University of Ljubljana
Faculty of Computer and Information Science

Tadej Matek

# Anomaly detection in computer networks using higher-order dependencies

master's thesis

the 2nd cycle master's study programme
computer and information science

Supervisor: doc. dr. Lovro Šubelj

Ljubljana, 2017

Copyright. The results of this master's thesis are the intellectual property of the author and the Faculty of Computer and Information Science, University of Ljubljana. For the publication or exploitation of the master's thesis results, a written consent of the author, the Faculty of Computer and Information Science, and the supervisor is necessary.

©2017 Tadej Matek

# Contents

# Povzetek

**Naslov:** Odkrivanje anomalij v računalniških omrežjih z uporabo odvisnosti višjega reda

Dandanes poznamo neomejeno število omrežnih napadov, ki izkoriščajo ranljivosti v strukturi Interneta in omrežnih protokolov. V našem delu se lotimo problema zaznavanja anomalij v omrežnem prometu z vidika omrežne znanosti. Interakcije med različnimi omrežnimi protokoli modeliramo kot dinamiko v grafu. Prikažemo, da je običajni pristop izgradnje grafa močno omejen in neprimeren za modeliranje vzorcev v poteh, ki vsebujejo več kot dva koraka. Razvijemo metodo za zaznavanje anomalij, ki temelji na upoštevanju vzorcev višjega reda in prikažemo, da pravilno zazna poplavo UDP paketkov. Raziščemo tudi medsebojno interakcijo med omrežnimi protokoli in najpogostejše vzorce v omrežnem prometu.

## Ključne besede

*anomalije v omrežnem prometu, omrežni napadi, omrežna znanost, odvisnosti višjega reda, omrežni protokoli*

# Abstract

**Title:** Anomaly detection in computer networks using higher-order dependencies

Nowadays, countless network attacks are known, exploiting the vulnerability of network protocols and Internet topology. In our work, we tackle the problem of anomaly detection in computer communication networks from the standpoint of network analysis. We model the interactions between different network protocols as dynamics in a graph. We demonstrate that the traditional approach to constructing a graph is inadequate and fails to capture correlations in paths of length larger than two. We devise an anomaly detection procedure based on higher-order dependencies and show that it correctly identifies an UDP flood attack. We give insights into how computer communication protocols interact and what are the most common traffic patterns in the Internet.

## Keywords

*network anomalies, network attacks, network science, higher-order dependencies, Internet protocols*

# Razširjeni povzetek

## I Uvod

V magistrskem delu se osredotočimo na anomalije v omrežnem prometu, v okviru interakcije med omrežnimi protokoli na aplikacijski in transportni plasti TCP/IP modela. Zaznave anomalij se lotimo s pomočjo omrežne znanosti, relativno nove vede, ki je postala uporabna za modeliranje interakcij med entitetami v realnem svetu. Najpomembnejši prispevki omrežne znanosti so spoznanja, da si omrežja predstavljena s formalizmom grafa, delijo skupne lastnosti in izkazujejo posebne zakonitosti. Najbolj znane aplikacije omrežne znanosti so napovedovanje interakcij med entitetami v prihodnosti [1], določanje pomembnosti posameznih vozlišč v omrežjih [2, 3, 4] in odkrivanje skupin vozlišč, ki so podobno povezana z ostalimi vozlišči v omrežju [5].

Uporabo omrežnih protokolov modeliramo z grafi, kjer so vozlišča posamezni protokoli, povezave pa označujejo zaporedno uporabo dveh protokolov. Za zaznavanje anomalij in vzorcev uporabe protokolov se osredotočimo na dinamiko v omrežjih. Sama beseda "dinamika"označuje procese, ki se spreminjajo s časom in pomeni nasprotje klasičnih statičnih omrežij. Pred kratkim so raziskovalci ugotovili, da je običajna predstavitev omrežij z grafom nezadostna, saj ni zmožna opisati korelacij v poteh dolžine več kot dve. Posebno pozornost so dobile odvisnosti višjega reda, ki omogočajo bolj natančno modeliranje interakcij in dinamike v omrežjih.

# II Odvisnosti višjega reda

Pri običajnem pristopu v omrežni znanosti zaporedje podatkov oziroma lokacij modeliramo z grafom in sicer tako, da množico lokacij preslikamo v vozlišča, vsa pod-zaporedja dolžine dve (poti) pa v povezave. Tako npr. za zaporedje `A, B, A, A, B, C, B, A, B` dobimo množico vozlišč $V = \{A, B, C\}$ ter množico povezav $E = \{AB, BA, AA, BC, CB, BA, AB\}$. Vrsta del na področju odvisnosti višjega reda je pokazala, da je takšen pristop močno omejen na določenih vrstah podatkov [6, 7, 8, 9, 10]. Pomembno je upoštevati tudi odvisnosti višjega reda oziroma poti dolžine več kot dve. V kolikor za zgornjo sekvenco upoštevamo tudi poti dolžine tri, dobimo množico vozlišč $V = \{AB, BA, AA, BC, CB\}$ ter množico povezav $E = \{ABA, BAA, AAB, ABC, BCB, CBA, BAB\}$. Dobljenemu grafu pravimo tudi graf drugega reda, saj modelira interakcije dolžine tri. Opazimo, da vozlišča v takšnih grafih niso več lokacije v fizičnem smislu temveč sedaj vsebujejo podatek o zgodovini naključnega sprehajalca v grafu. Fizična lokacija $B$ ima tako npr. dve pripadajoči vozlišči drugega reda, $AB$ in $CB$.

Scholtes [9] v svojem delu predstavi splošno ogrodje za modeliranje odvisnosti višjega reda, vse do največje velikosti spomina $K$. Verjetnosti prehodov med vozlišči v omrežju višjega reda so podane v enačbi (3.1). Časovna omrežja, kjer imajo povezave tudi časovno oznako, ki označuje kdaj se povezava zgodi, se lepo navezujejo na odvisnosti višjega reda. Glavni povzročitelj odvisnosti višjega reda so korelacije v vrstnem redu obiska lokacij. Časovne oznake lahko močno vplivajo na ta vrstni red in povzročijo nenatančnost v običajnem modeliranju omrežij.

# III Zaznavanje anomalij

Enačbo (3.1) uporabimo v postopku za zaznavanje anomalij v dinamiki omrežij. Ideja zaznavanja anomalij temelji na primerjavi dveh modelov odvisnosti višjega reda. Model s krajšim spominom $M_{k-1}$ primerjamo z modelom z daljšim spominom $M_k$. V kolikor je odvisnost reda $k$ statistično značilna,

potem jo bo model s krajšim spominom napovedal kot odstopanje od trenutnih verjetnosti, shranjenih v modelu, model z zadostnim spominom pa jo bo prepoznal kot v skladu s prehodnimi verjetnostmi. Omenjen postopek lahko preprosto razširimo na zaznavanje anomalij, kjer model naučen na učnih podatkih označi vsako pot iz testnih podatkov kot bodisi v skladu z modelom ali kot odstopanje od modela.

Sam postopek preverimo nad umetno generiranimi podatki, ki vsebujejo vnaprej določene odvisnosti višjega reda. Rezultati kažejo, da postopek pravilno zazna odvisnosti višjega reda in pravilno napove odstopanje od naučenega modela.

# IV Rezultati

Razvito metodo za zaznavanje anomalij preizkusimo v omrežjih, ki vsebujejo omrežne protokole in njihovo zaporedno uporabo. Sprva določimo vlogo vozlišč v omrežju, katera so ciljna vrata v protokolih TCP in UDP. Samim vratom dodamo še oznako, ali se je ciljni IP naslov zamenjal ali ostal enak. Povezava pomeni zaporedno uporabo protokola s strani enega uporabnika v omrežju. Sprva preizkusimo podatke CAIDA [11] in ugotovimo, da je pomembno upoštevati tudi karakteristike omrežja ter kako so bili podatki pridobljeni. Odvisnosti višjega reda namreč v podatkh CAIDA niso bile prisotne zaradi premajhnega prekrivanja med potmi različnih uporabnikov.

Na podatkih iz omrežja Berkeley iz leta 1995 [12] odkrijemo odvisnosti višjega reda pri interakciji omrežnih protokolov. V tabeli 5.2 prikažemo nekaj najbolj zanimivih odkritih odvisnosti, ki so hkrati v skladu z našim razumevanjem delovanja omrežnih protokolov. Sliki 5.1 in 5.2 dodatno prikažeta kako se prehodne verjetnosti spremenijo, pri upoštevanju večje zgodovine obiska vozlišč.

Metodo zaznavanja anomalij preizkusimo nad podatki UCLA CSD [13], ki vsebujejo umetno generiran "UDP flood"napad. Sam napad poteka tako, da napadalec pošlje več velikih UDP paketov na naključna vrata žrtve, pri tem

pa ponaredi izvorni IP naslov. Rezultati so pokazali, da je napad zaznan kot odstopanje v normalnem prometu in sicer smo našli preko 700 000 neobičajnih vzorcev drugega in tretjega reda. Hkrati smo ugotovili, da model, ki upošteva odvisnosti višjega reda, bolj natančno napove odstopanja od normalnega prometa, v primerjavi z običajnim modeliranjem omrežij.

# V Zaključek

V delu smo se ukvarjali z analizo delovanja omrežnih protokolov in njihove interakcije. Zanimalo nas je predvsem kako uporabniki in naprave v komunikacijskih omrežjih uporabljajo omrežne protokole ter kateri so najpogostejši vzorci. V neposredni povezavi z najpogostejšimi vzorci v omrežnem prometu so anomalije, ki so predstavljene kot odstopanje od normalnega prometa. Z modeliranjem dinamike kot odvisnosti višjega reda smo bili zmožni pokazati ne le da je pomembno upoštevati korelacije v daljših poteh, temveč tudi zaznati preprosti "UDP flood"napad.

Naš postopek je seveda možno izboljšati, predvsem bi bilo potrebno pridobiti več uporabnih podatkov za testiranje zaznavanja anomalij. Glavna slabost modela je ta, da je potrebna velika količina učnih podatkov, v kolikor se želimo izogniti t.i. "false positive"primerom. Verjamemo pa, da je glavni prispevek dela zavedanje, da so odvisnosti višjega reda pomembne pri modeliranju in so prisotne tudi v komunikacijskih omrežjih.

# Chapter 1

# Introduction

Network science, a relatively new discipline, provides a fundamental realization that many interactions in the modern world can be described as networks. As a consequence, the methods of network science are now widely applicable in both social and natural sciences. Examples of networks that have been studied include various social networks [14], traffic networks [15], e-mail networks [16], mobile communication networks [17], citation networks [18] and even biological intracellular networks [19]. Each network may exhibit special properties and several different types of networks may be based a common set of principles. Such is for example the scale-free property, where the degree distribution of nodes in the network follows a power-law rule [20]. Also known as the 80/20 rule, the power-law rule states that roughly 20 percent of all nodes are linked by the remaining 80 percent of nodes. The highly linked nodes are known as hubs and directly affect how quickly information can travel in the network.

The most well known applications of network science include link prediction, node importance and community detection methods. Link prediction enables us to predict which nodes in the network are most likely to interact in the future (*i.e.* become linked, connected) [1]. These methods are frequently used in recommendation systems. Node importance is perhaps most well known due to the famous PageRank algorithm for ranking web pages

in search results [4]. In addition, there are numerous other methods which measure node importance [2, 3]. Last but not least, community detection allows discovery of node groups in which there is more intra- than inter-group communication, revealing clusters in the network [5].

Arguably the largest physical human made network, the Internet, has also been extensively analyzed in the field of network science. Among the results are the realizations of special topological properties of the Internet, such as link redundancy and the scale-free property [21]. However, no real analysis has yet been performed on common user traffic in computer communication networks, from a network analysis perspective. We are specifically interested in how devices in the Internet communicate and what sequences of network protocols are most frequent. In our work, we construct network representations where nodes correspond to computer-network protocols and interactions to sequential use of these protocols.

The downside of inter-connectivity which the Internet offers, is its vulnerability. Countless network attacks are known today [22], abusing the network topology of routers, switches and vulnerabilities in network protocols. Of the most common attacks in computer communication networks, DDoS (Distributed Denial of Service) attacks rank high. There are various ways to classify DDoS attacks and the majority of such attacks exploit either a specific protocol or application [23]. In general, a victim is overwhelmed with artificial network packets that arrive from a large set of infected endpoints in the network, and rendered inoperable. Intrusion detection systems (IDS) are agents in the network, which actively monitor the traffic in an attempt to detect such anomalies. Two major types of IDS systems exist. Signature-based systems use pattern-matching techniques to match an anomaly in the network with a database of pre-recorded known attacks, while anomaly-based systems build a statistical model of the normal network traffic and then attempt to detect any deviation from this normality [24]. The latter type of IDS systems may employ anything from neural networks, Bayesian networks, Markov chains/models, time series models, genetic algorithms, clustering,

outlier detection and more [25, 24]. The end goal is to learn the properties of an attack-free network traffic and then use the learned model to measure distance of anomalous traffic from the normal traffic. More novel approaches include artificial immune systems [26], which attempt to overcome anomalies in a similar way as a human body does.

