Chapter 3

Bayesian Activity Modelling for Network Flow Data

Henry Clausen[*,§], Mark Briers[†] and Niall M. Adams[*,‡]

[*]*Department of Mathematics, Imperial College London, London SW7 2AZ, UK*
[†]*Alan Turing Institute, 96 Euston Rd, Kings Cross, London NW1 2DB, UK*
[‡]*Data Science Institute, Imperial College London, London SW7 2AZ, UK*
[§]*henry.clausen@ed.ac.uk*

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 chapter, 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.

1. Introduction

Advanced persistent threats (APTs), which are responsible for many severe data breaches in enterprise computer networks, are characterised by the intruder maintaining a presence in the compromised network for long-term control and data collection, possibly lasting months (Tankard, 2011).

APT attacks often circumvent strong firewalls, which protect high-value assets, through a technique called *pivoting* (also known as *lateral*

movement). 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 with no high-level access and the corresponding protection. 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 *Command-and-Control* (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 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,

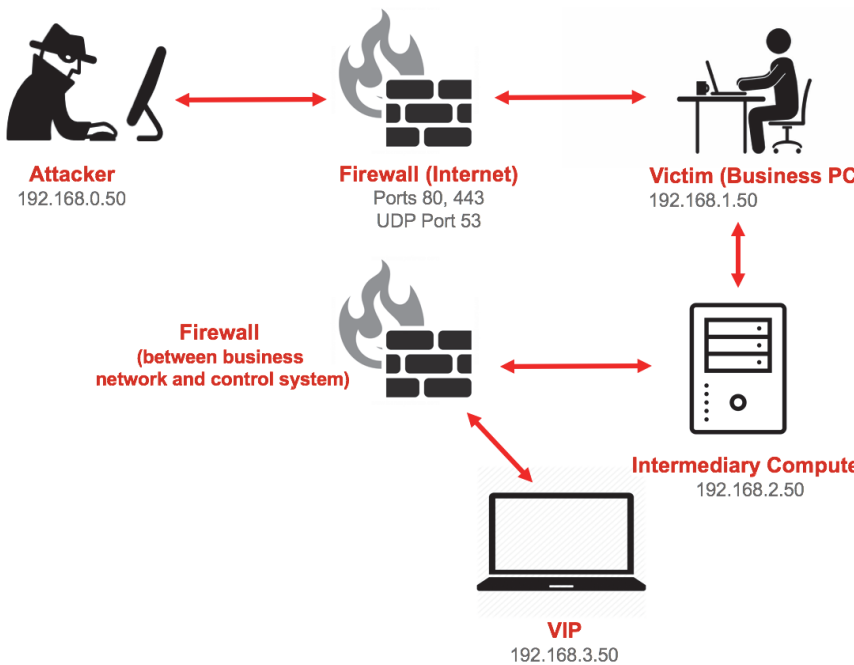

Fig. 1: A typical pivoting attack chain. The attacker penetrates the network at a weak point and moves through the network to find its target.

and the detection of such is a clear indication of an ongoing network intrusion. A key aspect that is necessary to identify such chains is the distinction of human-induced network activity from machine-driven activity, 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.

1.1. *Network flow arrivals and problems with regular MMPP framework*

A network flow is a summary of a connection between two computers and contains information about the connection such as a timestamp and duration, the transmitted packages and data size, and the involved IP addresses, transmission protocol, and connection ports. Due to their richness of network information, network flow logs are 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.

The aim of this work is to use the arrival times of network flows on individual computers to identify different states of user activity. Network arrivals are often simulated as Poisson processes for analytic simplicity, and a well-established and tractable model of arrival events with alternating rates is the *Markov-modulated Poisson process* (MMPP), which assumes piece-wise Poisson distributed arrivals with Poisson rates switched via an underlying Markov process. However, a number of traffic studies have shown that packet or flow inter-arrivals are not exponentially distributed, but show among other characteristics a significantly stronger tail behaviour (Leland *et al.*, 1993; Muscariello *et al.*, 2005; Paxson and Floyd, 1995). These tails cannot be accurately modelled by a Poisson distribution. Consequently, in a simple MMPP model tail observations receive a disproportionately low likelihood, which in turn causes state changes in the sampled MMPP for an observed tail event. Therefore, it is crucial to take the shape of the observed event arrival distribution into account.

