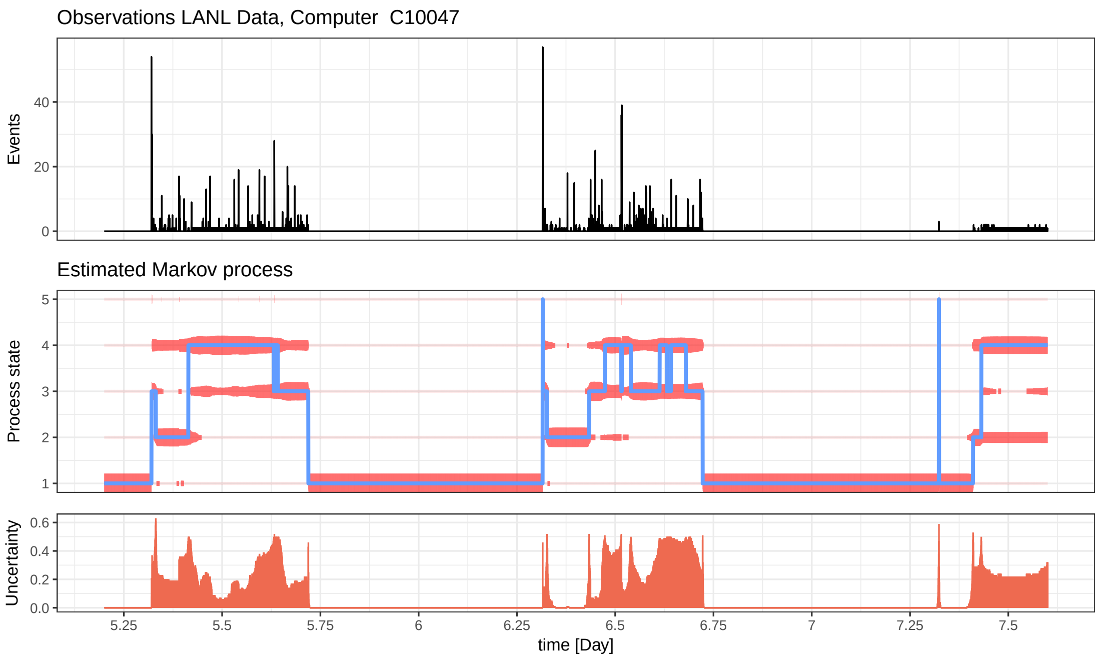

Figure C.7.: Excerpt of sampled $\{X_t\}$ for a selected computer in the LANL data.

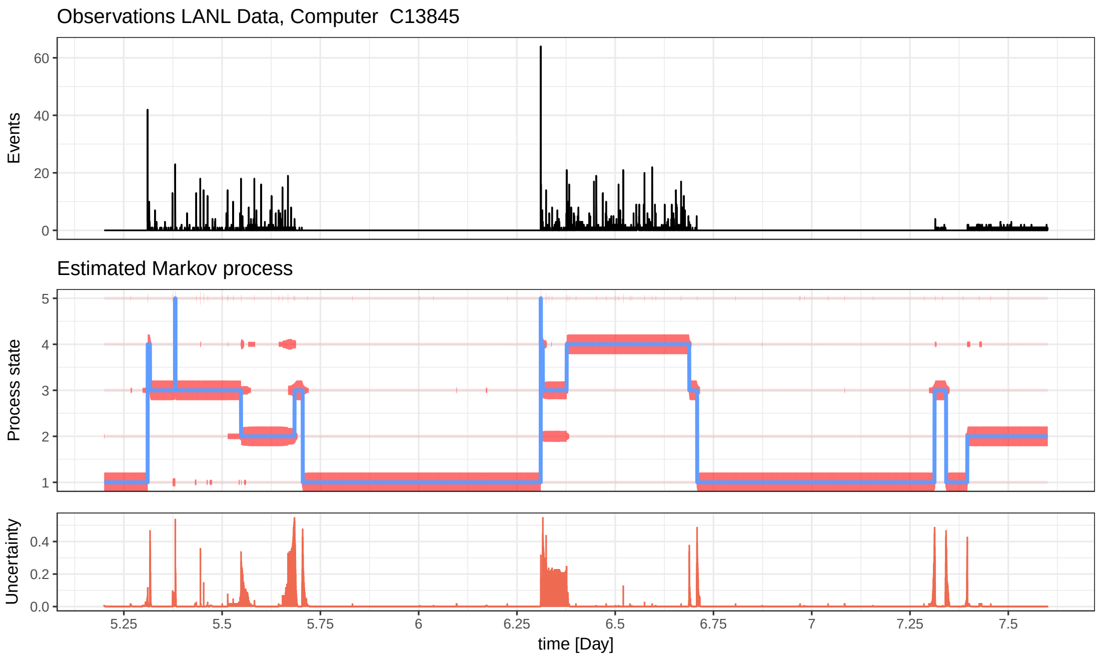

Figure C.8.: Excerpt of sampled $\{X_t\}$ for a selected computer in the LANL data.

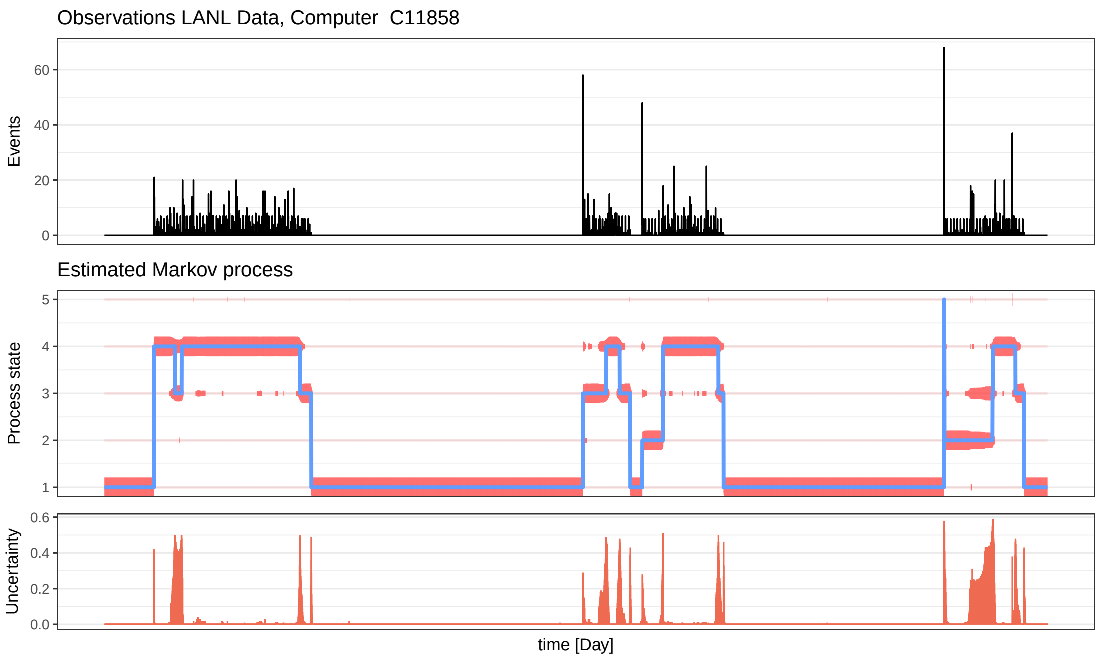

Figure C.9.: Excerpt of sampled $\{X_t\}$ for a selected computer in the LANL data.