To join the fields of anomaly detection and network science, we look into the dynamics of the network representing the use of computer-network protocols. The word "dynamics" points to the evolution of interactions over a period of time and is the opposite of the word "static", which represents most network representations. In the recent years, the standard network representation has been revisited. Researchers have discovered that in some cases it fails to adequately capture dynamics in the network, resulting from time-dependent processes. Specifically, it has been discovered that transition probabilities in the network are affected by the increase in the memory size of the Markov chain (higher-order chains), which represents the network dynamics [6, 7, 8, 9, 10].

The goal of our work is to provide insights into how computer network protocols are used by the endpoints in the network and how different protocols interact. We devise a procedure to detect anomalies in protocol traffic with the hope of detecting network attacks as well as uncommon traffic patterns. We highlight the importance of taking into account higher-order dynamics and its effects on model accuracy.

# Chapter 2

# Higher-order dependencies

Traditional network analysis methods use a network representation to capture the topology and interactions of entities in question. A set of nodes corresponding to entities is identified and a relation between entities is established. Two entities are in a relation, or linked, if there was an interaction between these two entities. A missing link implies no interaction. By associating transition probabilities with links, the network can be viewed upon as a Markov chain. Markov chain models the dynamic processes occurring in the real world, *e.g.* users moving between web pages using hyperlinks or passengers flying between different cities. Markov chains satisfy the *Markov property*. Considering a random walker in the network, who uses links to move between nodes, the probability of the next visited node depends solely on the current location of the walker. More formally, let the stochastic variable $X_t$ denote the location of the random walker at time $t \in [0, T]$. The location of the random walker in the future $X_{t+1}$ is given by a probability distribution:

$$\begin{aligned} Pr(X_{t+1} = x_{t+1} \mid X_t = x_t, X_{t-1} = x_{t-1}, \ldots, X_0 = x_0) = \\ Pr(X_{t+1} = x_{t+1} \mid X_t = x_t) \end{aligned} \tag{2.1}$$

The standard network representation implicitly assumes Markov chains with Markov property. Only recent research has given us insight that such

representation can be severely misleading and inacurate [10, 9, 6, 27, 28]. Consider a simple example in Table 2.1. The network representation most used is represented by the graph in Figure 2.1. To construct such a network, we identify all the nodes and then link a pair of nodes, if at least one interaction took place between those nodes in the entire dataset. Furthermore, we can assign weights to links, by summing the number of interactions between node pairs in the dataset.

| | | | |
|---|---|---|---|
| 1. | D → A | 7. | B → B |
| 2. | A → B | 8. | D → A |
| 3. | B → B | 9. | A → C |
| 4. | D → A | 10. | A → C |
| 5. | A → B | 11. | C → D |
| 6. | B → B | 12. | A → C |

**Table 2.1:** A simple example dataset of entities and their interactions.

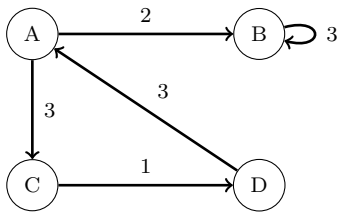



**Figure 2.1:** Constructed network representation for the dataset in Table 2.1. Note that the edges are weighted, according to the number of times an interaction took place between node pairs.

A random walker starting in node $D$, arrives to node $A$ with $Pr(X_{t+1} = A \mid X_t = D) = 1$, following from the relative frequencies of outgoing link weights at node $D$. Continuing at node $A$, the random walker will choose to move to node $B$ with probability 0.4 and to node $C$ with probability 0.6, according to relative frequencies of outgoing link weights at node $A$. However,

if we look closely at the dataset in Table 2.1, specifically at paths of length larger than one, we notice that the pattern $D \rightarrow A \rightarrow B$ is predominant. The consequence of this specific pattern is the fact that the random walker is more likely to move to node $B$ than to node $C$, if it has arrived to node $A$ from node $D$. The network representation in Figure 2.1 is inadequate, predicting that the random walker will most likely move to node $C$ from node $A$, whereas in reality, this depends on how the random walker came to node $A$ in the first place.

First-order networks, such as the one in Figure 2.1, work with the assumption that paths of length larger than one are transitive. For example, the two links in the network, $(C, D)$ and $(D, A)$ imply there is a relationship between $C$ and $A$. In reality, however, whether a relationship between $C$ and $A$ exists, depends on the ordering of links. Notice from Table 2.1, that the link $(D, A)$ occurs before link $(C, D)$, invalidating the transitivity assumption. An important observation is, that the *ordering* of links affects causality [27]. Traditional network construction discards ordering of interactions, invalidating the transitivity assumption, which most network analysis methods rely on.

Regardless of the fact that the dataset in Table 2.1 is merely a toy example, the Markov property of Markov chains implicitly modeled by traditional network representations may lead to misleading results on real world datasets, as the patterns in longer node sequences are ignored and the transitivity assumptions invalidated. Increasing the memory of the random walker, by capturing longer sequences of past node visitations, leads to higher-order Markov chains.

## 2.1 Higher-order networks

Scholtes [9] presents a general framework for modeling higher-order Markov chains. Higher-order networks are most frequently used when the input is sequential data, a sequence of node visitations or, alternatively, a multi-set

of paths observed. The main idea of a higher-order network of order $k > 1$ is that the amount of memory a random walker stores (amount of previously visited nodes) is equal to $k$:

$$\begin{aligned} Pr(X_{t+1} = x_{t+1} \mid X_t = x_t, X_{t-1} = x_{t-1}, \ldots, X_0 = x_0) = \\ Pr(X_{t+1} = x_{t+1} \mid X_t = x_t, \ldots, X_{t-k+1} = x_{t-k+1}) \end{aligned} \tag{2.2}$$

Each node in a higher-order graph $G^{(k)}$ is a higher-order state node, which contains $k$ previously visited locations. Links between such higher-order state nodes denote a shift of node visitation history by one node forward. Let us look at an example of higher-order network construction for $k = 2$, given the following multi-set $S$ of paths:

$$S = \{\ (D\ \rightarrow\ A\ \rightarrow\ B\ \rightarrow\ B),\ (D\ \rightarrow\ A\ \rightarrow\ B\ \rightarrow\ B\ \rightarrow\ B),$$
$$(D\ \rightarrow\ A\ \rightarrow\ C),\ (A\ \rightarrow\ C\ \rightarrow\ D),\ (A\ \rightarrow\ C)\ \}$$

Nodes in $G^{(2)}$ correspond to paths of length one, in contrast to a traditional first-order network, where nodes correspond to single vertices (alternatively paths of length zero). Identifying all distinct subpaths of length one given paths in $S$ yields our set of nodes in $G^{(2)}$. Adding links between successive subpaths of length one, yields our set of interactions. Again, the transition probabilities can be approximated using relative frequencies of outgoing links at a specific node. For example, $Pr(B \mid D, A) = \frac{2}{3}$ as there are exactly two paths $(D \rightarrow A \rightarrow B)$, while in total there are three paths $(D \rightarrow A \rightarrow x)$, where $x$ is any location. The resulting network represented by graph $G^{(2)}$ is visible in Figure 2.2.

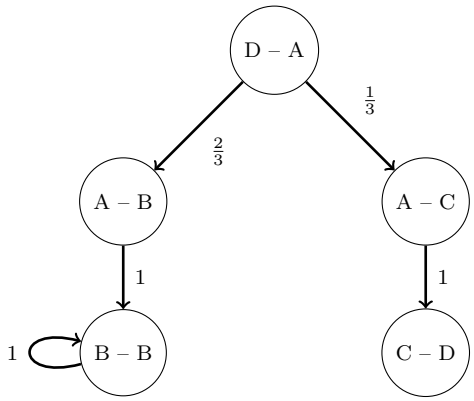



**Figure 2.2:** A higher-order network represented by a graph with higher-order nodes. Each higher-order node is a memory node with memory size of $k = 2$. Links between higher-order nodes indicate shifts in observation history by one node forward.

The traditional first-order networks are simply a special case of constructing a higher-order network by setting $k = 1$, obtaining graph from Figure 2.1. Furthermore, the dataset in Table 2.1 can be obtained by extracting all subpaths of length one from our multi-set of paths $S$.

It is important to note the distinction between state or memory nodes and actual locations. For instance, in Figure 2.2, node $A - B$ is a state node, indicating that the random walker arrived to location $B$ from location $A$. The actual location associated with state node $A - B$ is therefore $B$. Every location may have multiple associated state nodes, *e.g.* location $B$ is associated with state nodes $A - B$ and $B - B$. While first-order networks capture topological dimension of the data, as every node corresponds to an actual location, the higher-order networks capture correlations in larger sequences of paths, modeling the dynamics that occur in a network [9].

## 2.2 Variable-order networks

How do we choose $k$, the amount of memory of the higher-order network, so that the representation of the underlying dynamic processes is accurate, while keeping the complexity of our model low? It is well known that a higher-order model with order $k$ may underfit the data, while a higher-order model with order $k+1$ may already overfit the data [6]. As a result, a lot of research is conducted on the so-called variable-order networks, where a single model can contain state nodes with variable memory up to a maximum memory of $K$. This allows for more flexibility in modeling the dynamic processes, increasing accuracy of the model, while keeping the degrees of freedom of a model relatively manageable. Higher-order Markov chains suffer from the curse of dimensionality as, according to Persson *et al.* [6], their number of parameters requires exponentially increasing sizes of data to prevent overfitting. For a $k$-th order model, the adjacency matrix of a higher-order directed graph $G^{(k)}$ contains $|V|^{k+1}$ transition probabilities which need to be estimated, where $V$ is the set of locations (paths of length zero, ordinary nodes). Because the rows of such adjacency matrix need to sum to one, the total number of free parameters is $|V|^{k+1} - |V|$ (we can obtain the final transition probabilities by subtracting the rest from one). This follows from the fact that only the transitions which shift the observation history forward by one location are allowed.

There exist several approaches for constructing a variable-order network. For example, Scholtes [9] defines a so-called multi-order model $M_K$, which consists of individual higher-order models of orders $k = 0, 1, \ldots, K$. The probability of generating a path $(v_0 \to \ldots \to v_l)$ with the model $M_K$ is defined as the product of transition probabilities of subpaths with increasing order $k = 0, 1, \ldots, K$, while also taking into account that the length of path $l$ may be larger than the maximum order of the model $K$. For instance, the probability of $M_3$ generating the path $(a \to b \to c \to d \to e)$ is equal to:

$$Pr(a,b,c,d,e) = Pr(a) \cdot \ Pr(b \mid a) \cdot Pr(c \mid a,b) \cdot \\ Pr(d \mid a,b,c) \cdot Pr(e \mid b,c,d) \tag{2.3}$$

A model of order $k = 0$ can be viewed upon as specifying prior probabilities of the random walker starting it's walk in each of the locations. Scholtes then suggests a likelihood ratio optimization procedure to determine the optimal $K$ value of a multi-order model $M_K$. First, a likelihood function $L(M_K \mid S)$ is defined based on the product of probabilities of all paths in $S$. Then, choosing a model $M_{K+1}$ in favor of a simpler model $M_K$ depends on whether the likelihood of the former is significantly higher than the likelihood of the latter model. As soon as this is no longer the case, the optimal order $K_{opt}$ has been found.

Xu *et al.* [10] provide an even more compact representation of a variable-order Markov chain by storing in the variable-order network only the state nodes corresponding to higher-order correlations found in the data, rather than considering all possible transition probabilities between higher-order nodes. Using their approach, the parameter explosion resulting from $|V|^{k+1} - |V|$ transition probabilities is eliminated, as only the statistically significant higher-order correlations are considered. Their variable-order construction algorithm bears some similarity to the Apriori algorithm [29], used for mining frequent item sets. In the first stage of their algorithm, a set of rules corresponding to higher-order patterns is extracted from the sequential data. The algorithm starts out with first-order dependencies (paths of length one) and then attempts to extend them to higher correlation orders, up to a maximum order $K$. If the memory increase of a state node significantly alters outgoing transition probabilities compared to the outgoing probability distribution of the original state node, a higher-order rule is justified. The *minimum rule support* parameter allows the authors to control the compactness of the resulting variable-order network and to prevent overfitting. A low minimum support allows the algorithm to identify a large set of rules, even if the rules are insignificant, fitting the model to noise in the data. A higher minimum

support identifies only rules that occur frequently, preventing overfitting. In the second stage, the extracted rules are used to create a variable-order network representation, which encompasses both state/memory nodes and physical/location nodes.