Recent work argues convincingly that network traffic is much better modelled using self-similar processes (Leland *et al.*, 1993; Park and Willinger, 2000). However, currently none of the mathematical models allow an analytical solution when used for network traffic generation. In this chapter, we will propose a hierarchical extension of the simple MMPP model that is able to capture the tail behaviour of flow arrivals in a more accurate way while still retaining the analytical simplicity of a Markovian arrival process. We will then embed this model in a Gibbs sampling framework and demonstrate how to estimate different states of user activity from network arrival data. We will assume the number of identifiable states to be known. A justified model selection with interpretable activity states will be postponed to future work.

This work has been done in the context of a master's thesis. For more detailed information we refer to Clausen (2017).

2. Methodology

A MMPP is a Poisson process whose intensity depends on the current state of an independently evolving continuous time Markov chain with finite state-space.

An MMPP is parametrised as following:

Definition 1 (Clausen 2017; Rydén 1996). Let $[0, t_{\text{obs}}]$ be the time window of observation. Let $\{X_t\}$ for $t \in [0, t_{\text{obs}}]$ be a discrete finite-state homogeneous Markov process evolving inside this time window on a state space $\{1, \ldots, M\}$ of cardinality M. Let $\mathbf{Q}$ be the infinitesimal generator of $\{X_t\}$, satisfying $Q_{ii} = -\sum_{j \neq i} Q_{ij}$, $Q_{ii} \leq 0$.

Let $\boldsymbol{\lambda} \triangleq \{\lambda_1, \ldots, \lambda_M\}$ be the Poisson process rates. $N(t)$ is 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. The two-dimensional set $\{X_t, N(t)\}$ is called the MMPP.

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. 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.

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 $\{I_1 = [t_0 = 0, t_1), \ldots, I_n = [t_{n-1}, t_n = t_{\text{obs}})\}$ 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.

In order to build a Gibbs sampling framework, we are specifically interested in sampling from the following distribution:

$$P(\{X_t\}|\{z_1, \ldots, z_n\}, \mathbf{Q}, \boldsymbol{\lambda}).$$

We will now outline the steps involved in the sampling procedure.

2.1. *Sampling* $\{X_t\}$

We will first look at the following probability:

$$P_{jk}^{(z_i)} = P \text{ (there are } z_i \text{ events in } (t_{n-1}, t_n) \text{ and } X_{t^*} = k \,|X_0 = j).$$

We can derive this probability by defining a meta-Markov process V_t: Let $z_{\max} = \max_{i\in\{1,\ldots,n\}}(z_i)$. Define the new $(M \cdot z_{\max} + 1)$-dimensional state-space $S = (1^{(0)}, \ldots, M^{(0)}, 1^{(1)}, \ldots, M^{(1)}, \ldots, 1^{(z_{\max})}, \ldots, M^{(z_{\max})}, 1^*)$. Let $\{t'_1, t'_2, \ldots\}$ be a (possibly empty) set of event arrival times inside $[t_{n-1}, t_n)$.

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

$$V_t = \begin{cases} X_t^{(0)} & t_{n-1} \le t \le t_1' \\ \vdots & \\ X_t^{(i)} & t_i' \le t < t_{i+1}' \quad t \in (t_{n-1}, t_n). \\ \vdots & \\ 1^* & t_{z_{\max}+1}' \le t. \end{cases} \tag{1}$$

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

$$\mathbf{G_V} = \begin{pmatrix} \mathbf{Q} - \Lambda & \mathbf{3} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} - \Lambda & \mathbf{3} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \dots & \dots & \mathbf{0} & \mathbf{Q} - \Lambda & \boldsymbol{\lambda} \\ \mathbf{0} & \dots & \dots & \mathbf{0} & \mathbf{0} & 0 \end{pmatrix}. \tag{2}$$

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

$$P_{jk}^{(z_i)} = P(V_{t_n} = k^{(z_i)} \mid V_{t_n - 1} = j^{(0)}) = \left[\exp(\mathbf{G_V} t^*) \right]_{j, k + z_i \cdot M}.$$

For more information on the exact calculation of $P_{jk}^{(z_i)}$, see Clausen (2017); Fearnhead and Sherlock (2006).

