KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in the Field of Technology Computer Science and Engineering and the Main Field of Study Probability Theory and Statistics

Second cycle, 30 credits

Emulation and Stochastic Modeling of Client Populations in IT Infrastructures

ARVID LAGERQVIST

Stockholm, Sweden, 2023

Emulation and Stochastic Modeling of Client Populations in IT Infrastructures

ARVID LAGERQVIST

Master's Programme, Applied and Computational Mathematics, 120 credits
Date: August 9, 2023

Supervisors: Kim Hammar, Mattias Sandberg
Examiner: Elias Jarlebring
School of Engineering Sciences
Swedish title: Emulation av klientpopulationer för IT-infrastrukturer
Swedish subtitle: För användning i ”Digital Twins”-teknologi

Abstract

This Master thesis studies stochastic modeling and emulation of client populations in IT infrastructures. We extend a novel emulation system for creating high-fidelity digital twins of IT infrastructures to allow emulation of large and heterogeneous client populations. In our approach, we model client arrivals with an exponential-polynomial-trignometric rate function and model client interactions with Markov chains. We show that these models are scalable and allow theoretical insight into client behavior and evolution, which can guide system development. We further analyze how model parameters can be estimated directly from system measurements, reducing the need for domain experts. To validate our models we implement them on a testbed. Our results show that our models can generate a diverse set of client populations and that this capability can be used to drive computational algorithms based on decision theory and statistical learning to optimize system performance.

Sammanfattning

Nyckelord

Digitala tvillingar, Stokastiska processer, Markovkedjor, Tidsvarierande Poissonprocesser

Acknowledgments

I would like to thank Kim Hammar for having guided me through the project and provided extremely useful and actionable feedback throughout the process. I would also like to thank Mattias Sandberg for his guidance and illuminating discussions about the project and the direction of it, a very useful second opinion from someone that wasn't so tightly coupled to the project.

Stockholm, August 2023
Arvid Lagerqvist

Contents

List of Figures

List of Tables

The list of acronyms and abbreviations should be in alphabetical order based on the spelling of the acronym or abbreviation.

Chapter 1

Introduction

This thesis investigates mathematical models of user behavior in networked systems and how such models can guide system development, focusing on an intrusion response use case. In this introductory chapter, We motivate the research, introduce the area under study, and provide a roadmap for the remainder of the thesis.

1.1 Motivation

In the past few years, virtualization technologies have matured to the point that it is now feasible to deploy large virtual IT infrastructures on commodity hardware. Virtual infrastructures differ from physical ones in that they consist of lightweight virtual containers or virtual machines that enable a higher level of control by shifting functions from hardware to software. Building on this capability, *digital twin* has emerged as a key technology in system automation [1]. A digital twin is a virtual replica of a real-world system that provides a controlled environment for virtual operations, the outcomes of which can be used to optimize operations in the real-world system.

A promising application of digital twins is to optimize control policies for IT infrastructures [2, 3, 4, 5, 6, 7, 8] (see Fig. 1.1). In this line of research, data collected from digital twins is used to drive computational algorithms based on decision theory and statistical learning for computing effective control policies. Compared to purely analytical models and evaluation in production, digital twins provides three key benefits for this use case: (*i*) it provides a safe and realistic test environment; (*ii*) it provides evaluative feedback that enables closed-loop learning of policies; and (*iii*), it allows collecting data and evaluating policies without affecting operational workflows on the real-world

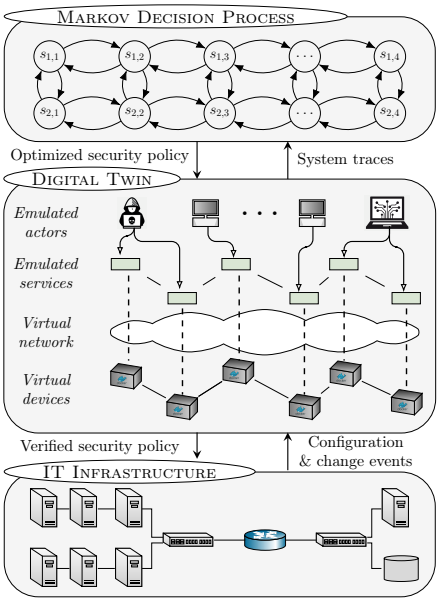

Figure 1.1: This thesis investigates analytical modeling and emulation of client behavior in IT infrastructures; our formal models and emulator can be used as part of a digital twin to find and evaluate security policies for an IT infrastructure; the digital twin is a virtual replica of the IT infrastructure and is used to evaluate security policies and collect data; the collected data is used to instantiate simulations of Markov decision processes and to learn effective policies through reinforcement learning [2, 3, 4, 5, 6, 7, 8].

infrastructure.

A major challenge in building digital twins is to capture the complex dynamics of real-world IT infrastructures. In general, creating a digital twin of an IT infrastructure involves six emulation functions: (1) functions for emulating physical resources (e.g. CPU, storage, and memory); (2) functions for emulating network communication (e.g. IP networks); (3) functions for emulating network conditions (e.g. packet loss probabilities, jitter, and bit rates); (4) emulating system operations (e.g. security operations); (5) emulating cyber attacks (e.g. remote-code execution attacks); and (6), emulating client populations (e.g. user behavior and interaction with the system).

This thesis studies the latter task: emulation of client populations. The main challenge in emulating client populations is the hetereogenity of typical user behavior and the complexity of their interactions with computer systems. To deal with this complexity, we use *analytical modeling* and

stochastic processes to obtain insight and guide the development of an emulator for client populations. Specifically, we use time-varying Poisson processes in conjuction with Markov chains to model client behavior on an IT infrastructure. To evaluate our approach, we implement a novel emulator that generates real client requests on real or virtual systems.

The implemented emulator extends the software framework developed in [2, 3, 4, 5, 6, 7, 8], which allows emulation of large-scale IT infrastructures in a virtual environment.

1.2 Use Case: Intrusion Response

To evaluate the formal models and implementations developed throughout this thesis, we focus on the following use case.

We consider an intrusion response scenario that involves the IT infrastructure of an organization (see Figure 1.2). The operator of this infrastructure, which we call the defender, takes measures to protect it against an attacker while providing services to a client population. The infrastructure includes a set of servers that run the services and an Intrusion Detection and Prevention System (IDPS) that logs events in real-time. Clients access the services through a public gateway, which is also open to the attacker.

The attacker's goal is to intrude on the infrastructure and compromise its servers. To achieve this, the attacker explores the infrastructure through reconnaissance and exploits vulnerabilities while avoiding detection by the defender. The attacker follows a pre-defined attack policy, which is defined in [6].

The defender continuously monitors the infrastructure through accessing and analyzing IDPS alerts and other statistics. It can take a fixed number of defensive actions, each of which has a cost and a chance of stopping an ongoing attack. An example of a defensive action is to drop network traffic that triggers IDPS alerts of a certain priority. The defender takes defensive actions in a pre-determined order, starting with the action that has the lowest cost. The final action blocks all external access to the gateway, which disrupts any intrusion as well as the services to the clients.

When deciding the time for taking a defensive action, the defender balances two objectives: (*i*) maintain services to its clients; and (*ii*), stop a possible intrusion at the lowest cost. The optimal policy for the defender is to monitor the infrastructure and maintain services until the moment when the attacker enters through the gateway, at which time the attack must be stopped

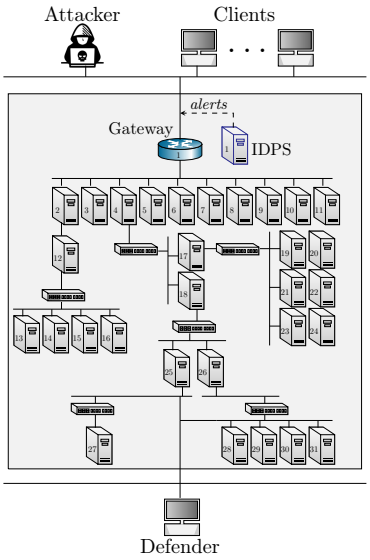

Figure 1.2: The IT infrastructure and the actors in the intrusion response use case that is used for evaluation in this thesis [2, 3, 4, 5, 6, 7, 8].

at minimal cost through defensive actions. The challenge for the defender is to identify this precise moment.

1.3 Research Questions

The main research question of this thesis, as well as a series of supplementary questions, are presented below.