Persson *et al.* [6] view the dynamics of the network — higher-order correlations — as flows of information. They start out with one state node for every location and then "unlump" the state nodes, producing more and more state nodes for every location, according to the largest decrease in entropy rate. Once a sufficient number of state nodes is reached, links are added among these nodes. Such sparse Markov chains have a strictly decreasing entropy rate with increasing numbers of higher-order nodes. An optimal Markov chain is selected using model selection techniques.

## 2.3 Temporal dimension

As we have seen in the previous sections, dynamic processes on a network are based on the ordering of interactions between entities. However, the dynamic processes evolve at certain time scales, which are usually much shorter than the observation period, during which the dataset was sampled [9]. Additional temporal dimension, which assigns timing information to interactions, can alter the ordering of these interactions, causing different higher-order correlations. Scholtes *et al.* provide an in-depth overview of temporal networks [27], which we summarize here. In a temporal network, each interaction is a tuple $(A, B, t)$, where $A$ and $B$ are entities in interaction, while $t \in [0, T]$ is a discrete timestamp denoting the time at which the interaction took place. A *time-respecting path* is a sequence of interactions, with increasing timestamps, *i.e.* $(A, B, t_1),\ (B, C, t_2),\ (C, D, t_3),\ \ldots,\ (X, Z, t_n);\ t_1 \leq t_2 \leq \cdots \leq t_n$. The requirement of increasing timestamps causes an ordering of the links, one that may be different from the implicit ordering provided by sequential order of interactions in the dataset. Scholtes [9] in his later work demonstrates, that the temporal dimension merely affects the ordering of the interactions

and that no change is required to the higher-order Markov chain framework. The input is still a multi-set of paths, albeit with a different ordering of nodes in paths, due to timestamps.

However, the timing information of interactions allows us to observe dynamic processes at different time scales. This can be achieved through the notion of *maximum time difference* $\delta$ for time-respecting paths [27]. A time-respecting path $(v_o, v_1, t_1),\ \ldots,\ (v_{l-1}, v_l, t_l)$ conforms to a maximum time difference $\delta$ if and only if $0 < t_{i+1} - t_i \leq \delta,\ i = 0, \ldots, l$. In other words, a time-respecting path with maximum time difference $\delta$ consists of only interactions that are maximum $\delta$ time apart. Choosing $\delta$ is equivalent to filtering out paths in the multi-set of paths $S$. A small value of $\delta$ will only keep the paths that capture dynamic processes at small time scales, while choosing a large value of $\delta$ will filter less, preserving most of the paths found in the data, including those that occur over large periods of time. The optimal choice of value for $\delta$ is a difficult problem in itself. Scholtes suggests approximating $\delta$ using inter-interaction time distribution [9].

The following set of timestamped interactions better demonstrates the role of $\delta$ parameter:

$$A \xrightarrow{1} C,\ C \xrightarrow{2} E,\ B \xrightarrow{4} C,\ C \xrightarrow{5} D,\ B \xrightarrow{7} C,\ C \xrightarrow{15} E$$

Using $\delta = 1$ filters out the paths $A \xrightarrow{1} C \xrightarrow{5} D$ and $B \xrightarrow{7} C \xrightarrow{15} E$, using $\delta = 4$ filters out only the path $B \xrightarrow{7} C \xrightarrow{15} E$, while using $\delta = \infty$ preserves all the paths.

A more subtle approach to encoding temporal information is by considering *conditional waiting times*, proposed by Matamalas *et al.* [28]. Take for instance a path $A \to A \to A \to B \to B \to B \to B$. The variable-order Markov models in the previous sections fail to account for the amount of time spent in every physical location node. One of the reasons for this is that the higher-order rules that would capture such patterns have insufficient support and are pruned. Other reasons include an insufficient maximum allowed memory size $K$ of a model. A common solution is to reduce the sequence of

node visitations to unique entries only, *i.e.* to convert the above path into $A \to B$. This however discards the temporal information implicitly provided by the number of successive location nodes.

The conditional waiting time of a node encodes additional information on how long is a random walker likely to spend at a particular location, given the previous steps the walker took. This is important in domains such as disease spreading or information diffusion. Failing to account for conditional waiting times could lead to poor predictions on the speed of diffusion processes occurring in the network [28].

Matamalas *et al.* propose a solution to this problem called *adaptive memory*. The core idea of this approach is to allow additional entries in the full transition matrix of a higher-order Markov chain, while generating subpaths by skipping successive nodes corresponding to the same location. It is best to take a look at an example. Taking the path $A \to A \to A \to B \to B \to B \to B$ and a fixed-order Markov model with $k = 2$, the adaptive memory approach would generate the following subpaths, which are later converted to higher-order nodes:

$$A \to A \to A,$$

$$A \to A \to B,$$

$$A \to B \to B,$$

$$A\cancel{B} \to B \to B,$$

$$A\cancel{B}\cancel{B} \to B \to B$$

Because of the limited memory of $k = 2$, previous location $A$ would have been lost upon reaching the subpath $B \to B \to B \to B$. Forcing the previous location to remain as the first node in the higher-order pattern allows for better predictions, while the conditional waiting time is reflected in the cardinality of the pattern, namely pattern $A \to B \to B$ which now occurs three times in total.

In terms of transition matrices, the above approach allows for additional transitions not allowed in higher-order Markov chains. A transition of $(A - B) \rightarrow (A - B)$ for example is not allowed in the transition matrix of a higher-order Markov chain, as the node visitation history is not shifted by one node forward, while this is perfectly acceptable in adaptive memory approach.

# Chapter 3

# Significant pattern detection

Chapter 2 revealed that in general, modeling higher-order dependencies of increasing order results in a combinatorial explosion. With $n = 10$ first-order nodes (physical locations), there are $10^2 = 100$ possible second-order rules, $10^3 = 1000$ third-order rules, etc., describing the dynamics of a network. To mitigate the issue, variable-order models are used which encode higher-order dependencies of various orders, specifically those found in the data. However, even a variable-order model encodes a substantial amount of higher-order patterns as it captures both the noise in the data and the important patterns. In order to obtain interesting information from the data, we desire to filter out higher-order dependencies and find only the statistically significant ones. This enables us to observe which node visitation histories significantly affect the transitions of the random walker. We seek a way to sort higher-order patterns according to their importance.

The variable-order model proposed by Scholtes [9] does not allow for individual higher-order rule discovery, but tests the whole model $M_{k+1}$ versus $M_k$. Such approach gives limited information on which specific higher-order rules are significant, *i.e.* not produced by noise in the data. A good initial proposal is that of Xu *et al.* [10], who use the Kullback-Leibler divergence measure [30] to detect significant deviations in the outgoing probability distributions caused by increased history size. Every state node in a variable-order

network has an outgoing probability distribution associated with it, as the next location depends on the transition probabilities given the current history. For instance, the state node $A-B-C$ may have the following outgoing probability distribution $p$:

$$(A-B-C) \rightarrow A : 0.12$$

$$(A-B-C) \rightarrow C : 0.49$$

$$(A-B-C) \rightarrow D : 0.39$$

Xu *et al.* detect significant higher-order patterns by increasing the history size of a specific pattern, then measuring whether there is a significant change in the outgoing probability distributions, comparing the previous and the new distributions. Increasing the history of pattern $(A, B, C)$, *e.g.* could alter the transition probabilities, producing the altered probability distribution $q$:

$$(D-A-B-C) \rightarrow A : 0.11$$

$$(D-A-B-C) \rightarrow C : 0.33$$

$$(D-A-B-C) \rightarrow D : 0.56$$

The Kullback-Leibler divergence $D_{KL}(p, q)$ measures the expected loss of information when approximating probability distribution $p$ with distribution $q$ and is defined as the expectation of the log difference between probabilities of two discrete distributions:

$$D_{KL}(p, q) = \sum_i p(x_i) \cdot \log \frac{p(x_i)}{q(x_i)}$$

Xu *et al.* [10] find the altered probability distribution $q$ significant, if

$$D_{KL}(p, q) > \frac{Order(q)}{\log_2(Support(q))},$$

where $Order(q) = 4$ is the history size of higher-order rules, while $Support(q)$ denotes the number of occurrences of a particular state node. The assumption of such approach is that patterns of high order are less likely, while the patterns which have high support (occur frequently in the data) are more likely.

Such approach also does not allow for individual higher-order rule testing, as entire outgoing probability distributions are compared. If, for example, $q$ was found to be significantly different than $p$, then all the higher-order rules in $q$ would be deemed significant, even though it is clear that pattern $(D - A - B - C - A)$ is not significant since adding node $D$ does not alter the transition probability much (0.12 versus 0.11).

Instead we use a different procedure, which relies on hypothesis testing. We use the variable-order model $M_K$ proposed by Scholtes [9], which encodes all higher-order patterns up to maximum order $K$. The probability of $M_K$ generating a specific pattern generalizes from example (2.3), but we skip the prior probability of a random walker starting in specific node ($k > 0$):

$$\begin{aligned} Pr(v_1, v_2, \ldots, v_l) = Pr(v_2 \mid v_1) \cdot Pr(v_3 \mid v_1, v_2) \cdot \ldots \cdot Pr(v_i \mid v_{i-K+1}, \ldots, v_{i-1}) \\ \cdot Pr(v_{i+1} \mid v_{i-K+2}, \ldots, v_i) \cdot \ldots \cdot Pr(v_l \mid v_{l-K+1}, \ldots, v_{l-1}) \end{aligned} \tag{3.1}$$

The intuition behind our approach is that if a higher-order pattern $(v_0, v_1, \ldots, v_K)$ of order $K$ is significant, then it should be predicted well by the model $M_K$ (pattern is in accordance with the model), but predicted poorly by a model which uses less history, $M_{K-1}$. If the pattern is predicted poorly by both $M_K$ and $M_{K-1}$ then this is a result of either an anomaly in the data, inadequate amount of training data for our model or the result of limitations in the model itself.

In order to test how well is a pattern predicted by the model $M_K$, we use hypothesis testing procedure. Assume again a pattern $(v_0, v_1, \ldots, v_K)$ of length $K$ for simplicity. The model $M_K$ outputs the total probability $p_m$ of generating this pattern, according to equation (3.1). The resulting value

states the probability of obtaining this pattern, if a random walk starting in node $v_0$ of length exactly $K$ was made in a traditional, first-order network. The length of the random walk denotes the amount of visited locations in the first-order network including $v_0$. In comparison with the model, we also have an empirical value for probability of the pattern. Let $r$ denote the total number occurrences of pattern $(v_0, v_1, \ldots, v_K)$ and let $n$ denote the total number of subpaths of size $K$ starting in $v_0$. The empirical probability is simply a relative frequency $p_e = \frac{r}{n}$, according to the data.

How well a model predicts the pattern depends on how similar are both probabilities $p_m$ and $p_e$. Because consecutive random walks in the network are independent, the total number of occurrences of $(v_0, v_1, \ldots, v_K)$ follows a binomial distribution with parameters $p_m$ and $n$. We use a two-tailed binomial test to assert whether $p_m$ deviates significantly from $p_e$:

$$H_0 : p_e = p_m$$

$$H_1 : p_e \neq p_m$$

Rejecting the null hypothesis tells us that the pattern cannot be predicted well by $M_K$. Using a two-tailed test allows us to check if a pattern is either over- or under-represented in $M_K$. Because with large values of parameter $n$ the binomial test breaks down due to increasing complexity of having to calculate the cumulative distribution function involving factorials, we replace the binomial test with a two-tailed z-test, *i.e.* we approximate the binomial distribution with the standard normal distribution. This follows from the central limit theorem, as the normal approximation is accurate for large values of $n$. The test statistic becomes:

$$z = \frac{|p_e - p_m|}{\sqrt{\hat{p}(1-\hat{p}) \cdot 2/n}} \tag{3.2}$$

where

$$\hat{p} = \frac{n \cdot p_e + n \cdot p_m}{2n} = \frac{p_e + p_m}{2} \tag{3.3}$$

The $p$-value is then calculated as $1 - ncdf(z) + ncdf(-z)$, where $ncdf$ is the standard normal cumulative distribution function. We reject the null hypothesis $H_0$ if $p$-value is below a specified significance threshold $\alpha$. The entire procedure can be iteratively expanded to test patterns of increasing order $K$, while the model $M_K$ is updated with significant patterns as we proceed.