2.1.1. *Forward–Backward algorithm*

Due to its Markovian nature, we can formulate the MMPP as a discrete *hidden Markov model* (HMM), with the Markov chain in the model being the state of the Markov process at each time interval. Since the state-space of our Markov model is discrete, we can sample the Markov chain recursively in two steps using the so-called *Forward–Backward algorithm* (Baum, 1972; Devijver, 1985). The backward-part of the Forward–Backward algorithm calculates the posterior distribution of each element of the Markov chain conditional on the number of observations and the previous element recursively while the forward step samples each element sequentially. We will

outline the most important steps in the context of the Markov-modulated Poisson process. For a more general description, see Baum (1972); Fearnhead and Sherlock (2006).

Define

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

For $k = n$, we have

$$\begin{aligned}\mathbf{A}^{(n)}_{ij} = P^{(z_n)}_{ij} &= P(z_n, X_{t_n} = j | X_{t_{n-1}} = i) \\ &= \left[\exp(\mathbf{G_V} t^*)\right]_{i, j+z_n \cdot M}.\end{aligned}$$

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

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

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:

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

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

$$\begin{aligned}P(X_{t_k} = s | \{z_1, \ldots, z_n\}, X_{t_{k-1}} \\ = s_{k-1}) = \frac{\left[\exp(\mathbf{G_V} t^*)\right]_{s_{k-1}, s+z_k \cdot M} \left[\mathbf{A}^{(k+1)} \mathbf{1}\right]_s}{\left[\mathbf{A}^{(k)} \mathbf{1}\right]_{s_{k-1}}}.\end{aligned}$$

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

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

2.1.2. *Simulate full underlying Markov process*

We are now able to sample the state of the Markov process $\{X_t\}$ at the times $\{t_0, \ldots, t_n\}$. Finally, we want to sample the whole trajectory of the Markov process conditionally, i.e., we want to sample from all

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

where we used the definition of $\{V_t\}$ and its generator from (1) and (2). We are therefore simply looking at the evolution of the Markov process $\{V_t\}$ with specified endpoints. Sampling the trajectory of $\{V_t\}$ gives us both a sampled trajectory of $\{X_t\}$ as well as a sample of the exact arrival times $\{t'_1, \ldots, t'_l\}$ where l is the number of all observed flows.

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*. We use both techniques where appropriate during the sampling of $\{V_t\}$, depending on the particular endpoint states V_{t_k}, $V_{t_{k+1}}$ to benefit from the advantages of both methods. A detailed description of both algorithms and their particular advantages and disadvantages is given in Clausen (2017); Hobolth and Stone (2009).

2.2. *Hierarchical MMPP extension*

It is known that flow arrivals are not piecewise exponentially distributed, but have a strong tail behaviour which has severe consequences when trying to fit a simple MMPP. Muscariello *et al.* (2005) suggests the use of a hierarchical Markovian Poisson model in order generate characteristics that match those measured on network traffic while still maintaining analytical tractability. However, in their work they neither provide a general approach to fit the involved parameters nor a way to calculate or sample from $P(\{X_t\}|\{z_1, \ldots, z_n\})$, which eventually is our ultimate interest in this work. We now propose a model that adopts the idea of MMPP arrivals acting as a latent generator of observed events while remaining analytically tractable. We will then embed our model in a Gibbs sampling framework that samples the Markov process trajectory and the model parameters jointly. A graphical model representation of the proposed model is given in Figure 2.

As before, let $\{X_t\}$ for $t \in [0, t_{\text{obs}}]$ be the unobserved discrete Markov process with infinitesimal generator $\mathbf{Q}$, and let $N(t)$ be a Poisson process with rate $\lambda(t) = \lambda_{i=X_t}$. However, $N(t)$ does not generate flow events anymore, but unobserved events called *sessions*.[a] Let

[a] In reference to Muscariello *et al.* (2005).