- How can one model heterogeneous client populations for IT infrastructures?
  - How can the model be made flexible?
    - It must be able to model many different client types.
    - It must able to model many different infrastructures.
  - How can the model be made easily interpretable?
    - The parameter count must be held as low as possible.
    - The different parts of the model must have clear analogs to real world phenomena.
  - How can the usability of the model be best optimized, considering the specific software implementation as well as the model design?

- How can the model be made scalable, i.e. able to model client populations and infrastructures of arbitrary size?
- How can the model be made realistic, is it able to model the relevant real world phenomena?
- Is the model possible to implement and be used in a real simulation system?

1.4 Related work

There is a lot of research regarding how to generate realistic web traffic load for performance evaluation of computer systems [9, 10, 11, 12]. Both [9] and [10] are studies of a specific traffic generating system, however their evaluation is done from different perspectives. [9] evaluates the traffic generation of the system from a performance perspective while [10] evaluates from a cybersecurity perspective, specifically based on how well signature-based intrusion detection systems (SIDS) are tested using the generator. There is comparatively little research where the evaluation criteria are related to cybersecurity as opposed to performance evaluations. The question of how to generate realistic client traffic load has been tackled in a few different ways that can be characterized in to two different categories, analytical models and data/trace driven models. All models exhibits both analytical and data driven characteristics. Most models are analytical in some way since even if they are mostly data driven they often require some choice of what parameters are estimated using the data and what type of generating model to use. A purely data driven model might just replay traces collected from a system such as ”Monkey see, monkey do” [13], the weakness of such a model is that it likely generalises very poorly to unseen scenarios. [14] is an almost entirely analytical article that focuses on one aspect of web traffic that is also noted in [9], namely that it is often self-similar. One way of modeling self similarity for traffic load is using fractional Brownian motion, which is done in [14]. The self-similarity property of network traffic is especially important for performance evaluation according to [9]. There have also been more data driven studies of traffic load generation in the context of cybersecurity [15, 10]. One example of this is [15], the focus in this study however was to create realistic workloads that reflected normal user behaviour and not necessarily to create challenging scenarios for the cybersecurity testbed.

The main references for this thesis will be the work of Papadopouli et. al. [16, 17, 18] where client arrivals, association patterns and connection

duration on a campus network was modeled using time varying Poisson processes, Markov chains and Bipareto distributions for the service times. The purpose of the modeling in those studies was to enable optimization of the networking infrastructure based on the access patterns of the users, in this thesis the purpose will be to create a model that generates realistic and, from a cybersecurity perspective, challenging client load and behaviour. In this thesis a model will be presented that has the capacity to adapt to provided client traffic data or pre-programmed behaviour.

In this thesis a Markov model based on Markov chains will be used, similar models are referred to as Variable order Markov models (VMMs). VMMs are used in many areas, among them are machine learning, lossless compression, and dna sequence prediction. [19, 20, 21]

This thesis provides an investigation of how one can model and emulate client behaviour for creating digital twins of IT infrastructures. Such digital twins are used for system automation and optimizing operations in IT infrastructures, for example optimizing security strategies [2, 3, 4, 5, 6, 7, 8]. In [4] reinforcement learning is used to train an agent to defend against intrusions and since the training environment directly affects the objective function it could be useful to understand the environment better and have more control over its parameters.

1.5 Outline

In the next section (Section 1.4) research about similar topics is discussed, such as performance evaluation, load generation, and Markov models. The next chapter (Chapter describes the necessary mathematical concepts that are used in the model. Section 2.1 presents the time-varying Poisson process and how to simulate it. Section 2.2 describes Markov chains and some useful properties of them that will be used in later sections. Chapter 3 gives the model of the clients in its entirety, including both the arrival processes and the service chains. In sections 3.1 and 3.3 the mathematical model is presented and in Section 3.4 a small example scenario is described and implemented. The last three sections of the chapter treats how to calculate expected network load (Section 3.5), the space complexity of the algorithm (Section 3.6) and a summary of the chapter (Section 3.7). The fourth and final chapter evaluates the work of the thesis and puts it into a larger context. Section 4.1 is a description of the testbed used to run the model for evaluation. The following sections present the results of the practical evaluation and discuss them as well as draws conclusions from them.

Chapter 2

Mathematical background

This chapter presents the mathematical concepts that are used to create the model. It also describes some further calculations that can be done with the mathematical building blocks of the model, that are applied to the model in later sections.

2.1 Time-varying Poisson processes

A time-varying Poisson process is a Poisson process where the Poisson parameter depends on the time parameter t i.e. $Po(\lambda(t))$ where $\lambda(t)$ is any continuous function of t. A time-varying Poisson process can also be called a non-homogeneous Poisson process. The non-homogeneous Poisson process $Po(\lambda(t))$ has rate $\lambda(t)$ at time t. To generate a time-varying Poisson process with rate function $\lambda(t)$ one can use a process called *thinning* which was presented in [22]. The basic idea is to use a homogeneous (constant rate) Poisson process $Po(\lambda^*)$ whose rate function is greater than $\lambda(t)$ at all times t and *thin* the output of this process by removing points generated by $Po(\lambda^*)$ with probability $1 - \frac{\lambda(t)}{\lambda^*}$. The full algorithm and proof is detailed in [22]. Other simulation methods as well as several advanced models for time-varying Poisson processes are presented in [23]. The appeal of the thinning process for simulation of the time-varying Poisson process lies in its simplicity and ease of implementation, however [23] discusses various efficiency gains that can be made by making further assumptions about the rate function.

2.1.1 The rate function and maximum likelihood estimation

One of the more simple choices for the form of the rate function $\lambda(t)$ would be to have it be an exponential function, i.e. of the form $\lambda(t) = e^{\theta_0+\theta_1 t}$. This leads to a model that has nice statistical properties and leads to simpler calculations than other methods. However, this model requires a monotone increasing or decreasing function which is does not entirely fit the use case of this thesis. To incorporate periodicity as well as giving room for more global patterns one can use the EPTMP model which was presented in [24]. EPTMP stands for Exponential-Polynomial-Trigonometric rate function having Multiple Periodicities and it has the form

$$\lambda(t) = exp\left(\sum_{i=0}^{m} \theta_i t^i + \sum_{k=1}^{p} \gamma_k sin(\omega_k t + \phi_k)\right) = exp(h(t; m, p, \boldsymbol{\theta})) \quad (2.1)$$

where the parameters to estimate are

$$\boldsymbol{\theta} = [\theta_0, \theta_1, ..., \theta_m, \gamma_1, ..., \gamma_p, \phi_1, ..., \phi_p, \omega_1, ..., \omega_p]$$

As stated in [23] these parameters can be interpreted as follows: $\theta_0, ..., \theta_m$ represent the overall trend in frequency of events over a long time frame. The periodic part of the function is divided into three parts: $\gamma_1, ..., \gamma_p$ are amplitudes, $\phi_1, ..., \phi_p$ are period shifts, and $\omega_1, ..., \omega_p$ are frequencies. To perform maximum likelihood estimation the *log-likelihood function* must be used. The log-likelihood function for the rate function (2.1) given that one has a realisation of the process $Po(\lambda(t))$ on the interval $(0, T]$ with arrival times $t_1 < t_2 < ... < t_n$ is

$$\ell(\mathbf{t}, n|\boldsymbol{\theta}) = \sum_{i=0}^{m} \theta_i \sum_{j=1}^{n} t_j^i + \sum_{k=1}^{p} \sum_{j=1}^{n} \gamma_k sin(\omega_k t_j + \phi_k) - \int_0^T exp(h(u; m, p, \boldsymbol{\theta}) du$$

The estimated parameters are then

$$\hat{\boldsymbol{\theta}}_{ML} = \arg\max_{\boldsymbol{\theta}} \ \ell(\mathbf{t}, n|\boldsymbol{\theta})$$

2.1.1.1 Using maximum likelihood estimation from sampled data

This section is based on Section 3.1.1 in [25]. Given sampled data $x = (x_1, x_2, ..., x_m), m \geq 1$

2.1.2 Initial estimates of the parameters

As suggested in [23] one can use a periodogram of the data to get initial estimates for frequencies of the periodic part of the rate function.