**Algorithm 1** Significant pattern detection algorithm

**Require:** multiset of paths $S$, maximum order $K$, significance threshold $\alpha$
**Ensure:** list of significant patterns
  initialize *rules* to an empty array
  **for** $k = 2$ **to** $K$ **do**
    $S_k$ = all subpaths of length $k$ from $S$
    **for all** $(v_0, v_1, \ldots, v_k) \in S_k$ **do**
      $r$ = total number of occurrences of $(v_0, v_1, \ldots, v_k)$
      $n$ = total number of $p \in S_k$, which start with $v_0$
      $p_e = \frac{r}{n}$
      $p_{M_k}$ = probability of $(v_1, v_2, \ldots, v_k)$ according to $M_k$
      $p_{M_{k-1}}$ = probability of $(v_1, v_2, \ldots, v_k)$ according to $M_{k-1}$
      $z_k = ztest(p_e, p_{M_k}, n)$ according to (3.2)
      $z_{k-1} = ztest(p_e, p_{M_{k-1}}, n)$ according to (3.2)
      **if** $p$-value of $z_{k-1} \leq \alpha$ **and** $p$-value of $z_k > \alpha$ **then**
        add $((v_0, v_1, \ldots, v_k), p$-value of $z_{k-1})$ to *rules* array
      **end if**
    **end for**
  **end for**
  sort *rules* according to increasing $p$-values
  **return** *rules*

Algorithm 1 has an added advantage of returning a sorted list of significant patterns according to increasing $p$-values. The lower the $p$-value, the more certain we are in rejecting $H_0$ and accepting a pattern as significant.

## 3.1 Artificial data

We now test the procedure described in Algorithm 1 on generated, artificial trajectories. We use a simple directed graph represented by a square $n \times n$ grid as the basis for the random walks, which generate the trajectories to be used as input.

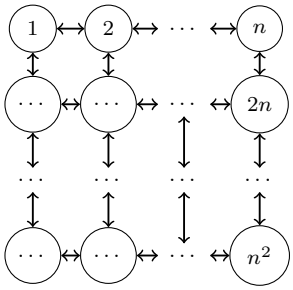



**Figure 3.1:** A directed graph represented by a square $n \times n$ grid.

We use the directed graph presented in Figure 3.1 to perform 10 000 random walks, each of length $K + 10$, based on a model giving transition probabilities. With the model we can define a set of higher-order dependencies by altering the transition probabilities based on specific history of visited nodes.

We focus first on $n = 2$, *i.e.* a $2 \times 2$ square grid, and use a model with two second-order dependencies, with no other higher-order dependencies, as visible in (3.4).

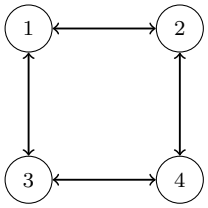



**Figure 3.2:** A $2 \times 2$ grid with two second-order dependencies.

$$\begin{array}{cc} 1 \to 2 : 0.5 & 1 \to 3 : 0.5 \\ 2 \to 1 : 0.5 & 2 \to 4 : 0.5 \\ 3 \to 1 : 0.5 & 3 \to 4 : 0.5 \\ 4 \to 2 : 0.5 & 4 \to 3 : 0.5 \\ (2,1) \to 2 : 0.1 & (2,1) \to 3 : 0.9 \\ (4,2) \to 1 : 0.1 & (4,2) \to 4 : 0.9 \end{array} \tag{3.4}$$

Table 3.1 shows the returned significant patterns by running Algorithm 1. The results are based on a search for up to fifth order patterns ($K = 5$). As it is clearly seen, the algorithm correctly identified that there are no patterns of order higher than two. Furthermore, the second-order patterns $(4, 2, 1)$ and $(2, 1, 2)$, which are under-represented in the model (3.4), have the lowest $p$-value, while the patterns which are over-represented in the model, $(2, 1, 3)$ and $(4, 2, 4)$, are also in the list of significant patterns.

It is interesting to explore why did the model find patterns $(1, 2, 1)$ and $(1, 2, 4)$ significant, as they were not specifically encoded in the model (3.4). If we break up the patterns into the probability product predicted by the model $M_1$ (according to equation (3.1)), we find the following probabilities: $Pr(1, 2, 1) = Pr(2 \mid 1) \cdot Pr(1 \mid 2)$ and $Pr(1, 2, 4) = Pr(2 \mid 1) \cdot Pr(4 \mid 2)$. From model (3.4), we find that $Pr(1 \mid 2) = 0.5$. However, in the empirical probabilities calculated from the generated trajectories, we find that $Pr(1 \mid 2) = 0.4$, causing a difference in probabilities, a low $p$-value and therefore the rule is marked as significant. The reason for such deviation from the model is the higher-order dependency rule $(2, 1) \to 2 : 0.1$ encoded in the model, which causes the random walker to use the first-order connection $1 \to 2$ much less than it normally would. It is apparent that higher-order rules affect lower-order rules as well.

| Pattern | $p$-value |
|---|---|
| $(4, 2, 1)$ | $1.01 \cdot 10^{-282}$ |
| $(2, 1, 2)$ | $7.75 \cdot 10^{-261}$ |
| $(1, 2, 1)$ | $8.17 \cdot 10^{-166}$ |
| $(1, 2, 4)$ | $1.40 \cdot 10^{-117}$ |
| $(2, 1, 3)$ | $2.42 \cdot 10^{-109}$ |
| $(4, 2, 4)$ | $4.89 \cdot 10^{-100}$ |
| $(3, 1, 2)$ | $1.84 \cdot 10^{-58}$ |
| $(3, 1, 3)$ | $3.61 \cdot 10^{-51}$ |

**Table 3.1:** A list of significant patterns and their $p$-values, returned by Algorithm 1, with $K = 5$, $\alpha = 0.01$ and 10 000 generated trajectories based on model (3.4) and a $2 \times 2$ grid from Figure 3.2.

We now look at a second example, a model with a single fifth-order dependency, as given in (3.5), with the same $2 \times 2$ grid as in Figure 3.2. Table 3.2 shows how $p$-values of pattern are increasing with increasing history size, as anticipated. Model $M_1$ has the lowest $p$-value as it has the lowest history size and is unable to account for higher-order correlations. Model $M_4$ still falls short of the $\alpha = 0.01$ significance level, as the lack of fifth node in observation history critically affects the transition probabilities.

$$\begin{array}{cc} 1 \to 2 : 0.5 & 1 \to 3 : 0.5 \\ 2 \to 1 : 0.5 & 2 \to 4 : 0.5 \\ 3 \to 1 : 0.5 & 3 \to 4 : 0.5 \\ 4 \to 2 : 0.5 & 4 \to 3 : 0.5 \\ (3,1,3,1,2) \to 1 : 0.1 & (3,1,3,1,2) \to 4 : 0.9 \end{array} \tag{3.5}$$

From this example it is evident that higher-order dependencies affect the significance of lower-order dependencies as well. Algorithm 1 finds patterns $(1, 3, 1, 2, 4)$, $(3, 1, 2, 4)$ and $(1, 2, 4)$ significant in addition to the fifth-order pattern $(3, 1, 3, 1, 2, 4)$. Notice how the lower-order patterns are suffixes of the original fifth-order pattern. The reason for lower-order patterns being

significant in addition to the fifth-order pattern is the final transition $2 \to 4$, which has an extreme transition probability of 0.9 according to the model (3.5), in the fifth-order rule. This extreme probability affects the transitions of the random walker and because all suffixes of the pattern $(3, 1, 3, 1, 2, 4)$ also take into account the transition $2 \to 4$, they are found significant. If, for instance, we change the transition probabilities of the model (3.5), so that $(3, 1, 3, 1, 2) \to 1 : 0.3$ and $(3, 1, 3, 1, 2) \to 4 : 0.7$, Algorithm 1 no longer finds lower-order patterns $(3, 1, 2, 4)$ and $(1, 2, 4)$ significant, because the fifth-order dependency no longer has a drastic effect.

| Model | $p$-value of $(3, 1, 3, 1, 2, 4)$ |
|---|---|
| $M_1$ | $4.81 \cdot 10^{-39}$ |
| $M_2$ | $6.12 \cdot 10^{-34}$ |
| $M_3$ | $8.89 \cdot 10^{-28}$ |
| $M_4$ | $1.68 \cdot 10^{-13}$ |
| $M_5$ | 0.9588 |

**Table 3.2:** The increasing $p$-values of models with increasing memory size, up to the size of pattern, $K = 5$. Based on model (3.5) and a $2 \times 2$ square grid from Figure 3.2.

## 3.2 Anomaly detection

The procedure described in Algorithm 1 can be looked upon as the training phase, in which we fit a variable-order model $M_K$ of maximum order $K$ to the trajectory data. The added bonus of the training phase is the identification of higher-order patterns that significantly affect the transition probabilities. But what if we are given a new set of trajectories, namely a testing set of data and are asked what patterns are poorly predicted by the current model $M_K$? Such questions are common in anomaly detection, where a model is first trained on a large set of training data, to capture the most common traffic patterns, and is then given an alternative data which may contain

patterns of different frequencies, resulting in an anomaly.

The anomaly detection procedure is quite simple and similar to that of Algorithm 1. We again perform statistical testing if the empirical probability of specific pattern deviates significantly from probability predicted by the model, which in-turn depends on the training phase. The entire procedure is summarized in Algorithm 2.

**Algorithm 2** Anomaly detection algorithm

**Require:** testing set of paths $S_{test}$, model $M_K$ from the training phase, significance threshold $\alpha$

**Ensure:** list of significant patterns, *i.e.* anomalies

```
initialize anomalies to an empty array
for k = 2 to K do
    S_k = all subpaths of length k from S_test
    for all (v_0, v_1, ..., v_k) ∈ S_k do
        r = total number of occurrences of (v_0, v_1, ..., v_k)
        n = total number of p ∈ S_k, which start with v_0
        p_e = r/n
        p_{M_K} = probability of (v_1, v_2, ..., v_k) according to M_K
        z = ztest(p_e, p_{M_K}, n) according to (3.2)
        if p-value of z ≤ α then
            add ((v_0, v_1, ..., v_k), p-value of z) to anomalies array
        end if
    end for
end for
sort anomalies according to increasing p-values
return anomalies
```

We perform an experiment on two different models giving transition probabilities for the random walker on a $2 \times 2$ grid from Figure 3.2. The first model encodes a second-order rule of $(2,4) \to 2 : 0.1$ and $(2,4) \to 3 : 0.9$, while the second model encodes a second-order rule of $(1,3) \to 1 : 0.1$ and

$(1, 3) \to 4 : 0.9$. We use both models to produce 10 000 trajectories of size 12 each. We use the first 10 000 trajectories from the first model as the training set and the second 10 000 trajectories from the second model as the testing set.

| Anomaly | $p$-value |
|---|---|
| $(1, 3, 4)$ | 0.0 |
| $(2, 4, 2)$ | 0.0 |
| $(2, 4, 3)$ | 0.0 |
| $(1, 3, 1)$ | 0.0 |
| $(4, 2, 4)$ | $2.37 \cdot 10^{-138}$ |
| $(3, 1, 2)$ | $1.24 \cdot 10^{-123}$ |
| $(4, 2, 1)$ | $1.13 \cdot 10^{-121}$ |
| $(4, 3, 4)$ | $4.05 \cdot 10^{-109}$ |
| $(3, 1, 3)$ | $1.00 \cdot 10^{-106}$ |
| $(4, 3, 1)$ | $5.97 \cdot 10^{-101}$ |
| $(3, 4, 3)$ | $5.66 \cdot 10^{-99}$ |
| $(3, 4, 2)$ | $1.24 \cdot 10^{-88}$ |

**Table 3.3:** A list of anomalies and their $p$-values, returned by Algorithm 2, with $K = 2$ and $\alpha = 0.01$.

Table 3.3 displays the resulting anomalies found by Algorithm 2. As expected, patterns $(1, 3, 4)$ and $(1, 3, 1)$ are anomalies because they are encoded in the second model used to generate testing data, while patterns $(2, 4, 2)$ and $(2, 4, 3)$ are also anomalies because they are not encoded in the second model producing the testing data, but were present in the first model producing the training data. Other anomalies are the direct result of the first four anomalies, as the second-order patterns affect transition probabilities of first-order patterns as well.

When using the training data as the testing data, no anomalies are found, which is in accordance with our expectations.

# Chapter 4

# Computer communication networks

Chapter 2 and Chapter 3 have defined higher-order dependencies, gave us a way to find important dynamics in the vast datasets and are in theory ready to be applied to any desirable sequence-like data for which we seek to understand dynamics. However, in practice, we are often faced with additional problems such as noisy data, missing values, too much data or the the dataset does not exhibit higher-order correlations at all. Before we dive into computer network traces, it is vital to understand what we can expect from such datasets. In this chapter, we review the basics behind the two most used networking protocols to this day, and give an overview of the type of dynamics that arise from computer communications.