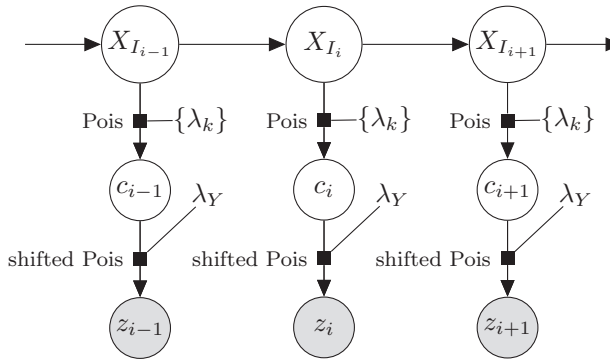

Fig. 2: Graphical model for our proposed hierarchical MMPP.

$\{I_1 = [t_0 = 0, t_1), \ldots, I_n = [t_{n-1}, t_n = t_{\text{obs}})\}$ 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, a session s_i instantaneously generates y_i events that are observed. $y_i \in \mathbb{N}_{>0}$ is a discrete random variable following a shifted Poisson distribution:

$$y_i - 1 | \boldsymbol{\lambda} \sim \text{Pois}(\lambda_Y).$$

The shift introduced is beneficial since it ensures that every session generates at least one flow event. The flow events z_k observed during interval I_k are now defined as

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

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

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

2.3. *Gibbs sampler*

Fearnhead and Sherlock (2006) proposed a Gibbs sampling framework for MMPP models that estimates $\{X_t\}$ and the model parameters jointly. We now extend this Gibbs sampler to sample from the following distribution:

$$P(\{X_t\}, \{c_1, \ldots, c_n\}, \mathbf{Q}, \boldsymbol{\lambda} | \{z_1, \ldots, z_n\}, I),$$

where I represents the available prior information. To implement such a Gibbs sampler, we use the following conditional distributions:

1. $P(\{X_t\}, \{t'_1, \ldots, t'_l\} | \{z_1, \ldots, z_n\}, \{c_1, \ldots, c_n\}, \mathbf{Q}, \boldsymbol{\lambda}, \lambda_Y)$
2. $P(\{c_1, \ldots, c_n\} | \{z_1, \ldots, z_n\}, \{X_t\}, \mathbf{Q}, \boldsymbol{\lambda}, \lambda_Y)$
3. $P(\mathbf{Q}, \boldsymbol{\lambda}, \lambda_Y | \{z_1, \ldots, z_n\}, \{c_1, \ldots, c_n\}, \{X_t\}, I)$

1. In Section 2.1.2, we have seen how to sample the trajectory of an MMPP,[b] i.e., how to sample from

$$P(\{X_t\}, \{t'_1, \ldots, t'_l\} | \{c_1, \ldots, c_n\}, \mathbf{Q}, \boldsymbol{\lambda}).$$

Since $\{z_1, \ldots, z_n\}$ and $\{X_t\}$ are independent conditional on $\{c_1, \ldots, c_n\}$, we can rewrite

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

to be equal to the expression we already know how to sample from.

2. We define

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

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}:

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

This allows us to calculate the posterior distribution of c_k:

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

[b] Please note that c_i replaced z_i as the arrivals of the MMPP which are now latent while z_i now represents the observed events in the extended model.

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

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

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

where

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

is the time spent in state i,

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

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

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

is the number of transitions from state i into state j (Fearnhead and Sherlock, 2006). It is clear that the densities of λ_i and q_{ii} are Gamma while the joint densities of q_{ij}/q_{ii}, $j \in \{1, \ldots, M\}/i$ are Dirichlet distributed. Moreover, we know that the likelihood of $\{z_1, \ldots, z_k\}$ conditional on $\{c_1, \ldots, c_n\}$ is the product of shifted Poisson distributions with rates $c_k \lambda_Y$. If we employ a Bayesian framework with conjugate priors, it is possible to calculate the posterior distributions of $\mathbf{Q}, \boldsymbol{\lambda}, \lambda_Y$ conditional on $\{X_t\}$, $\{t'_1, \ldots, t'_l\}$, and $\{z_1, \ldots, z_n\}$.

We impose Gamma-priors on λ_i, q_{ii}, and λ_Y with hyper-parameters $\alpha_{\lambda,i}$, $\alpha_{q,i}$, $\alpha_{Y,i}$ and $\beta_{\lambda,i}$, $\beta_{q,i}$, $\beta_{Y,i}$:

$$\lambda_i \sim \Gamma(\alpha_{\lambda,i}, \beta_{\lambda,i}),$$
$$q_{ii} \sim \Gamma(\alpha_{q,i}, \beta_{q,i}),$$
$$\lambda_Y \sim \Gamma(\alpha_{Y,i}, \beta_{Y,i}).$$

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

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

The posterior distributions are then given by

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

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

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

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

where $\mathbf{r}_i = (r_{1,i},\ldots,r_{i-1,i},r_{i+1,i},\ldots r_{M,i})^T$.