2.2 Markov chains

Consider a set of states $0, 1, 2, ..., M$. A Markov chain is a special case of a stochastic process with outcomes in this set of states. Consider a stochastic process $\{X_t\}$. $\{X_t\}$ is considered a first order Markov chain if the following property holds

$$P(X_{t+1} = j | X_0 = l_0, X_1 = l_1, ..., X_{t-1} = l_{t-1}, X_t = i) = P(X_{t+1} = j | X_t = i)$$

for $t = 0, 1, 2, ...$ and every sequence $i, j, l_0, l_1, ..., l_{t-1}$. Meaning that the next state only depends on the state before it. The probabilities of transitioning from one state to another can be described using a *transition matrix*. The elements of the transition matrix $\mathbf{P}$ are defined as

$$\mathbf{P}_{i,j} = P(X_{t+1} = j | X_t = i)$$

where the element on the i:th row and j:th column describe the probability of transitioning from state i to state j. A Markov chain of first order can be defined using the state space and transition matrix as follows

Definition 1. A Markov chain $w = \langle \mathbf{S}, \mathbf{P} \rangle$ consists of the set of states $\mathbf{S} = \{1, 2, ..., M\}$ and the transition matrix $\mathbf{P}$. Where $\mathbf{P}_{i,j}$ denotes the probability that the next state of the Markov chain is $j \in \mathbf{S}$ given that the current state is $i \in \mathbf{S}$.

A second order Markov chain also considers the previous state in addition to the current state when determining the probabilities for the next state. Meaning that $\{X_t\}$ is a second order Markov chain if

$$P(X_{t+1} = k | X_0 = l_0, x_1 = l_1, ..., X_{t-2} = l_{t-2}, X_{t-1} = j, X_t = i) = \\ P(X_{t+1} = k | X_t = i, X_{t-1} = j)$$

for $t = 0, 1, 2, ...$ and every sequence $i, j, k, l_0, l_1, ..., l_{t-1}$. The transition probabilities for a second order Markov chain can be described using the three-dimensional transition matrix $\mathbf{P}^{(2)}$ where

$$\mathbf{P}^{(2)}_{i,j,k} = P(X_{t+1} = k | X_t = i, X_{t-1} = j)$$

meaning that $\mathbf{P}^{(2)}_{i,j,k}$ is the probability that we end up in state k if the previous state was j and the current state is i.

This pattern can be extended up to any order n yielding an n:th-order Markov chain [26, 16]. When the term Markov chains is used in the rest of this thesis, this will be in referrence to Markov chains of first order.

2.2.1 Converting higher order Markov chains to first order

A Markov chain of order k can be converted to a first order Markov by a simple procedure. To illustrate the method we use the example of a second order Markov chain with the state space $\{a, b, c\}$. The possible previous states for the transition matrix $\mathbf{P}^{(2)}$ are (a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, a), (c, b), (c, c). We can encode all of these combinations in the names of the states of a new Markov chain which has the state space $\{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, a), (c, b), (c, c)\}$, this Markov chain is of first order. [27]

2.2.2 Hitting times for Markov chains

The following definition and explanation of hitting times is adapted from [28]. The hitting time of a subset of states $A \subset \mathbf{S}$ for a Markov chain is defined as the minimum number of time steps it takes to reach a state in the subset A. Formally the hitting time is modeled as a random variable

$$H_A = \min\{n \in \{0, 1, 2, ...\} | X_n \in A\}.$$

In this thesis we will only handle the case when A is a single state i.e.

$$H_j = \min\{n \in \{0, 1, 2, ...\} | X_n = j\}. \tag{2.2}$$

The expected hitting time is defined as follows

Definition 2. The expected hitting time for state j when starting in state i is defined as

$$\eta_{i,j} = \mathbb{E}[H_j | X_0 = i]$$

To illustrate the method of calculating the expected hitting time we use an example from [28]. Consider a Markov chain with the transition matrix

$$\mathbf{P} = \begin{bmatrix} \frac{1}{4} & \frac{3}{4} & 0 \\ \frac{1}{4} & 0 & \frac{3}{4} \\ 0 & \frac{1}{4} & \frac{3}{4} \end{bmatrix}$$

Let suppose we want to find $\eta_{1,3}$ for this Markov chain. We start by setting up an equation for $\eta_{1,3}$

$$\eta_{1,3} = 1 + \mathbf{P}_{1,1} \cdot \eta_{1,3} + \mathbf{P}_{1,2} \cdot \eta_{2,3}. \tag{2.3}$$

Where $\eta_{1,2}$ is composed of one unit of time for the current time step and a weighted sum of the expected hitting times for the possible subsequent states. A natural continuation would be to set up an equation for the unknown $\eta_{2,3}$ next

$$\eta_{2,3} = 1 + \mathbf{P}_{2,1} \cdot \eta_{1,3} + \mathbf{P}_{2,3} \cdot \eta_{3,3}. \tag{2.4}$$

Combining equations (2.3) and (2.4) together with the fact that $\eta_{3,3} = 0$ we get the following system of equations

$$\begin{cases} \frac{3}{4}\eta_{1,3} - \frac{3}{4}\eta_{2,3} = 1 \\ -\frac{1}{4}\eta_{1,3} + \eta_{2,3} = 1. \end{cases}$$

When solved this gives us the value $\eta_{1,3} = \frac{28}{9} \approx 3.11$.

In general this method can be stated like this. Given a Markov chain $w = \langle \mathbf{S}, \mathbf{P} \rangle$ that has exactly one recurrent state $\varnothing$, which is also absorbing. Where $\mathbf{P} \in \mathbb{R}^{n \times n}$. Let $\eta_{i,\varnothing}$ be the expected hitting time for the absorbing state given that the process starts in state i. Then $\eta_{i,\varnothing}$ can be calculated as follows

$$\eta_{i,\varnothing} = \sum_{j=1}^{n} \mathbf{P}_{i,j} \cdot \eta_{j,\varnothing}. \tag{2.5}$$

This recursive formula together with the base case that $\eta_{\varnothing,\varnothing} = 0$ gives the expected time before the Markov chain enters the absorbing state.

2.2.3 Expected number of visits

The following is presented in [29] but is reiterated here for clarity. To calculate the expected number of visits for a given state in a Markov chain we first need to define the concepts of transient and recurrent states. Recurrent states are states that, if the stochastic process described by the Markov chain continued

for infinite time, would recur an infinite amount of times. The expected number of visits for recurrent states is therefore always infinity. Transient states on the other hand are states that only occur a finite amount of times, even if the process continues for infinite time. Given a first order Markov chain on the state space $\mathbf{S} = \{S_1, S_2, ..., S_n\}$ we define $T \subset \mathbf{S}$ as the set of transient states. We give the transient states new labels from 1 to b where b is the number of transient states. Let $P_T = (P_{i,j})\ i, j \in T$ be the transition matrix for the transient states. Let $V_{i,j}$ be the expected number of visits to S_j given $X_0 = S_i \in T$ then

$$V_{i,j} = \mathbb{E}\left[\sum_{n=0}^{\infty} \mathbb{I}_{\{X_n = j | X_0 = i\}}\right] = \sum_{n=0}^{\infty} P_{i,j}^n.$$

Where P^n is the n-step transition matrix for the Markov chain. Now given this [29] presents the following proposition and proof, which are given here in the context and notation of this thesis.

Proposition 1. Let I denote the $b \times b$ identity matrix. Then $V = I + P_T V$ which in turn yields

$$V = (I - P_T)^{-1} \tag{2.6}$$

Proof. Assume $X_0 = i \in T$. In the case $j = i$ since we, from the beginning, have an initial visit to j it follows that

$$V_{i,i} = 1 + \sum_{k \in T} P_{i,k} V_{k,i}.$$

In the case where $j \neq i$ we get the a similar relation but without the initial visit

$$V_{i,j} = \sum_{k \in T} P_{i,k} V_{k,j}.$$

These sums of products can be written in matrix form as $V = I + P_T V$ which gives $(I - P_T)V = I$. And since in general for square matrices A, B it holds that $det(AB) = det(A)det(B)$ we can conclude by letting $A = (I - P_T)$ and $B = V$ that $det((I - P_T)V) = det(I) = 1$ and therefore that both $(I - P_T)$ and V have non-zero determinants which makes them invertible. Using this knowledge we can write the expression as $V = (I - P_T)^{-1}$. □

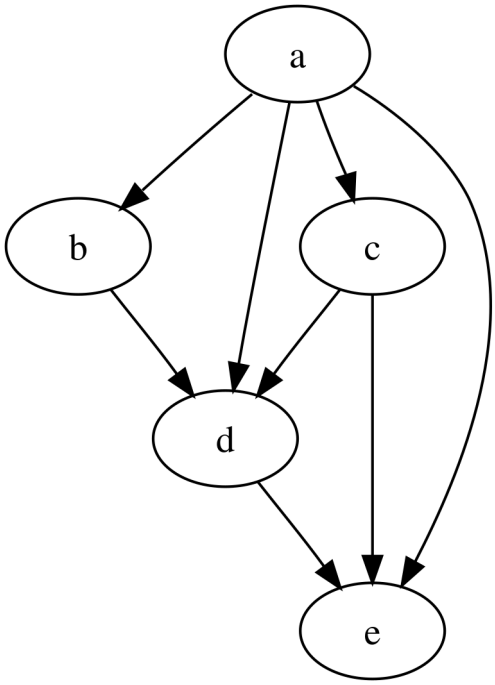

Figure 2.1: An example of a Directed Acyclic Graph

2.3 Directed Acyclic Graphs

Directed acyclic graphs (DAG) are graphs that have directed edges and contain no cycles, an example of a DAG can be seen in Figure 2.1. The graph is defined by two sets, the set of vertices usually denoted by V and the set of edges usually denoted by E. The set of edges is composed of ordered tuples with two vertices each, for example $\{(a, b), (a, c)\}$ indicating that there are edges going from a to b and from a to c. The edges may also be weighted meaning that each edge is also associated with a number that represents the cost or probability of that edge being traversed.