Our choice of domain are computer networks, more specifically the packet traces in the transport and application layers of the TCP/IP model. We focus on segments from the two most used protocols in the transport layer, the Transfer Control Protocol (TCP) and the User Datagram Protocol (UDP). Regardless of the protocol, the transport layer encapsulates traffic from the application layer and provides a logical communication link between two endpoints in a network. The link is termed logical because the underlying network structure and topology, consisting of routers, switches and links, is

abstracted away. Both endpoints communicate as if they were connected directly, ignoring the details of the lower network layers in the TCP/IP model. The fundamental task of the transport layer is to extend the host-to-host communication, which the network layer (along with the IP protocol) provides, to process-to-process communication of the processes running on individual hosts [31]. An additional task of the transport layer protocols is to perform multiplexing and demultiplexing operations, forwarding messages from a single network interface to different processes and vice-versa. This is achieved through the use of a 16-bit identifier called port or socket number, assigned to each of the running processes. The first 1024 port numbers are reserved and termed *well-known port numbers*. Although all transport layer data traces contain only TCP and UDP segments, it is possible to determine the application layer protocol encapsulated in the segment simply by observing well-known port numbers. For instance, the port number 80 corresponds to the HTTP protocol. With port numbers larger than 1023 the application layer protocol can no longer be reliably determined.

Whereas the transport layer provides a logical communication link between processes running on individual hosts, the network layer provides a logical communication link between individual hosts in the global network structure, abstracting away the switches and physical links that connect the hosts. The IP protocol, which assigns a unique 32-bit identifier to each host termed the IP number (for simplicity, we focus specifically on the IPv4 protocol), allows for addressing of devices in the network structure (either local or global). Thus, a network trace that is captured at one of the hosts in the network is expected to have at least the following attributes: source IP address, source port number, destination IP address and destination port number. The source identifiers are necessary for bidirectional communication.

Figure 4.1 depicts the complexity that naturally arises from the layered architecture of computer communication networks. There are several variables to consider for a network trace dataset, before attempting to use network analysis and higher-order methods. From the network layer perspec-

tive, an important consideration is where the tracing program was placed, *i.e.* on which host (IP address). For example, we can monitor the intra-network communication (communication between hosts `123.218.44.78` and `123.218.44.209` in Figure 4.1), the inter-network communication (communication between hosts `123.281.44.5` and `46.164.13.66` in Figure 4.1) or both. Most datasets we have tested are the result of a trace being run on the edge router between the sub-network and the rest of the Internet (inter-network communication). Such traces are also the most interesting, as the majority of application protocols run across the Internet rather than in sub-networks (e-mail and web pages). Such traces also make the most sense, as network attacks most frequently originate from outside the sub-network.

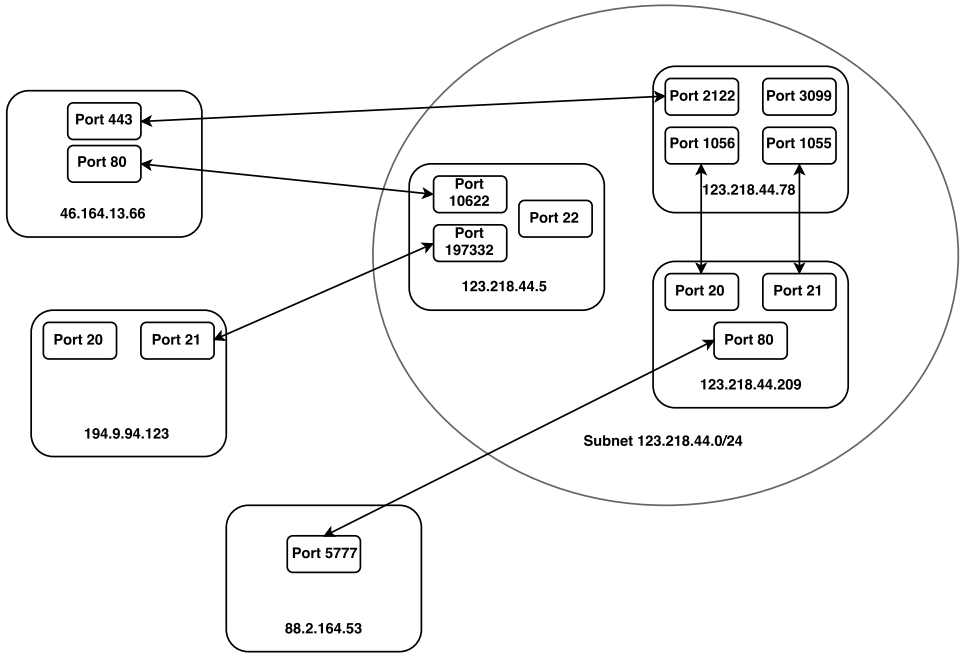



**Figure 4.1:** An example of a computer network architecture, depicting the process-to-process communication, while also outlining the significance of IP addresses. The sub-network uses public IP addresses, ignoring details such as network address translation (NAT), to keep complexity to a minimum.

A second variable to consider is how to interpret the network trace data from the network analysis standpoint, *i.e.* what would be the nodes in the network and what the relations between those nodes. We found the following interpretation satisfactory. A node corresponds to a particular packet destination in the dataset. Clients, or packet sources in the dataset, perform a walk in the network of such nodes, producing trajectories of packet destinations (clients repeatedly choose different packet destinations to send packets to). Two nodes are connected or in a relation, if any client used them in succession (in a traditional, first-order network). The definition of a node is intentionally vague, because a packet destination can be defined in multiple ways. The simplest approach is to treat all destination IP addresses as packet destinations and therefore nodes. Another approach is to use port numbers, and thereby application layer protocols, as packet destinations, or a combination of both. As the majority of publicly available datasets use some form of IP address anonymization procedure and because the IP address itself gives almost no valuable information on the type of traffic, we decided to focus on port numbers and protocols instead. Using a combination of IP address and destination port is also an option, but is problematic, because there is little overlap in the destination IPs used by different clients. Clients are likely to use their own set of destination IP addresses that does not overlap with other clients, *e.g.* every website is hosted on a web server with it's own IP address and every client is more likely to browse it's own unique set of web sites. Even a single website can have multiple associated IP addresses to help with load balancing. Completely disregarding IP addresses however, is also not an option, because they do provide one important piece of information. Observing two consecutive connections of one client allows us to detect whether a destination host has changed or remained the same. In our final implementation, we use the information from two consecutive IP address identifiers and append it to the protocol/port number identifier. If a protocol was used on the same host it is marked as `protocol*`, while if the destination IP address was changed, no `*` identifier is present.

To better illustrate our choice of node and relation interpretation, consider the following example dataset in Table 4.1, which is a subset of TCP traffic trace between the Lawrence Berkeley Laboratory and the rest of the world [32]. In this example, packet sources or clients can be either the `Source IP` idenfitier or a combination of `(Source IP, Source port)` identifiers. We have found the `Source IP` identifier as the packet source to be adequate, as otherwise the trajectories are broken up into much smaller paths. The packet sources themselves are merely random walkers and are not translated into a network representation.

| Source IP | Source Port | Destination IP | Destination Port |
|---|---|---|---|
| 45 | 1262 | 46 | 25 |
| 196 | 1852 | 197 | 23 |
| 120 | 1728 | 448 | 80 |
| 544 | 20 | 543 | 40583 |
| 45 | 25 | 140 | 1547 |
| 45 | 1399 | 705 | 25 |
| 544 | 20 | 543 | 41617 |
| 120 | 1766 | 121 | 25 |
| 120 | 1816 | 1026 | 25 |
| 45 | 1518 | 1170 | 25 |
| 45 | 1563 | 1277 | 25 |
| 45 | 25 | 1220 | 1284 |
| 196 | 25 | 170 | 3366 |
| 120 | 1997 | 1262 | 80 |
| 120 | 2031 | 1262 | 80 |
| 45 | 25 | 1 | 4664 |

**Table 4.1:** An example network trace dataset to illustrate different possible interpretations of nodes and edges from a network analysis standpoint. IP addresses have been anonymized and replaced with integer identifiers.

Like previously mentioned, there are several options for packet destinations or nodes in the first-order network. We group the dataset in Table 4.1 according to the packet source (`Source IP` column) and convert the packet destinations of every packet source into a sequence-like data or trajectories. The ordering of individual packet destinations within each trajectory depends on the original ordering in the dataset, which may be arbitrarily changed by permuting the rows of the dataset (or by sorting the dataset based on a timestamp column). For example, the following trajectories are obtained if we choose `Destination port` as our packet destination:

```
45: 25, 1547, 25, 25, 25, 1284, 4664
196: 23, 3366
120: 80, 25, 25, 80, 80
544: 40583, 41617
```

The trajectories can then be used to produce first-order and higher-order graphs as well. An example of first-order graph is visible in Figure 4.2.

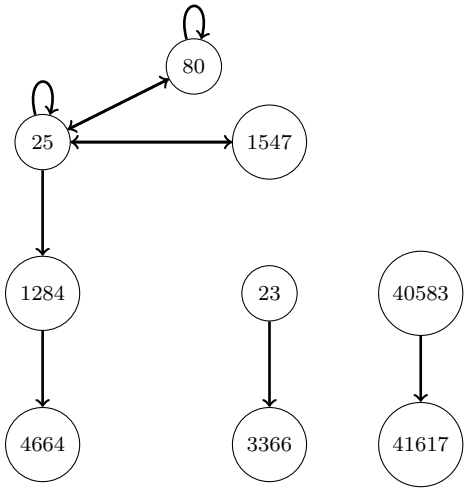



**Figure 4.2:** A first-order network constructed from example dataset in Table 4.1 by using `Destination port` as packet destination.

Notice that using only `Destination port` column as the packet destina-

tion leads to loss of information. Specifically, the sub-sequence `25, 80, 80` of packet source `120`, corresponding to repeated use of `HTTP` protocol, does not reveal if the protocol was used to communicate with one host or with different hosts. By also taking into account the `Destination IP` column, an alternative set of trajectories is obtained, one that allows us to observe if the hosts have changed (`*` marks same hosts):

```
45: 25, 1547, 25, 25, 25, 1284, 4664
196: 23, 3366
120: 80, 25, 25, 80, 80*
544: 40583, 41617*
```

Rarely are we provided with a ready-to-use dataset such as the one in Table 4.1. TCP protocol, for instance, segments the data and sends it in chunks during a single session. Furthermore, every received segment must be acknowledged, resulting in an overflow of packets that have to be analyzed. As a result, raw TCP trace data contains all packets, both those used to initiate a session and those used during the session. We are of course interested only in successive protocols being used, so we keep only the packets related to session initiation. UDP protocol has no such issue, as it sends the data as-is (the application layer has to take care of segmentation). Specific quirks in the application layer protocols are captured as dynamics, represented by higher-order correlation. For instance, in Figure 4.1, clients `123.218.44.78` and `123.218.44.209` communicate using FTP protocol's *active mode*, but this is well captured by modeling dynamics, removing the need to handle such edge cases manually.

Last but not least, we have to consider that the type of communication network may not exhibit higher-order correlations at all. Such was the case when we used our significance detection framework on anonymized internet traces from CAIDA's *equinix-chicago* monitor on high-speed Internet backbone links [11]. As the transport layer protocols are point-to-point logical connections and because we opted for a network representation using destination port numbers as nodes, the presence of recurring paths of length larger

than one depends not only on where the traffic monitor is placed, but also on how much overlap there is among different trajectories of packet sources. In the case of CAIDA, the traffic monitor was placed on a high-speed backbone link, causing not only an enormous amount of traffic, but also the type of traffic was mostly forwarded packets. This resulted in an insignificant overlap between trajectories of different packet sources and consequentially no higher-order correlations. If we are to have any hope of detecting higher-order correlations in computer communication networks, we should focus on consumer sub-networks, rather than on key architectural endpoints in the global internet structure.

# Chapter 5

# Results

Chapter 4 outlined the type of datasets we are looking for within the domain of computer communication networks. We are interested in TCP and UDP packet traces of consumer-oriented networks and we are looking for data traces which were taken at a border router of the sub-network. We first present our results of higher-order network analysis on a TCP trace provided by Paxson [12]. The dataset contains a wide-area monthly trace of all TCP packets between the Lawrence Berkeley Laboratory and the rest of the world. The tracing program ran between September 16, 1993 through October 15, 1993. The dataset is quite old and not really representative of the current dynamics in the global Internet structure. Back in 1993, internet communication was not yet widely spread and security did not yet pose an issue. Consequentially, a lot of insecure protocols such as Telnet were widely used, while more modern and secure protocols such as SSH were just emerging. A lot of old protocols found in this dataset are nowadays obsolete, such as the old Line Printer Daemon (LPD) protocol that was replaced by the Internet Printing protocol (IPP) or the Gopher protocol superseded by the HTTP protocol. Nonetheless, it is useful to analyze the significant higher-order patterns found by our framework, as we know what to expect in terms of dynamics for every protocol. However, in order to understand the dynamics found by our framework, we first require specific knowledge of the domain.