3. Data

3.1. *LANL data*

This work was motivated by a data set containing 16 consecutive days of network flow data from the LANL's corporate, internal computer network (Kent, 2015). The network consists of 17,684 computers which can be identified through their individual label through the data. We applied our developed Gibbs sampler to several selected computers that were identified to be subject to human control. Our framework was able to identify different types of computer activity, which appear to resemble a very reasonable trace of human activity on the individual computers. However, we do not have any information about the actual activity conducted on each computer to verify our sampled results. Therefore, we conducted a controlled experiment on a specific computer which is described in the next section in order to validate the results of our model. We will focus on the data obtained from that experiment in this work. Results and further details on the LANL data can be found in Clausen (2017).

3.2. *Imperial College data*

Network flows do not give direct insight in specific user activities. To acquire some ground truth about the traces of human activity in network flow logs,

we conducted a controlled experiment during which a user conducts different selected activities on a single computer inside the Imperial College network.

The data set contains network flow data originating from a single source IP address inside the Imperial College network over a 3-hour time span. The computer corresponding to this IP address 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, with surfing and information gathering, but also without any user activity. 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:

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 programme open)

14:40 User log-out
User log-in

14:42 Open Mozilla Firefox, open several sites[c]
No further activity

15:00 Visiting `www.soundcloud.com`, downloading three files with a total size of 2 GB

15:04 Watching several videos on `www.youtube.com`, all of which are several minutes long

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

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

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. Any form of 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 therefore remove this communication from our data. More information about this process can be found in Clausen (2017).

Figure 3 depicts the number of cleaned flow events binned into 5-second intervals during the experiment. The colours represent the activity schedule.

3.2.1. *Spikes*

In the Imperial College data, we observe several large spikes, standing out from the rest of the data. These spikes primarily stem from new web

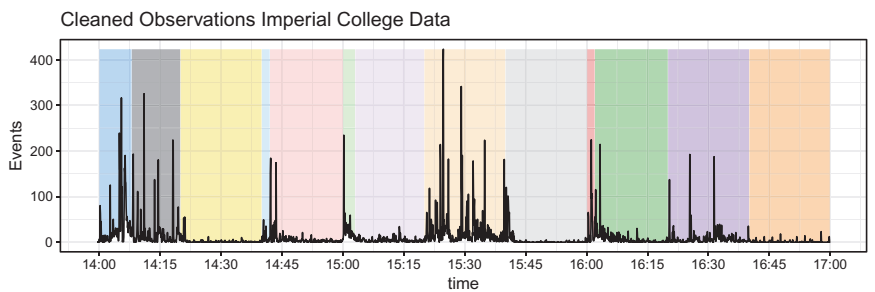

Fig. 3: Number of Imperial College flow events, binned into 5-seconds intervals. The coloured intervals correspond to the different stages during the experiment.

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, of which 297 are directed to the IP-address `155.198.142.8`, all via UDP-port 53 (which is responsible for DNS requests). We will identify these DNS-spikes as an additional new device state in our framework since their detection might be of interest for a broader intrusion detection framework.

3.3. *Distribution comparison*

Figure 4(a) depicts an excerpt of the Imperial College data during which the flow arrival rate supposedly stays constant, while Figure 4(b) shows the