Chapter 3

Modeling client behaviour

The clients behavior will be modeled in two stages: arrival process and service chain generation. At first individual time-varying Poisson processes with a rate function that depends on historical data for a specific client type will generate client arrivals. In the second stage a Markov model will be used to generate service access patterns for the individual client types.

3.1 Network infrastructure

We consider IT infrastructures comprising application servers connected in a communication network. These application servers run services that are consumed by clients through a public gateway (see Figure 3.1). An example of a specific network infrastructure is shown in figures 1.1 and 1.2. In this thesis we will not concern ourselves with the specific implementation of the network, except for in the practical evaluation part (see Chapter 4), and will instead focus on creating a more general model that might be used for many different scenarios that have this general structure. The idea is that the clients that arrive according to some process, that possibly can be modeled as in Section 3.2, to a public gateway and then interact with the system for some time by using the different services and then leaving the network. The series of interactions that a client makes with the system during a session is are modeled in Section 3.2.1. A goal with the model in this thesis is for the parameters of the clients to be easily manipulated and understandable enough so that experiments can be done with different configurations that simulate scenarios that might occur in real systems.

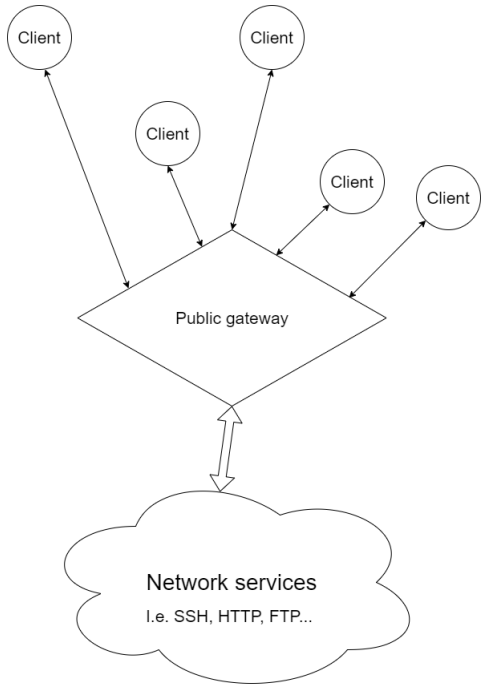

Figure 3.1: A simplified view of the network that will be modeled

3.2 System Model

In the model presented in this thesis there will be clients of different types, the different client types have individual arrival processes and decision processes, i.e. they are completely independent. Each type of client has an associated arrival process $Po(\lambda_{C_i}(t))$ that describes the connection rate of client type C_i. $\lambda_{C_i}(t)$ is an EPTMP rate function as described in Section 2.1.1 with parameters that are unique to the client type C_i.

Once the clients have arrived at the system they will interact with the system through a series of network functions, an example of network functions available in a network can be seen in Table 3.1. The available network functions are represented as a set of states $\mathbf{S} = \{S_1, S_2, ..., S_{N_s}\}$ where N_s is the number of network functions available in the given network. To model the behaviour of the clients we make the assumption that the clients have some predetermined set of workflows that they might execute when connecting to the system. Each one of these workflows may be modeled as a DAG, for an example see Figure 3.4.

Definition 3. A workflow $w = \langle V, E \rangle$ is defined as a directed acyclic graph of finite size where $V \subseteq \mathbf{S}$ and $E \subseteq \mathbf{S} \times \mathbf{S}$.

3.2.1 Markov chain representation of workflows

To model the clients traversal in the workflow we use a Markov chain defined as follows:

A workflow Markov chain $w = \langle \mathbf{S}, \mathbf{P} \rangle$ is defined by the state space $\mathbf{S}$ of states in the workflow and the transition matrix $\mathbf{P}$ which describes the probability of a certain state following another. A workflow markov chain has the following properties:

1. A workflow Markov chain always contains exactly one absorbing state, $\varnothing$.
2. All states other than the one absorbing state are transient.
3. Every edge going to the absorbing state is non-zero, i.e. $\mathbf{P}_{i,\varnothing} > 0, \forall i$.

The following proposition shows that the expected hitting time for the absorbing state is finite for all Markov chains fulfilling the above properties.

Proposition 2. For all Markov chains fulfilling the properties 1-3 above the following holds:

$$\eta_{i,\varnothing} < \infty, \ i = 1, 2, ..., M$$

Proof. Let p_{min} be the lowest exit probability, i.e.

$$p_{min} := \min_{i \in \mathbf{S}} \mathbf{P}_{i,\varnothing}$$

From property 3 we know that $p_{min} > 0$. Every step of the Markov chain can be seen as a Bernoulli trial where $p = \mathbf{P}_{i,\varnothing}$ and since we know that $\forall i \ \mathbf{P}_{i,\varnothing} \geq p_{min}$ we can say the following

$$\mathbb{E}[H_\varnothing] \leq \mathbb{E}[Y]$$

Where $Y \in Ge(p_{min})$, and since $\mathbb{E}[Y] = \frac{1}{p_{min}}$ we see that $\mathbb{E}[H_\varnothing] \leq \mathbb{E}[Y] = \frac{1}{p_{min}} < \infty$

□

The clients interact with the network according to the specific workflow that they are currently following. However, clients may have several different workflows that they are able to execute. The choice of workflow for the different client types is modeled as a discrete random variable W with outcomes in a set of available workflows $\mathbf{W}$. The probability mass function

$p(w)$ of the discrete random variable W defines the likelihood that a client of a given type will execute a given workflow.

Now that we have defined both the arrival process as well as the random variable that determines what workflow our client will execute we can define our clients.

Definition 4. A client type is defined by its arrival process together with its workflow random variable i.e. client $C = (Po(\lambda_C(t)), W)$

Examples of client types are: normal user, admin, web scraper or bot. To model the different workflows we use first order Markov chains. The state space for workflow w_i is S_{w_i} where $S_{w_i} \subseteq \mathbf{S}$. The transition matrix is $\mathbf{P}_{w_i}$. In equation (3.1) you can see what the transition matrix $\mathbf{P}_w$ would look like for the workflow w, depicted in graph form in Figure X.

$$\mathbf{P} = \begin{array}{c} \\ S_1 \\ S_2 \\ \varnothing \end{array} \begin{array}{c} \begin{array}{ccc} S_1 & S_2 & \varnothing \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \tag{3.1}$$

3.3 Expected service times

Given a client type with an arrival process and a workflow distribution one might wish to know the expected total service time that this client type will spend in the system. In the following subsections two ways of calculating this total service time are presented.

3.3.1 Expected service times using expected hitting time

One way of finding the total service time is to find the expected hitting time of the empty state $\varnothing$, as this represents the time it takes until termination of the connection. To illustrate the method we once again use the example of the Markov chain with the transition matrix seen in (3.1). We use the method shown in Section 2.2.2 to calculate the expected hitting time for the state $\varnothing$ given that the process starts in S_1 i.e. $\eta_{1,\varnothing}$. We begin by setting up the equations for S_1 and S_2

$$\begin{cases} \eta_{1,\varnothing} = 1 + \eta_{2,\varnothing} \\ \eta_{2,\varnothing} = 1 + \frac{1}{2}\eta_{2,\varnothing} + \frac{1}{2}\eta_{\varnothing,\varnothing}. \end{cases}$$

Functions
HTTP
SSH
SNMP
ICMP
IRC
PostgreSQL
FTP
DNS
Telnet

Table 3.1: The functions that are available to the clients through the network infrastructure

Solving for $\eta_{1,\varnothing}$ gives $\eta_{1,\varnothing} = 3$.