We therefore give a brief overview of the most used protocols of pre 2000 era in Table 5.1.

| **Protocol (associated ports)** | **Description** |
|---|---|
| Telnet [33] (23) | Remote terminal access. Used to connect to remote hosts and run command line commands remotely. Uses no encryption by default. |
| X window system (6000-6063) | Remote graphical user interface protocol. A server controls the hardware (display, keyboard, mouse), while clients connect to the server and issue graphical commands such as displaying a window, drawing on screen etc. |
| Finger [34] (79) | Provides status report and user information sharing of remote systems. The protocol was primarily used to check online status of people on remote machines and to exchange simple information such as name and e-mail addresses. |
| Remote login [35] (513) | Almost equal to Telnet protocol, allows users remote terminal access. Unlike with Telnet, the server can specify a list of trusted clients and those clients require no authentication. |
| Internet Relay Chat [36] (194, 6667) | Text chat based on client-server architecture. A group of clients connect to a chat server to exchange text messages. |
| Gopher [37] (70) | Known as the predecessor of the modern web, it provided online access to documents using a hierarchical directory structure. |

**Table 5.1:** An overview of the most frequently used application-layer protocols before the year 2000, as observed in Lawrence Berkeley traces [12]. Note that all of the protocols use TCP as the underlying transport-layer protocol. Some of these protocols are still widely used today.

## 5.1 Berkeley data set

We now provide the results of higher-order pattern detection of maximum order up to $K = 5$ on the Lawrence Berkeley TCP traces [12]. We list a few most interesting higher-order patterns which were found to be significant, as seen in Table 5.2.

| Pattern | Description |
|---|---|
| `ftp, ftp-data*, ftp-data*` | In accordance with FTP protocol, where a control connection is established first and then the data connection takes place on the same host. |
| `finger, smtp*, finger*` | Interesting because the client first used the Finger protocol to obtain information from the remote host, which is a SMTP server at the same time. |
| `ftp, login*, ftp*` | The client started a FTP control session and also used remote login procedure to access the remote host. |
| `login, login*, X11*` | The client first used remote login to access the remote host and then established a X11 session to run graphical programs on the remote host, most likely piggybacking over existing login session. |
| `printer, smtp, ftp` | A typical example of a client using all popular services of the Internet at that time. |
| `domain, domain, domain*, domain` | An example of DNS query where three different `DNS` servers are involved. Interesting also because DNS most usually operates using UDP as the underlying transfer protocol. |
| `smtp, printer, printer*, printer*, printer*, smtp` | A client which sends two e-mails and completes a print-job on one remote host. |

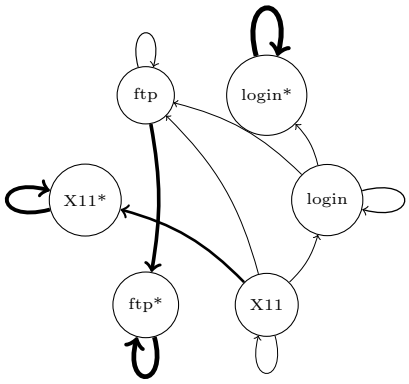
**Table 5.2:** Most interesting higher-order patterns which were found significant while investigating Berkeley TCP traces [12] ($\alpha = 0.01$). Protocols marked with $*$ identifier denote that the destination IP address remained the same. The interpretation of these patterns is only hypothetical.

We also visualize the results using graphs. Because there are many protocols (physical nodes) in the dataset and because each protocol has two cor-

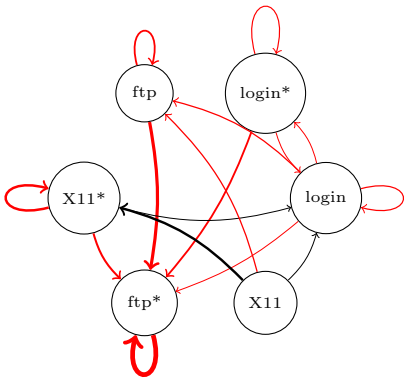
responding nodes (indicating the same or different IP address), we restrict ourselves to the following protocols: FTP, X11 and Telnet/Remote login. Furthermore, we join both the FTP control and data connections (ports 20 and 21) into a single protocol and we treat Telnet and Remote login protocols as one, due to their similarities. This greatly reduces the size of graphs we are about to present (induced subgraphs of subset of nodes), while still outlines the importance of higher-order dynamics. To further reduce the complexity we opt for the following visualization technique. We visualize each higher-order graph $G^{(k)}$ of order $k$ as a first-order graph. We do this by producing a set of first-order graphs, based on every history combination of size $k$. The transition probabilities of these first-order graphs are determined according to the specific node visitation history which the graph represents. For example, for a second-order graph $G^{(2)}$ and three most used protocols, we have five first-order graphs, $G^{(1)}_{FTP}$, $G^{(1)}_{Login}$ and $G^{(1)}_{X11}$, each representing transition probabilities for different node visitation histories. A third-order graph $G^{(3)}$ would have eighteen first-order graphs $G^{(1)}_{FTP,FTP}$, $G^{(1)}_{FTP,FTP*}$, $G^{(1)}_{FTP,Login}$ etc. Protocols used on the same remote host, *i.e.* marked with $*$, are not allowed as the first element in node visitation history. Because the number of different node visitation history combinations rises exponentially with increasing order $k$, we keep only those first-order graphs, which contain a significant higher-order pattern found in the previous step.

Figure 5.1 provides an overview of how dynamics change with the second-order dependencies in comparison with the traditional first-order dynamics. Graph in Figure 5.1(b) demonstrates how knowing that the previous used protocol was FTP results in all transitions converging back to the FTP protocol, but on the same remote host (IP address). The packet source is much more likely to use the FTP protocol again, coming from the Login and X11 protocols and, is now less likely to remain in these protocols (reduced weights on self-loops). Furthermore, you are now more likely to remain using the FTP protocol, as the weight on self-loop has been increased. A similar story can be observed in graphs from Figure 5.1(c) and Figure 5.1(d), where know-

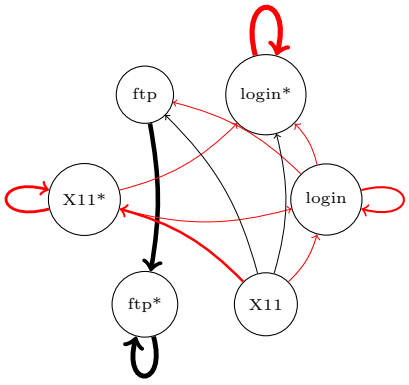
ing the previously used protocol leads to repeated use of the protocol in the future.



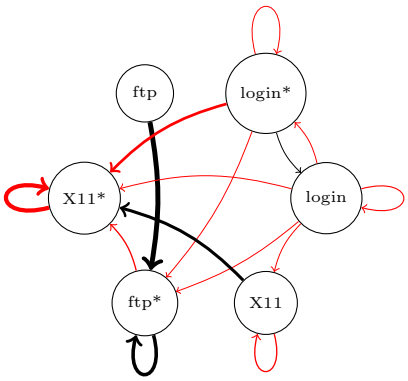

(a) Traditional, first-order graph $G^{(1)}$ comprised of the three chosen protocols. Each protocol has two nodes: one corresponding to the use of same IP address (*) and one corresponding to the use of different IP address.




(b) Graph $G^{(1)}_{FTP}$ giving an overview of how transition probabilities change when we know that the previously used protocol was FTP.




(c) Graph $G^{(1)}_{Login}$ giving an overview of how transition probabilities change when we know that the previously used protocol was Remote login/Telnet.

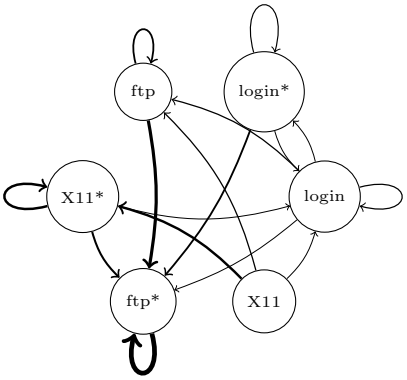



(d) Graph $G^{(1)}_{X11}$ giving an overview of how transition probabilities change when we know that the previously used protocol was X11.

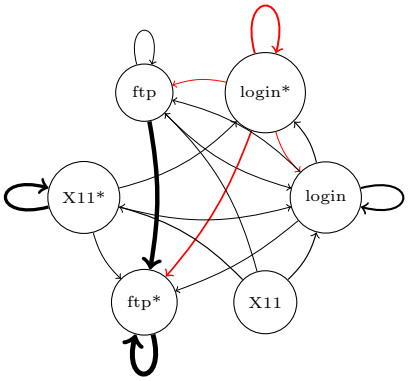
**Figure 5.1:** A comparison of the ordinary first-order graph $G^{(1)}$ versus the first-order graphs based on specific node visitation history. Transition probabilities are represented using edge widths, while edges marked in red denote the patterns that were found to be significant according to our significance detection framework ($\alpha = 0.01$). To reduce the size of graphs, edges with transition probability less than 0.05 were pruned. Full adjacency matrices are visible in Figure A.1.

We now turn our attention to third-order dependencies as depicted in Figure 5.2. Comparing graphs in Figure 5.2(a) and Figure 5.2(b) tells us that a sequence of protocols `ftp, login*, login*` is now more likely to keep you using the Remote login/Telnet protocol on the same remote host. Furthermore, the sequence `ftp, login*, ftp` is a lot less likely to use FTP protocol on another remote host, as the hosts were just changed (reduced self-loop weight on `ftp` node). Observing graphs in Figure 5.2(c) and Figure 5.2(d) demonstrates how the sequence `login, X11*, login*` is now most likely to lead you to the repeated use of X11 protocol on the same remote host (although the connection was not found to be significant). Then there are lots of minor changes in transition probabilities, such as the added self-loops on `X11` and `ftp` nodes. The only significant edge states that you are less likely to use the Remote login protocol but to continue using X11 protocol on the same remote host.

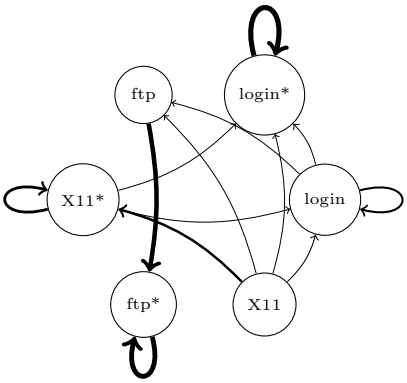
The results are based on the characteristics of the traffic in Berkeley subnetwork back in 1995. As we demonstrate in the upcoming chapter, choosing a different network with different traffic characteristics yields different higher-order dependencies. We must emphasize here that first-order dynamics fail to capture higher-order correlations, such as returning to the previously used protocol. Overall we found 372 significant second-order, 311 third-order, 184 fourth-order and 126 fifth-order dependencies, based on 460 different destination port numbers, which greatly affect the accuracy of the model.



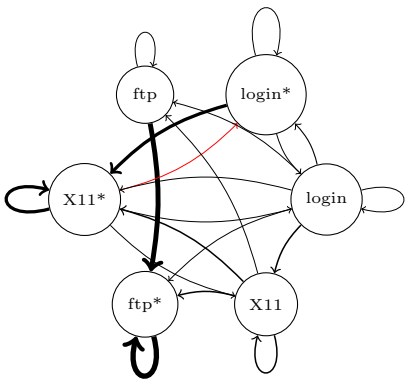

(a) Graph $G^{(1)}_{FTP}$ from Figure 5.1(b) giving an overview of transition probabilities, where the current history is the FTP protocol.




(b) Graph $G^{(1)}_{FTP,Login*}$ giving an overview of the next step in node visitation history, where the Login protocol was used additionally to the FTP protocol, on the same remote host.




(c) Graph $G^{(1)}_{Login}$ giving an overview of transition probabilities, where the current history is the Login protocol.




(d) Graph $G^{(1)}_{Login,X11*}$ giving an overview of the next step in node visitation history, where X11 protocol was used additionally to the Login protocol, on the same remote host.