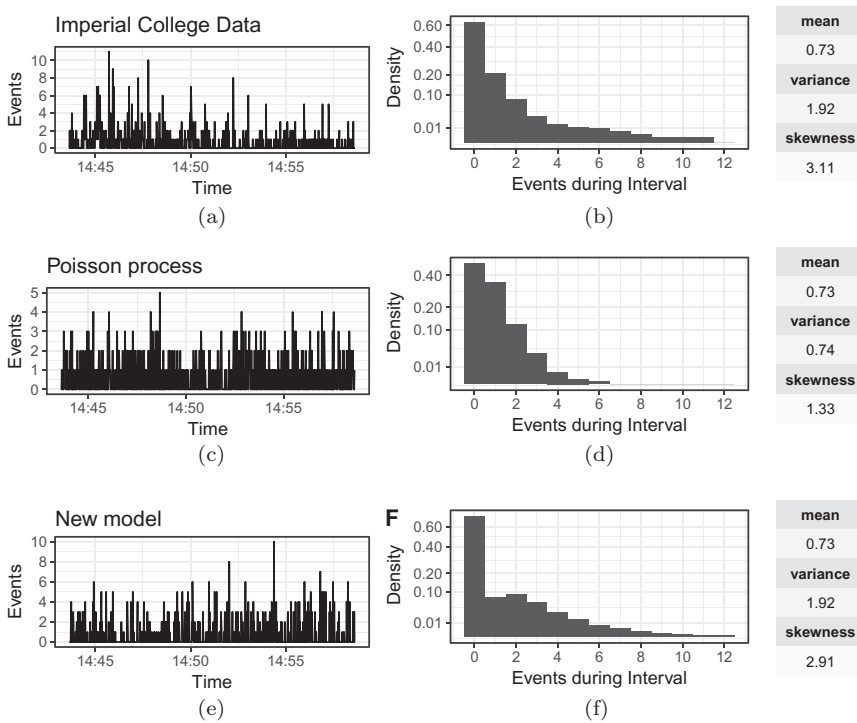

Fig. 4: Plot (a) shows an interval of the Imperial College flow data during which the flow arrival rate was supposedly constant (bin size 2 seconds). Plot (b) shows the corresponding empirical flow count distribution. Plots (c) and (d) show simulated arrivals from Poisson processes with the same mean as the data in (a). Plots (e) and (f) show simulated arrivals from our proposed model.

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.

A comparison of the first three moments of the depicted data shows that our proposed hierarchical Poisson model is better suited to imitate the tail behaviour of the observed data distribution.

3.4. *Results*

Using our proposed Gibbs sampler, we fit a 4-state MMPP model to the Imperial College data. We have to make sure that we choose the model parameter priors wide enough to allow for proper convergence of the posterior, but narrow enough to initialise the parameters correctly. Our choice of hyper parameters for λ_i is motivated by the range of observed flow events per bin while hyper parameters for q_{ii} are chosen to exclude unreasonable state decay times:

$$\alpha_{\lambda,i} = 1.2\frac{1}{\text{sec}}, \quad \beta_{\lambda,i} = 1,$$

$$\alpha_{q,1} = \alpha_{q,2} = \alpha_{q,3} = 5\frac{1}{\text{sec}},$$

$$\beta_{q,1} = \beta_{q,2} = \beta_{q,3} = 1000,$$

$$\alpha_{q,4} = 5\frac{1}{\text{sec}}, \quad \beta_{q,4} = 100.$$

Experience shows that $\lambda_Y > 10$ does not match the shape of a flow arrival distribution anymore, the hyper parameters are therefore chosen accordingly:

$$\alpha_{\lambda_Y} = 2, \quad \beta_{\lambda_Y} = 1.$$

Lastly, we employ an uninformative Dirichlet prior with $\boldsymbol{\alpha}_{D,i} = (1, 1, 1)^T$. We then proceed to generate 500 samples of $\{X_t\}, \mathbf{Q}, \boldsymbol{\lambda}, \lambda_Y|\{z_1, \ldots, z_n\}$ using our described Gibbs sampler. Figure 5 depicts an excerpt of the sampled $\{X_t\}$ for the Imperial College data.