3.3.2 Expected service times using expected number of visits

Another way of finding the total service time is to calculate the expected number of visits to every transient state and summing them to get the average number of time steps spent in transient states before reaching the absorbing state. We once again refer to the Markov chain with transition matrix shown in (3.1). Now we calculate the expected number of visits for the transient states using (2.6). In this example $P_T = \{S_1, S_2\}$ since $\varnothing$ is a recurrent, and in this case absorbing, state.

$$V = (I - P_T)^{-1} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 0 & 0.5 \end{bmatrix} \right)^{-1} = \begin{bmatrix} 1 & -1 \\ 0 & 0.5 \end{bmatrix}^{-1} = \begin{matrix} \\ S_1 \\ S_2 \end{matrix} \begin{matrix} \begin{matrix} S_1 & S_2 \end{matrix} \\ \begin{bmatrix} 1 & 2 \\ 0 & 2 \end{bmatrix} \end{matrix}$$

Summing all the elements of the first row of the matrix V gives $\eta_{1,\varnothing} = 3$ which is the same results as with the previous method.

3.4 A minimal example

To illustrate how the model works we will try it on a minimal example. Consider a client type C whose population is increasing exponentially with time and also varies periodically with one cycle frequency (for example daily).

The client type $C = (Po(\lambda_C(t)), W)$ where W has one outcome w with an associated probability of one, i.e. $p(w) = 1$. The parameters of $\lambda_C(t)$ are as follows $\theta_0 = 2, \theta_1 = 0.003, \gamma_1 = 0.5, \phi_1 = 0, \omega_1 = 0.02$. Running a simulation and plotting the outcomes of the arrival process $Po(\lambda_C(t)$ for 1000 time steps we get what is shown in Figure 3.2. An alternative arrival process is shown in Figure 3.3. To demonstrate the algorithm for the usage of services by the clients we use the workflow shown in Figure 3.4. The workflow Markov chain w that the describes the clients' traversal through this workflow has this transition matrix:

$$\begin{bmatrix} 0.1 & 0.3 & 0.5 & 0 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0.9 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0.9 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0 & 0.4 & 0.4 & 0.2 \\ 0 & 0 & 0 & 0 & 0 & 0.7 & 0.3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The Markov chain is shown in graph form in Figure 3.5. To show how the clients traverse through the network we simulated 100 outcomes of the Markov chain from $t = 0$ to $t = 7$. The results can be seen in Figure 3.6. One can gather from the graph that some paths are more likely to be traversed than others.

3.5 Calculating expected number of clients in the network

First we consider the case where there is only one client type $(Po(\lambda(t), W))$ in the population with one workflow w and we are considering the time interval $[t_0, t_1]$, in this case there are two variables that when they become large significantly affect the memory usage of this model: the maximum expected client population size at any one time (N) and the number of services (M). To analyze the expected number of clients in the network at any one time we use the $M/M/\infty$ queue described in Section X. The $M/M/\infty$ queue requires a constant arrival rate for the arrival Poisson process, since our arrival process has a varying arrival rate we have to make a decision on what constant arrival rate to choose based on our rate function. Since we are trying to find an upper bound of the amount of memory usage of the model it seems appropriate to use the maximum value of the rate function on the interval i.e.

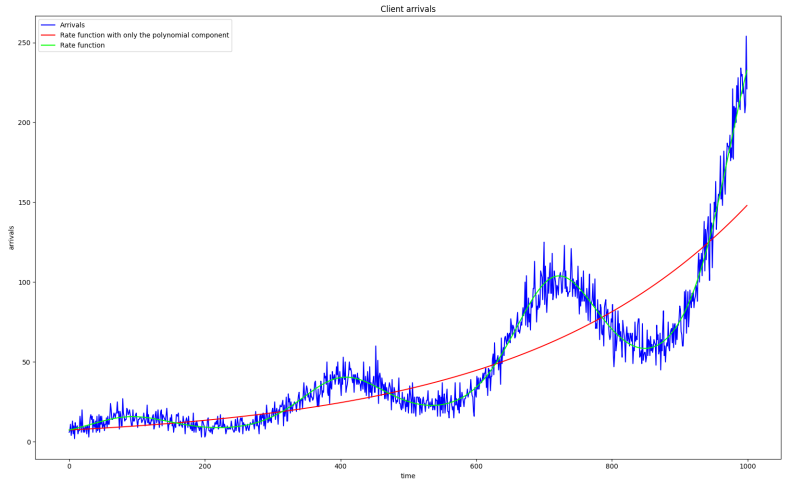

Figure 3.2: Client arrivals for an arrival process that is increasing exponentially with time and also has one periodic component

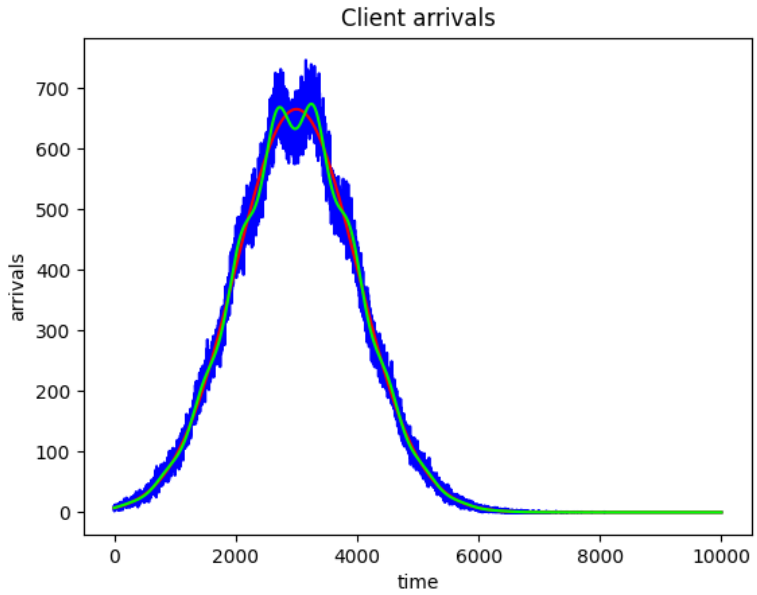

Figure 3.3: Client arrivals for an arrival process that has a models a large spike that decreases after some time

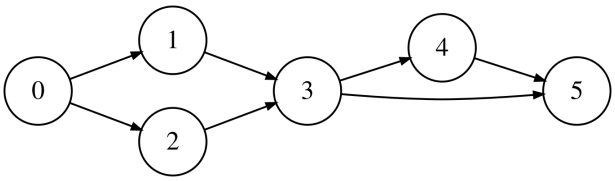

Figure 3.4: A workflow with five services

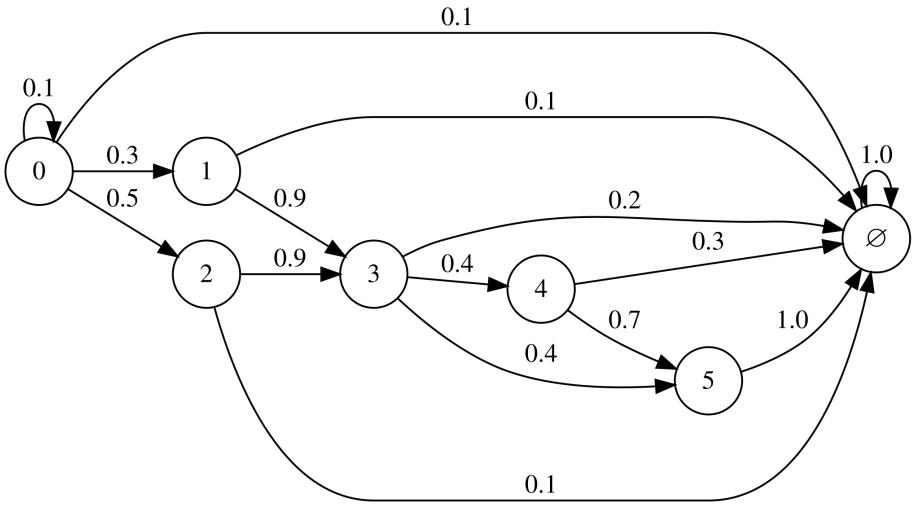

Figure 3.5: The workflow Markov chain w in graph form

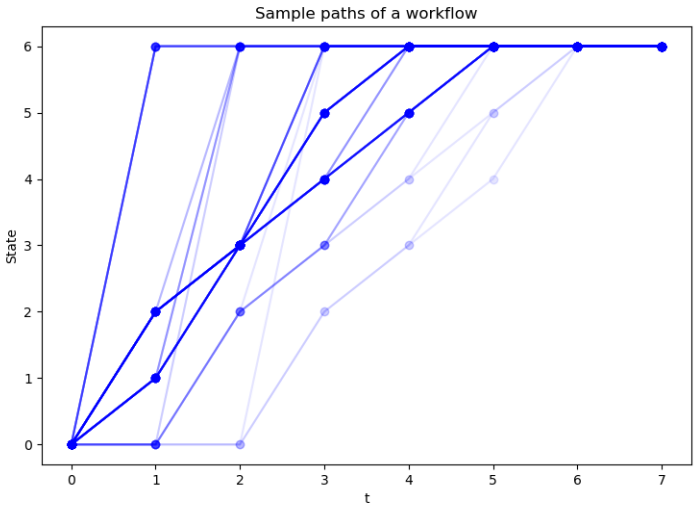

Figure 3.6: Sample trajectories of the workflow Markov chain w. The opacity of the line indicates the number of samples that traversed that specific trajectory.