**Figure 5.2:** A comparison of first-order graphs representing different node visitation history of size one versus the first-order graphs of increased node visitation history of size two. Transition probabilities are represented using edge widths, while edges marked in red denote the patterns that were found to be significant according to our significance detection framework ($\alpha = 0.01$). To reduce the size of graphs, edges with transition probability less than 0.05 were pruned. Full adjacency matrices are visible in Figure A.2.

## 5.2 UCLA data set

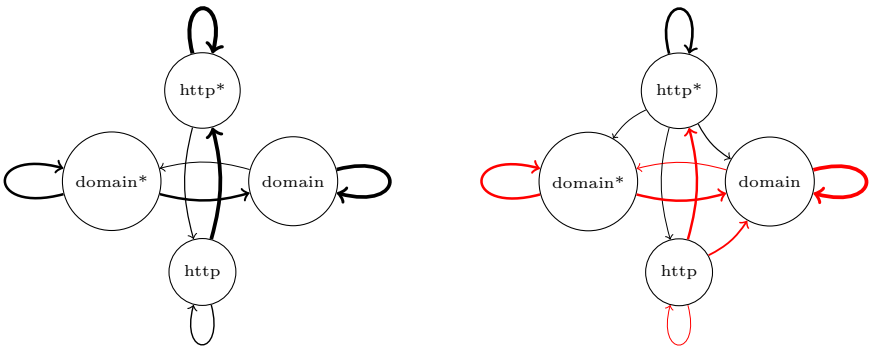
We perform similar analysis on a more recent dataset. We use both TCP and UDP traces collected on August 2001 at a border router of University of California, Computer Science Department, Los Angeles (UCLA CSD) [13]. We use three traces out of ten available, producing approximately 4.3 million connection records. The detection of higher-order dependencies of maximum order $K = 5$ reveals many interesting dependencies, as listed in Table 5.3.

| **Pattern** | **Description** |
|---|---|
| `domain, http, http*` | A classical example of DNS query which returns the IP address of a web server, which is then queried using HTTP protocol. |
| `http, http*, domain` | An example of following a hyperlink on a web site, which causes a DNS query to resolve the new domain name. |
| `http, https, https*` | An example of navigation from an insecure web site to SSL secured web site. |
| `pop3, pop3*, http` | Reading e-mails and browsing the web. |
| `domain, ssh, ssh*` | SSH session with domain name resolution. |
| `pop3, smtp, pop3` | Interaction with an e-mail client to both send and receive e-mails. |
| `imap, http, http*, imap*` | Usage of IMAP protocol for e-mail management, on the same remote host which also provides a web server. |
| `domain, domain*, domain*, domain, domain*, http` | A fifth-order dependency demonstrating a longer DNS query chain. |

**Table 5.3:** Most interesting higher-order patterns which were found significant while investigating UCLA CSD traces [13] ($\alpha = 0.01$). Protocols marked with `*` identifier denote that the destination IP address remained the same. The interpretation of these patterns is only hypothetical.

Figure 5.3 clearly demonstrates the failure of first-order modeling to capture the relationship between the DNS and HTTP protocols. Only a second-order model is able to capture the interaction between both protocols, high-

lighting the importance of higher-order dynamics.




(a) Traditional, first-order graph $G^{(1)}$ comprised of the DNS and HTTP protocols. Each protocol has two nodes: one corresponding to the use of same IP address ($*$) and one corresponding to the use of different IP address.

(b) Graph $G^{(1)}_{domain}$ giving an overview of how transition probabilities change when we know that the previously used protocol was DNS. Notice the failure of the first-order graph from Figure 5.3(a) to predict the connection between the HTTP and DNS protocols.

**Figure 5.3:** A comparison of the ordinary first-order graph $G^{(1)}$ versus the first-order graph based on DNS as the previously used protocol. Transition probabilities are represented using edge widths, while edges marked in red denote the patterns that were found to be significant according to our significance detection framework ($\alpha = 0.01$). To reduce the size of graphs, edges with transition probability less than 0.01 were pruned. Full adjacency matrices are visible in Figure A.3.

The UCLA CSD traces also contain a simulated attack scenario. The simulated attack was a UDP flood attack, where the victim is overwhelmed with large UDP packets coming to different ports. The attacker generates a series of packets, spoofing the source IP addresses. The generated packets are sent to multiple destination ports and the victim responds with an ICMP protocol message of type "Destination unreachable", for every received packet. This leads to exhaustion of resources and saturation of the network link, eventually rendering the victim unresponsive to other clients.

We perform anomaly detection procedure, as described in Algorithm 2 on the UDP traces containing UDP flood attack, using $\alpha = 0.01$. The attack dataset contains approximately 400 000 connection records. We use the trained model on the three traces described earlier as the model for normal traffic. We detect anomalies on second- and third-order subpaths. As the model for normal traffic mostly contains well-known port numbers and because UDP flood is designed to traverse as many different ports as possible, it is expected that the number of anomalies will be high. This is in fact the case. There are 390 671 anomalous second-order subpaths and 385 505 anomalous third-order subpaths. Only 3% of second-order subpaths are in accordance with normal traffic, while 4% of third-order subpaths are in accordance with normal traffic. A subset of found anomalies is visible in Table 5.4.

| Anomaly | $p$-value |
|---|---|
| `3111, 3176*, 3209*` | 0.0 |
| `1830, 2450*, 2576*` | 0.0 |
| `2261, 2524*, 2727*` | 0.0 |
| `7360, 7368*, 7452*` | 0.0 |
| `7028, 7168*, 7250*` | 0.0 |
| `5531, 5729*, 5739*` | 0.0 |
| `7997, 8037*, 8322*` | 0.0 |
| `2773, 2815*, 3063*` | 0.0 |
| ... | ... |

**Table 5.4:** A subset of second-order subpaths detected as anomalies. Each anomaly consists of random unprivileged port numbers that were used on the same remote host, which is in accordance with the UDP flood attack scenario.

There is a difference in anomaly detection using a model which captures higher-order dependencies versus a model which captures only first-order dependencies. The variable-order model is able to better capture the dynamics

found in the 4.3 million connection records used as training data and finds less anomalies than the first-order model (approximately 18 000 less). We next attempt to see how the anomaly detection handles traffic in accordance with previously seen data. This is important as in realistic scenarios attack traffic is interleaved with normal traffic and we would like to avoid false positives as much as possible. We divide the normal traffic on which the model was trained into 90% training data and 10% test data, according to increasing timestamps. We use the former for model training and the latter for anomaly detection. The results show that 41% of all second-order subpaths (63 474 in total) from test data are found to be anomalous and 36% of all third-order subpaths (71 252 in total) from test data are anomalous. The numbers are relatively high, but given many different application-layer protocols and only 3.8 million connection records as training data, this is reasonable.

Comparing variable-order model of maximum order $K = 3$ versus a first-order model, both trained on the training set of 90% data once again reveals that the variable-order model finds fewer anomalies than the first-order model (4631 less anomalies). It is clear that higher-order dependencies allow for better modeling of dynamics in the data, leading to fewer erroneous predictions on which subpaths are anomalous. In fact, the strength of difference in the set of anomalies predicted by the first-order model versus those predicted by the variable-order model could be used as a measure of how important higher-order patterns are, or to what degree the dataset exhibits higher-order correlations.

# Chapter 6

# Conclusion

The thesis has reviewed the latest research in the field of higher-order dependencies and outlined the importance of taking into account transitions of length larger than one (Chapter 2). The notion of higher-order networks was introduced, along with the notion of variable-order networks that are more economic in terms of the number of different transitions they encode. Scholtes [9] in his general framework presents an equation which the variable-order models use to encode transition probabilities. We use this equation (3.1) in our anomaly detection procedure, presented in Chapter 3. Related work has also highlighted the connection between time-based interactions and higher-order dynamics. The time dimension affects the ordering of interactions, which is the essence of higher-order correlations.

Algorithm 1 shows the proposed procedure for detecting significant individual higher-order patterns. Patterns of order $k$ are deemed significant because the variable-order model with reduced history size $M_{k-1}$ is unable to generate them with appropriate frequency, while the variable-order model $M_k$ is able to do so. The procedure is extended to allow also for anomaly detection, based on a trained variable-order model, as given in Algorithm 2. We check if our procedure is consistent on a set of artificial trajectories. The results show that the procedure correctly identifies higher-order dependencies that were artificially generated and also that dependencies of

higher-order cause appearance of lower-order dependencies. The anomaly detection procedure correctly identifies anomalies when presented with an alternative dataset.

Chapter 4 reveals the complexity of computer communication networks, resulting from the layered architecture of the TCP/IP model. Identifying the correct interpretation of nodes in our network and their interactions was a cruical step. For instance, taking into account also the IP addresses of remote hosts in addition to port numbers greatly affects the quality of our results. We also discovered that not all computer communication networks exhibit higher-order correlations. The CAIDA dataset [11] had poor overlap among trajectories of different packet sources as it was not a consumer-oriented network.

The results from Chapter 5 have reiterated the importance of modeling higher-order dynamics. Table 5.2 and Table 5.3 reveal many important higher-order patterns which cannot be captured with first-order graphs. Furthermore, these patterns also make sense, because we know how certain network protocols function and interact with other protocols. Graphs in Figure 5.1 and Figure 5.2 highlight the changes in transition probabilities caused by increased memory size. The magnitude of change in transition probabilities can be better observed in adjacency matrices in Figure A.1 and Figure A.2. Perhaps the best demonstration of the failure of first-order modeling is visible in Figure 5.3, where a first-order graph fails to capture interaction between the DNS and HTTP protocol, which we know is there.

The higher-order dependency modeling has a direct effect on the accuracy of anomaly detection. Compared with the first-order model, variable-order model detected fewer anomalies in a trace which contained a simulated UDP flood attack. The detected anomalies themselves are correctly identified attack patterns, corresponding to the repeated use of different ports on the same remote host, as per UDP flood definition.

The shortcoming of our work is in the field of anomaly detection. Because of lack of public datasets, which contain both normal and attack traffic, we

were only able to test our procedure on a single dataset. The results are optimistic as the attack traffic was isolated and not interleaved with normal traffic. We also performed an experiment where we trained the model on a 90/10 split of the normal traffic data. The results show that a relatively large number of patterns are found as false positives. The reason for this is lack of data in the training phase. This is also a weakness of anomaly detection using variable-order Markov chains. A large number of data is required to capture all variations in computer network traffic.

Future work should focus on obtaining more datasets for testing the anomaly detection procedure. We found it hard to gain access to a quality datasets, preferably taken at a border router of a sub-network. Additional datasets should give insights on how to improve our method. The method itself leaves room for acting on the detected anomalies. We merely present a procedure for detecting the anomalies. Further research should focus on how to use the results and act to stop network attacks. We believe, however, that the most important contribution of our work is the realization that higher-order dependencies cannot be disregarded and that computer communication networks do exhibit them.

# Appendix A

# Adjacency matrices for graphs

Adjacency matrices for graphs from Figure 5.1, Figure 5.2 and Figure 5.3, for clarity, as edge thickness may not be a good indicator.

| | login | login* | ftp | X11* | ftp* | X11 |
|---|---|---|---|---|---|---|
| login | 0.25 | 0.02 | 0.01 | 0.04 | 0.01 | 0.1 |
| login* | 0.13 | 0.95 | 0.0 | 0.03 | 0.01 | 0.03 |
| ftp | 0.06 | 0.0 | 0.21 | 0.02 | 0.04 | 0.05 |
| X11* | 0.02 | 0.0 | 0.0 | 0.73 | 0.0 | 0.5 |
| ftp* | 0.03 | 0.01 | 0.68 | 0.03 | 0.89 | 0.02 |
| X11 | 0.01 | 0.0 | 0.0 | 0.03 | 0.0 | 0.09 |

(a) Adjacency matrix for traditional, first-order graph $G^{(1)}$ from Figure 5.1(a).

| | login | login* | ftp | X11* | ftp* | X11 |
|---|---|---|---|---|---|---|
| login | 0.23 | 0.16 | 0.0 | 0.05 | 0.01 | 0.12 |
| login* | 0.16 | 0.22 | 0.0 | 0.04 | 0.01 | 0.0 |
| ftp | 0.24 | 0.05 | 0.34 | 0.0 | 0.04 | 0.24 |
| X11* | 0.02 | 0.02 | 0.0 | 0.47 | 0.0 | 0.46 |
| ftp* | 0.07 | 0.35 | 0.6 | 0.38 | 0.89 | 0.01 |
| X11 | 0.01 | 0.01 | 0.0 | 0.01 | 0.0 | 0.03 |

(b) Adjacency matrix for first-order graph $G^{(1)}_{FTP}$ from Figure 5.1(b).

| | login | login* | ftp | X11* | ftp* | X11 |
|---|---|---|---|---|---|---|
| login | 0.39 | 0.01 | 0.04 | 0.09 | 0.01 | 0.2 |
| login* | 0.19 | 0.97 | 0.01 | 0.12 | 0.02 | 0.07 |
| ftp | 0.05 | 0.0 | 0.04 | 0.03 | 0.02 | 0.05 |
| X11* | 0.01 | 0.0 | 0.0 | 0.54 | 0.0 | 0.47 |
| ftp* | 0.03 | 0.0 | 0.86 | 0.04 | 0.92 | 0.04 |
| X11 | 0.01 | 0.0 | 0.0 | 0.02 | 0.0 | 0.02 |

(c) Adjacency matrix for first-order graph $G^{(1)}_{Login}$ from Figure 5.1(c).

| | login | login* | ftp | X11* | ftp* | X11 |
|---|---|---|---|---|---|---|
| login | 0.21 | 0.1 | 0.0 | 0.03 | 0.02 | 0.05 |
| login* | 0.16 | 0.15 | 0.01 | 0.03 | 0.01 | 0.02 |
| ftp | 0.04 | 0.0 | 0.04 | 0.01 | 0.01 | 0.02 |
| X11* | 0.15 | 0.52 | 0.0 | 0.8 | 0.26 | 0.58 |
| ftp* | 0.06 | 0.08 | 0.93 | 0.02 | 0.67 | 0.01 |
| X11 | 0.19 | 0.01 | 0.02 | 0.03 | 0.0 | 0.25 |

(d) Adjacency matrix for first-order graph $G^{(1)}_{X11}$ from Figure 5.1(d).