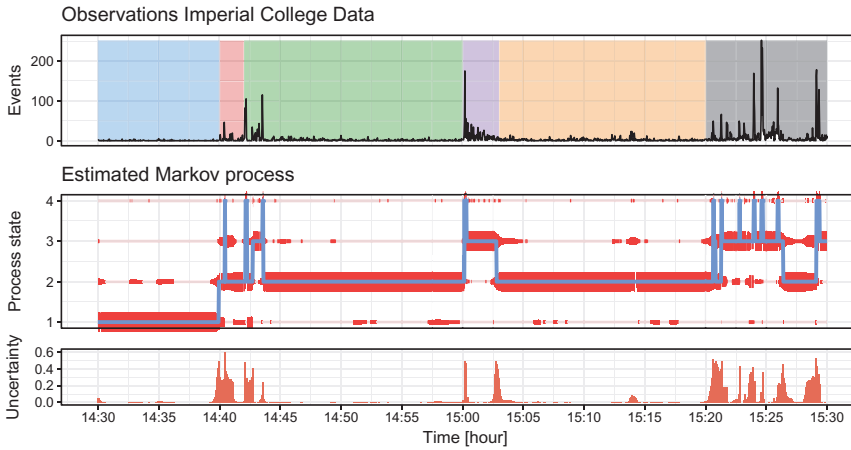

Fig. 5: Excerpt of sampled $\{X_t\}$ for the Imperial College data. Binned flow data (top), 500 samples of $\{X_t\}$ (using the Gibbs sampler described in Section 2.3) (middle), and uncertainty of the inferred state (bottom). The blue line indicates the sample mode at each point in time, while the thickness 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 at time t_i.

Especially in comparison to the results (Figure 6) from a conventional MMPP model using the Gibbs sampler from Fearnhead and Sherlock (2006), 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 the whole data. The identified states are in good correspondence 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-spikes 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 the exact time of the state change has a high uncertainty.

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 programmes open.
- State 2 corresponds to the presence of an inactive user, i.e., the internet browser or a similar programme is open, but no actions are taken. It is

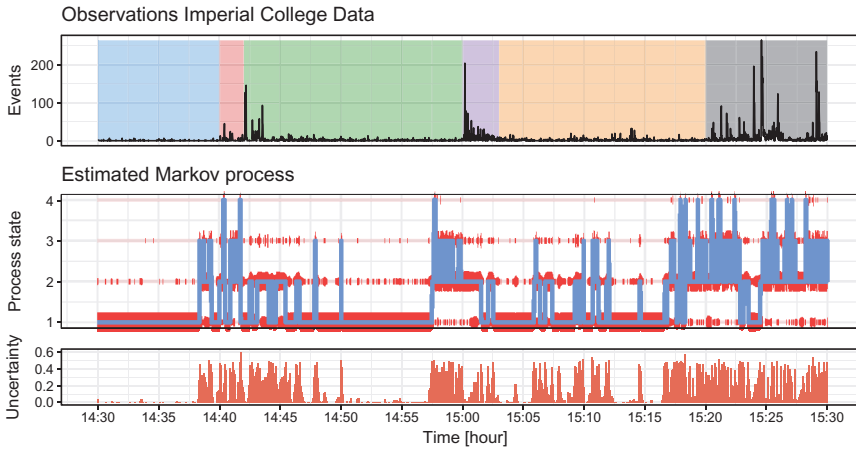

Fig. 6: Results from a conventional MMPP model. 500 samples were generated using the Gibbs sampler described in Fearnhead and Sherlock (2006).

remarkable that the watching of internet videos or ssh-communication 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 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 inference from activity on the computer to a general state of the person sitting behind the computer, we need additional assumptions and post-process modelling. All in all, there is a great correspondence of these four activity states that are sampled with our model, and the actual user activity.

3.4.1. *Convergence*

Figure 7 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 states 1 to 3 are sampled in a reasonable range, corresponding to state half-lives 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.

The posterior distributions of the generated sample chain of $\mathbf{Q}$, $\boldsymbol{\lambda}$ and λ_Y, depicted in Figure 7, all appear to be well explored with a single pronounced mode. Figure 8 depicts trace plots and auto-correlation plots of our sampled chain for selected parameters. All parameters show a fast convergence towards a stationary distribution, and a low auto-correlation, indicating a great efficiency of our sampler. The first 50 samples were discarded as burn-in.