$$\lambda = \max_{t \in [t_0, t_1]} \lambda(t)$$

Using the formula in Section 3.3.2 to get the expected service time* gives

$$\mu = \sum_{i=0}^{M} V_{1,i}$$

Now using the results described in Section X we conclude that the number of clients in the system is a Poisson random variable with mean and variance $\frac{\lambda}{\mu}$.

3.6 Space complexity

To assess whether the model is scalable enough one can look at the space complexity of the algorithm. Since there are no heavy calculations involved the time complexity is likely negligible. Since the clients are run as parallelized threads in the emulation we choose to calculate the space complexity per client. The only variables that affect the space complexity of a given client is the

*It is important to note that we assume that the service times are exponentially distributed here

number of services in that the workflow of the client type of the client. If there are N services in a given workflow, then the size of the transition matrix is $N \times N = N^2$ meaning that the space complexity of the model is $\mathcal{O}(N^2)$.

3.7 Summary

The model described in this chapter consists of two independent parts. The first part is the generation of client arrivals that is done with a time-varying Poisson process. The rate function of the process described in Section 2.1.1 makes it possible to model many different scenarios for the client population, as can be seen in figures X-Y. And since the client processes can be parallelized and the infrastructure is scalable with docker containers the limit on the amount of concurrent clients is very high, meaning that the model is scalable.

The second part of the model is the Markov chain that describes the clients behaviour in the network. The network is defined as a network of services that a given client can visit and revisit a number of times before exiting the network, and the probabilities of transition from state to state is encoded in a Markov chain. This approach is able to model any network where the clients access services in a sequential manner and the behaviour is possible to encode in a Markov chain. The model is agnostic to the underlying IT infrastructure and even the specific interface that the clients are using to interact with the system.

Chapter 4

Evaluation of the model

To evaluate the model it has been implemented to be used in the testbed presented in [2, 3, 4, 5, 6, 7, 8]. The client generator is run in a docker container that interacts with the rest of the network. The client generator is then used to simulate a set of scenarios that might occur in real systems.

4.1 The testbed used for evaluation

The testbed emulates an IT infrastructure by creating a virtual network. An overview of how it works can be seen in Figure 1.1The physical hosts that contain the services of the network are emulated using Docker containers [30], i.e. lightweight executable packages that include runtime systems, code, system tools, system libraries, and configurations. Allocation of CPU and memory resources to the containers is enforced using cgroups. Switches in the emulated network are emulated using docker containers that run Open vSwitch [31] and connect to controllers through the OpenFlow protocol [32]. Since the switches are programmed through flow tables, they can act either as classical layer 2 switches or as routers, depending on the flow table configurations. To emulate network connectivity virtual links implemented by Linux bridges are used. By using network namespaces that create logical copies of the physical hosts' network stacks network isolation is achieved. In the case that an emulated network spans multiple physical servers, the emulated traffic is tunneled over the physical network using VXLAN tunnels [33]. In other words, the physical network provides a substrate network, on top of which virtual networks are overlaid. Network conditions of virtual links are configured using the NetEm module in the Linux kernel [34]. This module allows fine-gained configuration of bit rates, packet delays, packet

loss probabilities, jitter, and packet reordering probabilities. The attackers are simulated using programs that select actions from a predefined set, which is available in [35]. To select the actions the attacker uses an *attacker policy*, which could depend on metrics collected by monitoring agents. The defender is emulated through the execution of gRPC API calls which are selected from a set defined in [35] according to a *defender policy*, which also could depend on metrics collected by monitoring agents.

4.2 Method

Interesting scenarios

- Large load spike - Two consecutive load spikes - Periodic spikes - Spike with periodic variation - Exponential growth that plateaus - Exponential decline from a high level

4.3 Results

A time-step in the emulation is defined as $t = 30s$.

4.4 Discussion

4.5 Conclusion

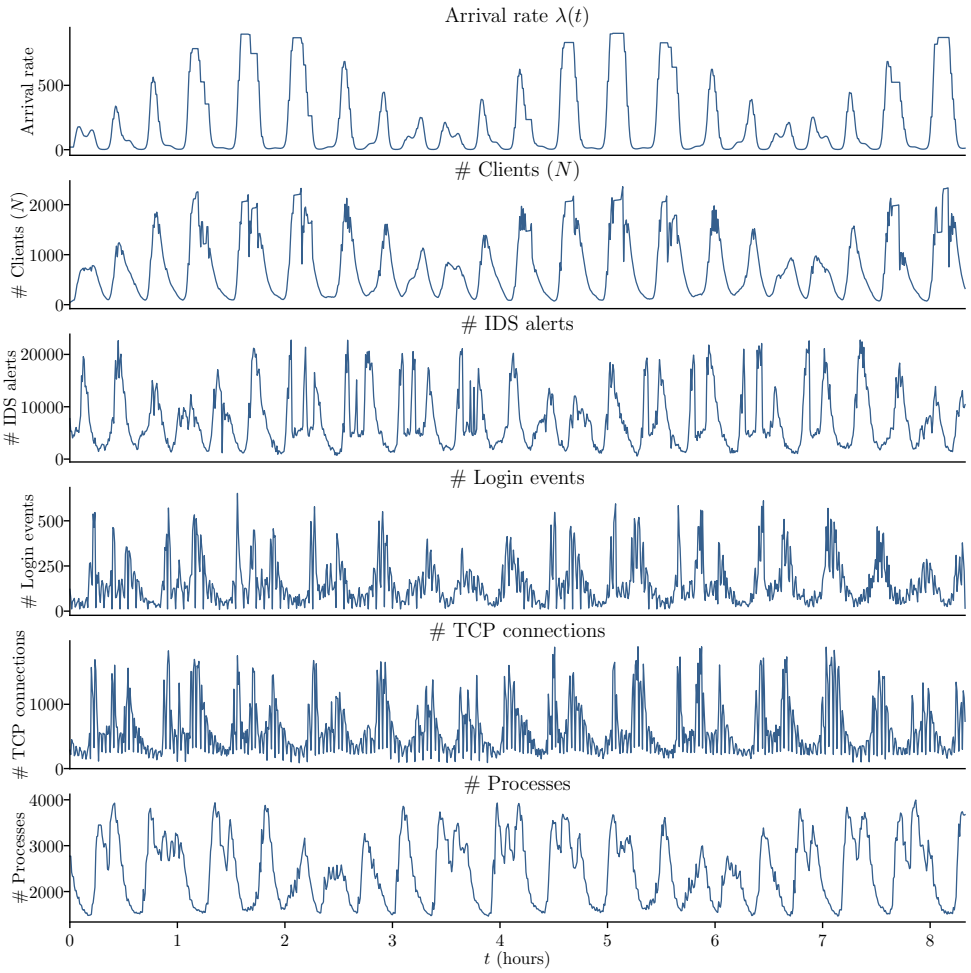

Figure 4.1: Empirical distributions of system metrics with an EPTMP arrival process using parameters $\boldsymbol{\theta} = [3.6269]$, $\boldsymbol{\gamma} = [2.1, 1.1]$, $\boldsymbol{\omega} = [0.1731, 0.3264]$, and $\boldsymbol{\phi} = [-0.6193, 0.5]$; and a single workflow w with a single service and expected service time 150s; the EPTMP parameters are based on [24, Case 1].