**Figure A.1:** Transition probabilities marked in red indicate a significant pattern. The changes in transition probability clearly indicate how it is important to take into account higher-order dynamics.

| | login | login* | ftp | X11* | ftp* | X11 |
|---|---|---|---|---|---|---|
| login | 0.23 | 0.16 | 0.0 | 0.05 | 0.01 | 0.12 |
| login* | 0.16 | 0.22 | 0.0 | 0.04 | 0.01 | 0.0 |
| ftp | 0.24 | 0.05 | 0.34 | 0.0 | 0.04 | 0.24 |
| X11* | 0.02 | 0.02 | 0.0 | 0.47 | 0.0 | 0.46 |
| ftp* | 0.07 | 0.35 | 0.6 | 0.38 | 0.89 | 0.01 |
| X11 | 0.01 | 0.01 | 0.0 | 0.01 | 0.0 | 0.03 |

(a) Adjacency matrix for first-order graph $G^{(1)}_{FTP}$ from Figure 5.2(a).

| | login | login* | ftp | X11* | ftp* | X11 |
|---|---|---|---|---|---|---|
| login | 0.4 | 0.13 | 0.06 | 0.08 | 0.02 | 0.22 |
| login* | 0.21 | 0.37 | 0.03 | 0.15 | 0.02 | 0.11 |
| ftp | 0.09 | 0.06 | 0.08 | 0.0 | 0.01 | 0.11 |
| X11* | 0.03 | 0.01 | 0.0 | 0.62 | 0.0 | 0.22 |
| ftp* | 0.05 | 0.33 | 0.83 | 0.08 | 0.93 | 0.0 |
| X11 | 0.01 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

(b) Adjacency matrix for first-order graph $G^{(1)}_{FTP,Login*}$ from Figure 5.2(b).

| | login | login* | ftp | X11* | ftp* | X11 |
|---|---|---|---|---|---|---|
| login | 0.39 | 0.01 | 0.04 | 0.09 | 0.01 | 0.2 |
| login* | 0.19 | 0.97 | 0.01 | 0.12 | 0.02 | 0.07 |
| ftp | 0.05 | 0.0 | 0.04 | 0.03 | 0.02 | 0.05 |
| X11* | 0.01 | 0.0 | 0.0 | 0.54 | 0.0 | 0.47 |
| ftp* | 0.03 | 0.0 | 0.86 | 0.04 | 0.92 | 0.04 |
| X11 | 0.01 | 0.0 | 0.0 | 0.02 | 0.0 | 0.02 |

(c) Adjacency matrix for first-order graph $G^{(1)}_{Login}$ from Figure 5.2(c).

| | login | login* | ftp | X11* | ftp* | X11 |
|---|---|---|---|---|---|---|
| login | 0.05 | 0.11 | 0.0 | 0.08 | 0.0 | 0.0 |
| login* | 0.21 | 0.15 | 0.0 | 0.1 | 0.0 | 0.0 |
| ftp | 0.05 | 0.0 | 0.07 | 0.02 | 0.0 | 0.14 |
| X11* | 0.08 | 0.62 | 0.0 | 0.61 | 0.0 | 0.29 |
| ftp* | 0.05 | 0.04 | 0.93 | 0.03 | 1.0 | 0.29 |
| X11 | 0.29 | 0.0 | 0.0 | 0.05 | 0.0 | 0.29 |

(d) Adjacency matrix for first-order graph $G^{(1)}_{Login,X11*}$ from Figure 5.2(d).

**Figure A.2:** Transition probabilities marked in red indicate a significant pattern. The changes in transition probability clearly indicate how it is important to take into account higher-order dynamics.

| | domain | http* | domain* | http |
|---|---|---|---|---|
| domain | 0.69 | 0.0 | 0.47 | 0.0 |
| http* | 0.0 | 0.76 | 0.0 | 0.7 |
| domain* | 0.19 | 0.0 | 0.48 | 0.0 |
| http | 0.0 | 0.23 | 0.0 | 0.27 |

(a) Adjacency matrix for traditional, first-order graph $G^{(1)}$ from Figure 5.3(a).

| | domain | http* | domain* | http |
|---|---|---|---|---|
| domain | 0.7 | 0.23 | 0.47 | 0.35 |
| http* | 0.0 | 0.5 | 0.0 | 0.46 |
| domain* | 0.2 | 0.05 | 0.48 | 0.0 |
| http | 0.0 | 0.14 | 0.0 | 0.15 |

(b) Adjacency matrix for first-order graph $G^{(1)}_{domain}$ from Figure 5.3(b).

**Figure A.3:** Transition probabilities marked in red indicate a significant pattern.

# Bibliography

[1] L. Lü, T. Zhou, Link prediction in complex networks: A survey, Physica A: statistical mechanics and its applications 390 (6) (2011) 1150–1170.

[2] P. Jensen, M. Morini, M. Karsai, T. Venturini, A. Vespignani, M. Jacomy, J.-P. Cointet, P. Mercklé, E. Fleury, Detecting global bridges in networks, Journal of Complex Networks (2015) cnv022.

[3] M. Newman, Networks: an introduction, Oxford university press, 2010.

[4] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: bringing order to the web.

[5] A. Lancichinetti, S. Fortunato, Community detection algorithms: a comparative analysis, Physical review E 80 (5) (2009) 056117.

[6] C. Persson, L. Bohlin, D. Edler, M. Rosvall, Maps of sparse markov chains efficiently reveal community structure in network flows with memory, arXiv preprint arXiv:1606.08328.

[7] M. Rosvall, A. V. Esquivel, A. Lancichinetti, J. D. West, R. Lambiotte, Memory in network flows and its effects on spreading dynamics and community detection, Nature communications 5.

[8] A. R. Benson, D. F. Gleich, J. Leskovec, Higher-order organization of complex networks, Science 353 (6295) (2016) 163–166.

[9] I. Scholtes, When is a network a network? multi-order graphical model selection in pathways and temporal networks, Arxiv preprint arXiv:1702.05499.

[10] J. Xu, T. L. Wickramarathne, N. V. Chawla, Representing higher-order dependencies in networks, Science advances 2 (5) (2016) e1600028.

[11] CAIDA, The caida ucsd anonymized internet traces 2016, `http://www.caida.org/data/passive/passive_2016_dataset.xml`, [Online; accessed 4-May-2017] (2016).

[12] V. Paxson, Empirically derived analytic models of wide-area tcp connections, IEEE/ACM Transactions on Networking (TON) 2 (4) (1994) 316–336.

[13] J. Mirković, G. Prier, P. Reiher, Attacking ddos at the source, in: Network Protocols, 2002. Proceedings. 10th IEEE International Conference on, IEEE, 2002, pp. 312–321.

[14] S. P. Borgatti, A. Mehra, D. J. Brass, G. Labianca, Network analysis in the social sciences, Science 323 (5916) (2009) 892–895.

[15] G. Bagler, Analysis of the airport network of india as a complex weighted network, Physica A: Statistical Mechanics and its Applications 387 (12) (2008) 2972–2980.

[16] H. Ebel, L.-I. Mielsch, S. Bornholdt, Scale-free topology of e-mail networks, Physical review E 66 (3) (2002) 035103.

[17] J.-P. Onnela, J. Saramäki, J. Hyvönen, G. Szabó, D. Lazer, K. Kaski, J. Kertész, A.-L. Barabási, Structure and tie strengths in mobile communication networks, Proceedings of the national academy of sciences 104 (18) (2007) 7332–7336.

[18] L. Šubelj, S. Žitnik, M. Bajec, Who reads and who cites? unveiling author citation dynamics by modeling citation networks, in: Proceedings of the International Conference on Network Science, Vol. 1.

[19] A.-L. Barabási, Z. N. Oltvai, Network biology: understanding the cell's functional organization, Nature reviews. Genetics 5 (2) (2004) 101.

[20] A.-L. Barabási, Network science, Phil. Trans. R. Soc. A 371 (1987) (2013) 20120375.

[21] R. Pastor-Satorras, A. Vázquez, A. Vespignani, Dynamical and correlation properties of the internet, Physical review letters 87 (25) (2001) 258701.

[22] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. Bhattacharyya, J. K. Kalita, Network attacks: Taxonomy, tools and systems, Journal of Network and Computer Applications 40 (2014) 307–324.

[23] C. Douligeris, A. Mitrokotsa, Ddos attacks and defense mechanisms: classification and state-of-the-art, Computer Networks 44 (5) (2004) 643–666.

[24] R. Di Pietro, L. V. Mancini, Intrusion detection systems, Vol. 38, Springer Science & Business Media, 2008.

[25] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, E. Vázquez, Anomaly-based network intrusion detection: Techniques, systems and challenges, Computers & security 28 (1) (2009) 18–28.

[26] J. Kim, P. J. Bentley, U. Aickelin, J. Greensmith, G. Tedesco, J. Twycross, Immune system approaches to intrusion detection–a review, Natural computing 6 (4) (2007) 413–466.

[27] I. Scholtes, N. Wider, A. Garas, Higher-order aggregate networks in the analysis of temporal networks: path structures and centralities, The European Physical Journal B 89 (3) (2016) 1–15.

[28] J. T. Matamalas, M. De Domenico, A. Arenas, Assessing reliable human mobility patterns from higher order memory in mobile communications, Journal of The Royal Society Interface 13 (121) (2016) 20160203.

[29] R. Agrawal, R. Srikant, et al., Fast algorithms for mining association rules, in: Proc. 20th int. conf. very large data bases, VLDB, Vol. 1215, 1994, pp. 487–499.

[30] S. Kullback, R. A. Leibler, On information and sufficiency, The annals of mathematical statistics 22 (1) (1951) 79–86.

[31] J. F. Kurose, K. W. Ross, Computer networking: a top-down approach, Vol. 5, Addison-Wesley Reading, 2010.

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

[33] J. Postel, J. Reynolds, Telnet protocol specification, STD 8, RFC Editor, `http://www.rfc-editor.org/rfc/rfc854.txt` (May 1983).
URL `http://www.rfc-editor.org/rfc/rfc854.txt`

[34] D. Zimmerman, The finger user information protocol, RFC 1288, RFC Editor, `http://www.rfc-editor.org/rfc/rfc1288.txt` (December 1991).
URL `http://www.rfc-editor.org/rfc/rfc1288.txt`

[35] B. Kantor, Bsd rlogin, RFC 1282, RFC Editor, `http://www.rfc-editor.org/rfc/rfc1282.txt` (December 1991).
URL `http://www.rfc-editor.org/rfc/rfc1282.txt`

[36] J. Oikarinen, D. Reed, Internet relay chat protocol, RFC 1459, RFC Editor, `http://www.rfc-editor.org/rfc/rfc1459.txt` (May 1993).
URL `http://www.rfc-editor.org/rfc/rfc1459.txt`

[37] F. Anklesaria, M. McCahill, P. Lindner, D. Johnson, D. Torrey, B. Albert, The internet gopher protocol (a distributed document search and

retrieval protocol), RFC 1436, RFC Editor, `http://www.rfc-editor.org/rfc/rfc1436.txt` (March 1993).
URL `http://www.rfc-editor.org/rfc/rfc1436.txt`