Since the individual Markov process states are interchangeable, an important aspect to be considered is the *label switching* problem (Jasra

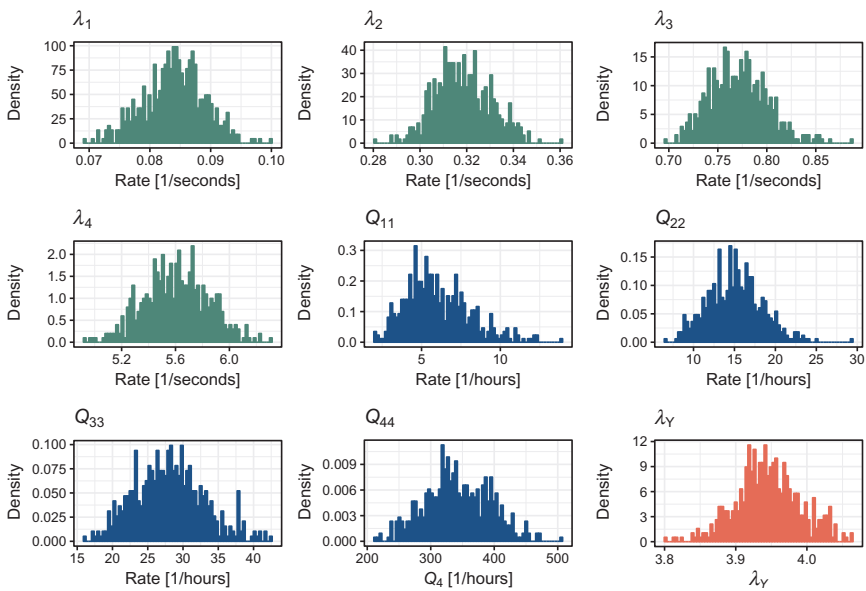

Fig. 7: Sampled posterior distributions of Q_{ii} (blue), λ_i (green) and λ_Y (red).

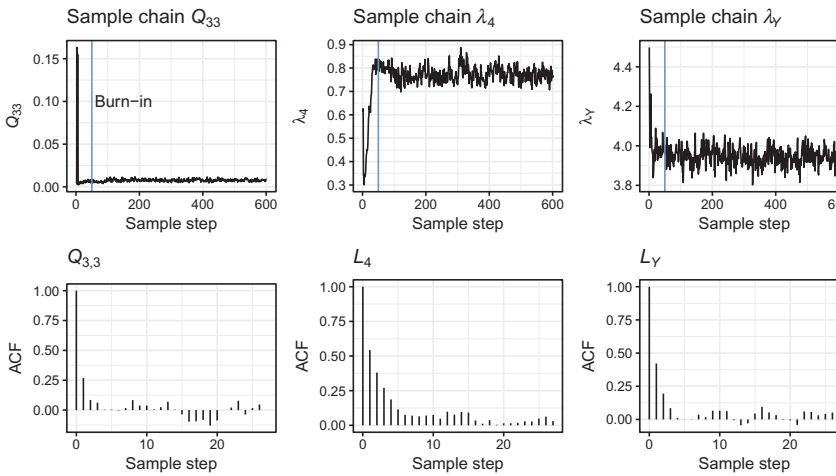

Fig. 8: Trace and auto-correlation plots for selected parameters. The blue line indicates the discarded burn-in.

et al., 2005). Since the marginal posterior distributions are well separated, we avoid this problem by ordering the initial samples of the model parameters. A more detailed discussion of this problem can be found in Clausen (2017).

4. Conclusion

In this chapter, we have established a novel Bayesian framework that is able to identify temporal patterns in the network flow generation of individual personal computers. We proposed a new hierarchical model that addresses the deviations of network flow arrivals from a Poisson process 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. We then adopted the concept of the exact Gibbs sampler of 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 with converging model parameters on multiple computers in the LANL data set. A controlled experiment which was the focus in this work verified that the identified patterns can be linked quite closely to states of human activity. In order to achieve a finer distinction of different activities with similar arrival rates, the incorporation of other flow quantities in a broader model based on MMPPs is possible. A discussion of a possible model extension including the flow durations is discussed in Clausen (2017).

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.

References

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

Clausen, H. (2017). *A Bayesian Approach to Human Behaviour Modelling in Computer Networks*, Master's thesis, Imperial College London.

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

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

Hobolth, A. and Stone, E. A. (2009). 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, pp. 1204–1233.

Jasra, A., Holmes, C. C. and Stephens, D. A. (2005). Markov chain Monte Carlo methods and the label switching problem in Bayesian mixture modeling, *Statistical Science* **20**, 1, pp. 50–67.

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

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

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

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

Park, K. and Willinger, W. (2000). *Self-similar Network Traffic and Performance Evaluation*, Wiley Online Library.

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

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

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