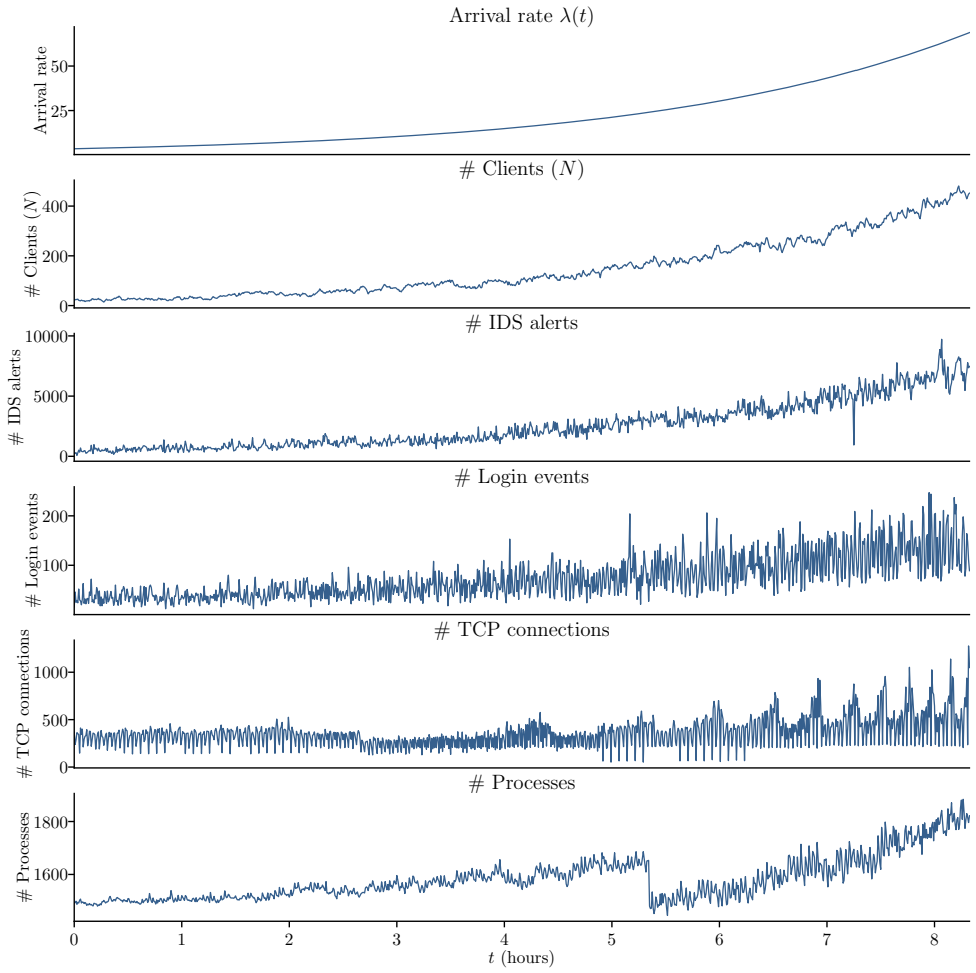

Figure 4.2: Empirical distributions of system metrics with an EPTMP arrival process using parameters $\boldsymbol{\theta} = [1, 0.003]$, $\boldsymbol{\gamma} = [0]$, $\boldsymbol{\omega} = [0]$, and $\boldsymbol{\phi} = [0]$; and a single workflow w with a single service and expected service time 150s.

References

[1] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on Industrial Informatics*, 2019. [Page 1.]

[2] K. Hammar and R. Stadler, "Finding effective security strategies through reinforcement learning and Self-Play," in *International Conference on Network and Service Management (CNSM 2020)*, Izmir, Turkey, 2020. [Pages ix, 1, 2, 3, 4, 6, and 25.]

[3] ——, "Learning intrusion prevention policies through optimal stopping," in *International Conference on Network and Service Management (CNSM 2021)*, Izmir, Turkey, 2021, https://arxiv.org/pdf/2106.07160.pdf. [Pages ix, 1, 2, 3, 4, 6, and 25.]

[4] ——, "Learning security strategies through game play and optimal stopping," *arXiv preprint arXiv:2205.14694*, 2022. [Pages ix, 1, 2, 3, 4, 6, and 25.]

[5] ——, "An online framework for adapting security policies in dynamic it environments," in *2022 18th International Conference on Network and Service Management (CNSM)*, 2022. doi: 10.23919/CNSM55787.2022.9964838 pp. 359–363. [Pages ix, 1, 2, 3, 4, 6, and 25.]

[6] ——, "Intrusion prevention through optimal stopping," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2333–2348, 2022. doi: 10.1109/TNSM.2022.3176781 [Pages ix, 1, 2, 3, 4, 6, and 25.]

[7] ——, "A system for interactive examination of learned security policies," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, 2022. doi: 10.1109/NOMS54207.2022.9789707 pp. 1–3. [Pages ix, 1, 2, 3, 4, 6, and 25.]

[8] ——, "Learning near-optimal intrusion responses against dynamic attackers," 2023, https://arxiv.org/abs/2301.06085. [Online]. Available: https://arxiv.org/abs/2301.06085 [Pages ix, 1, 2, 3, 4, 6, and 25.]

[9] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," in *Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, 1998, pp. 151–160. [Page 5.]

[10] S. Antonatos, K. G. Anagnostakis, and E. P. Markatos, "Generating realistic workloads for network intrusion detection systems," in *Proceedings of the 4th International Workshop on Software and Performance*, 2004, pp. 207–215. [Page 5.]

[11] R. Hashemian, D. Krishnamurthy, and M. Arlitt, "Web workload generation challenges–an empirical investigation," *Software: Practice and Experience*, vol. 42, no. 5, pp. 629–647, 2012. [Page 5.]

[12] C. Amza, E. Cecchet, A. Chanda, A. L. Cox, S. Elnikety, R. Gil, J. Marguerite, K. Rajamani, and W. Zwaenepoel, "Specification and implementation of dynamic web site benchmarks," in *5th Workshop on Workload Characterization*, no. CONF, 2002. [Page 5.]

[13] Y.-C. Cheng, U. Hölzle, N. Cardwell, S. Savage, and G. M. Voelker, "Monkey see, monkey do: A tool for tcp tracing and replaying," 2004. [Page 5.]

[14] A. O. Pashko and I. V. Rozora, "Accuracy of simulation for the network traffic in the form of fractional brownian motion," in *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*, 2018. doi: 10.1109/TCSET.2018.8336328 pp. 840–845. [Page 5.]

[15] C. V. Wright, C. Connelly, T. Braje, J. C. Rabek, L. M. Rossey, and R. K. Cunningham, "Generating client workloads and high-fidelity network traffic for controllable, repeatable experiments in computer security," in *International workshop on recent advances in intrusion detection*. Springer, 2010, pp. 218–237. [Page 5.]

[16] F. Chinchilla, M. Lindsey, and M. Papadopouli, "Analysis of wireless information locality and association patterns in a campus," in *IEEE INFOCOM 2004*, vol. 2. IEEE, 2004, pp. 906–917. [Pages 5 and 10.]

[17] M. Papadopouli, H. Shen, and M. Spanakis, "Characterizing the duration and association patterns of wireless access in a campus," in *11th European Wireless Conference 2005-Next Generation wireless and Mobile Communications and Services*. VDE, 2005, pp. 1–7. [Page 5.]

[18] ——, "Modeling client arrivals at access points in wireless campus-wide networks," in *2005 14th IEEE Workshop on Local & Metropolitan Area Networks*. IEEE, 2005, pp. 6–pp. [Page 5.]

[19] R. Begleiter, R. El-Yaniv, and G. Yona, "On prediction using variable order markov models," *Journal of Artificial Intelligence Research*, vol. 22, pp. 385–421, 2004. [Page 6.]

[20] J. Yang, J. Xu, M. Xu, N. Zheng, and Y. Chen, "Predicting next location using a variable order markov model," in *Proceedings of the 5th ACM SIGSPATIAL International Workshop on GeoStreaming*, 2014, pp. 37–42. [Page 6.]

[21] C. Dimitrakakis, "Bayesian variable order markov models," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 161–168. [Page 6.]

[22] P. A. Lewis and G. S. Shedler, *Simulation of Nonhomogeneous Poisson Processes by Thinning*, 1978. [Page 7.]

[23] A. Vedyushenko, "Non-homogeneous poisson process-estimation and simulation," 2018. [Pages 7, 8, and 9.]

[24] M. E. Kuhl, J. R. Wilson, and M. A. Johnson, "Estimating and simulating poisson processes having trends or multiple periodicities," *IIE transactions*, vol. 29, no. 3, pp. 201–211, 1997. [Pages ix, 8, and 27.]

[25] R. L. Streit, *Poisson point processes: imaging, tracking, and sensing*. Springer Science & Business Media, 2010. [Page 8.]

[26] F. Hillier and G. Lieberman, "Introduction to operations research, mcgraw hill," *Inc. New York*, pp. 4–15, 1995. [Page 10.]

[27] G. Kochanski, "Markov models, hidden and otherwise," *Retrieved July*, vol. 11, p. 2011, 2005. [Page 10.]

[28] M. Aldridge, “Math2750 introduction to markov processes,” 2022. [Online]. Available: https://mpaldridge.github.io/math2750/S08-hitting-times.html [Page 10.]

[29] K. Sigman, “Notes on transient states,” 2016. [Pages 11 and 12.]

[30] D. Merkel, “Docker: lightweight linux containers for consistent development and deployment,” *Linux journal*, vol. 2014, p. 2, 2014. [Page 25.]

[31] B. Pfaff *et al.*, “The design and implementation of open vSwitch,” in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, 2015. [Online]. Available: https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/pfaff [Page 25.]

[32] N. McKeown *et al.*, “Openflow: Enabling innovation in campus networks,” *SIGCOMM Comput. Commun. Rev.*, p. 69–74, mar 2008. doi: 10.1145/1355734.1355746. [Online]. Available: https://doi.org/10.1145/1355734.1355746 [Page 25.]

[33] M. Mahalingam *et al.*, “Virtual extensible local area network (vxlan): A framework for overlaying virtualized layer 2 networks over layer 3 networks,” 2014, https://www.rfc-editor.org/rfc/rfc7348. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7348 [Page 25.]

[34] S. Hemminger, “Network emulation with netem,” *Linux Conf*, 2005. [Page 25.]

[35] K. Hammar, “Cyber security learning environment,” 2023, https://limmen.dev/csle/. [Online]. Available: https://limmen.dev/csle/ [Page 26.]

TRITA-EECS-EX- 2023:0000

€€€€ For DIVA €€€€

{
”Author1”: { ”Last name”: ”Lagerqvist”,
”First name”: ”Arvid”,
”E-mail”: ”arvidlag@kth.se”,
”organisation”: {”L1”: ”School of Engineering Sciences”,
}
},
”Cycle”: ”2”,
”Course code”: ”SF250X”,
”Credits”: ”30.0”,
”Degree1”: {”Educational program”: ”Master’s Programme, Applied and Computational Mathematics, 120 credits”
,”programcode”: ”TTMAM”
,”Degree”: ”Degree of Master of Science in Engineering, Master’s degree in Applied and Computational Mathematics”
,”subjectArea”: ”Computer Science and Engineering”
},
”Title”: {
”Main title”: ”Emulation and Stochastic Modeling of Client Populations in IT Infrastructures”,
”Language”: ”eng” },
”Alternative title”: {
”Main title”: ”Emulation av klientpopulationer för IT-infrastrukturer”,
”Subtitle”: ”För användning i ”Digital Twins”-teknologi”,
”Language”: ”swe”
},
”Supervisor1”: { ”Last name”: ”Hammar”,
”First name”: ”Kim”,
”E-mail”: ”kimham@kth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
”L2”: ”Division of Network and Systems Engineering” }
},
”Supervisor2”: { ”Last name”: ”Sandberg”,
”First name”: ”Mattias”,
”E-mail”: ”msandb@kth.se”,
”organisation”: {”L1”: ”School of Engineering Sciences”,
”L2”: ”Numerisk Analys” }
},
”Examiner1”: { ”Last name”: ”Jarlebring”,
”First name”: ”Elias”,
”E-mail”: ”eliasj@kth.se”,
”organisation”: {”L1”: ”School of Engineering Sciences”,
”L2”: ”Numerisk Analys” }
},
”National Subject Categories”: ”10105, 10106”,
”Other information”: {”Year”: ”2023”, ”Number of pages”: ”1,33”},
”Copyrightleft”: ”copyright”,
”Series”: { ”Title of series”: ”TRITA-EECS-EX” , ”No. in series”: ”2023:0000” },
”Opponents”: { ”Name”: ”A. B. Normal & A. X. E. Normalè”},
”Presentation”: { ”Date”: ”2022-03-15 13:00”
,”Language”:”eng”
,”Room”: ”via Zoom https://kth-se.zoom.us/j/ddddddddddd”
,”Address”: ”Isafjordsgatan 22 (Kistagången 16)”
,”City”: ”Stockholm” },
”Number of lang instances”: ”2”,
”Abstract[eng]”: €€€€

```
\begin{comment}
\generalExpl{Enter your abstract here!}

Write an abstract that is about 250 and 350 words (1/2 A4-page)  with the following components:
% key parts of the abstract
\begin{itemize}
  \item What is the topic area? (optional) Introduces the subject area for the project.
  \item Short problem statement
  \item Why was this problem worth a Bachelor's/'Masters thesis project? (\ie, why is the problem
  both significant and of a suitable degree of difficulty for a Bachelor's/'Masters thesis project?
  Why has no one else solved it yet?)
  \item How did you solve the problem? What was your method/insight?
  \item Results/Conclusions/Consequences/Impact: What are your key results/\linebreak[4]conclusions?
  What will others do based on your results? What can be done now that you have finished - that could
  not be done before your thesis project was completed?
\end{itemize}
\end{comment}
```

€€€€,
”Keywords[eng]”: €€€€
Digitala tvillingar, Stokastiska processer, Markovkedjor, Tidsvarierande Poissonprocesser €€€€,
”Abstract[swe]”: €€€€

```
\begin{comment}
\generalExpl{Enter your Swedish abstract or summary here!}
\sweExpl{Alla avhandlingar vid KTH \textbf{måste ha} ett abstrakt på både \textit{engelska} och
\textit{svenska}.\\
Om du skriver din avhandling på svenska ska detta göras först (och placera det som det första
abstraktet) - och du bör revidera det vid behov.}

\engExpl{If you are writing your thesis in English, you can leave this until the draft version that
goes to your opponent for the written opposition. In this way, you can provide the English and
Swedish abstract/summary information that can be used in the announcement for your oral
presentation.\\If you are writing your thesis in English, then this section can be a summary targeted
at a more general reader. However, if you are writing your thesis in Swedish, then the reverse is
true - your abstract should be for your target audience, while an English summary can be written
targeted at a more general audience.\\This means that the English abstract and Swedish sammnfattning
or Swedish abstract and English summary need not be literal translations of each other.}

\warningExpl{Do not use the \textbackslash glspl\{\} command in an abstract that is not in English,
as my programs do not know how to generate plurals in other languages. Instead, you will need to
spell these terms out or give the proper plural form. In fact, it is a good idea not to use the
glossary commands at all in an abstract/summary in a language other than the language used in the
\texttt{acronyms.tex file} - since the glossary package does \textbf{not} support use of more than
one language.}

\engExpl{The abstract in the language used for the thesis should be the first abstract, while the
Summary/Sammanfattning in the other language can follow}
\end{comment}
```

€€€€,
"Keywords[swe]": €€€€
€€€€,
}

acronyms.tex

```
%%% Local Variables:
%%% mode: latex
%%% TeX-master: t
%%% End:
% The following command is used with glossaries-extra
\setabbreviationstyle[acronym]{long-short}
% The form of the entries in this file is \newacronym{label}{acronym}{phrase}
%                                      or \newacronym[options]{label}{acronym}{phrase}
% see "User Manual for glossaries.sty" for the  details about the options, one example is shown below
% note the specification of the long form plural in the line below
\newacronym[longplural={Debugging Information Entities}]{DIE}{DIE}{Debugging Information Entity}
%
% The following example also uses options
\newacronym[shortplural={OSes}, firstplural={operating systems (OSes)}]{OS}{OS}{operating system}

% note the use of a non-breaking dash in long text for the following acronym
\newacronym{IQL}{IQL}{Independent -QLearning}

\newacronym{KTH}{KTH}{KTH Royal Institute of Technology}

\newacronym{LAN}{LAN}{Local Area Network}
\newacronym{VM}{VM}{virtual machine}
% note the use of a non-breaking dash in the following acronym
\newacronym{WiFi}{-WiFi}{Wireless Fidelity}

\newacronym{WLAN}{WLAN}{Wireless Local Area Network}
\newacronym{UN}{UN}{United Nations}
\newacronym{SDG}{SDG}{Sustainable Development Goal}
```