THÈSE
présentée pour obtenir le grade de

Docteur de l'université Pierre et Marie Curie

Spécialité Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Flots de liens pour la modélisation d'interactions temporelles et application à l'analyse de trafic IP

Tiphaine VIARD

Soutenue publiquement le 28 septembre 2016 devant le jury composé de

Rapporteurs:	Pierre Borgnat	Directeur de recherches, CNRS
	Olivier Festor	Directeur de recherches, Inria
Examinateurs:	Arnaud Casteigts	Maître de conférences, Université de Bordeaux
	Marcelo Dias de Amorim	Directeur de recherches, CNRS
	Philippe Owezarski	Directeur de recherches, CNRS
	Véronique Serfaty	Responsable scientifique, DGA
Directeurs:	Clémence Magnien	Directrice de recherches, CNRS
	Matthieu Latapy	Directeur de recherches, CNRS

Acknowledgments

First and foremost, I owe my deepest gratitude to Matthieu Latapy and Clémence Magnien, who have supported me through every step of this journey with enthusiasm and scientific soundness. They taught me the basics (and the not-so-basics) of research, even though I have yet a lot to learn, I am sincerely grateful for the knowledge, time, guidance and support that they have given to me with great generosity.

My sincere thanks go to Pierre Borgnat and Olivier Festor, for reviewing my dissertation, and to Arnaud Casteigts, Marcelo Dias de Amorim, Philippe Owezarski and Véronique Serfaty for accepting the invitation to integrate the jury of my thesis defense, and for the attention you have given to my work.

All my lab mates, past and present, have contributed to fostering an enjoyable and stimulating atmosphere; Clémence, Matthieu, Fabien, Jean-Loup, Lionel, Robin, Daniel, Raphaël F.-S., Raphaël T., Noé, Thibaud, Aurore, Marwan, Keun-Woo, Louisa, Damien, Alice, Maximilien, Qinna, Élie R., Élie P., Sergey, Christophe, thank you for all the scientific meetings, the various breaks and the always-animated board game sessions. Special thanks go to Lucy, Thibaud, Hindol and Mridul, whom I had the pleasure to co-supervise, from which I have learnt way more than what I taught them; in particular, I would like to warmly thank Audrey for the 6 months spent working together, and her precious help in the final stages of this thesis.

I am immensely grateful to the members of the Fukuda Lab for welcoming me in Tokyo, and sharing your insights with me; it was a pleasure working with you; Kensuke, Romain, Johan, Benjamin, Joao, you have made my stay better than I could have imagined.

I wholeheartedly thank Nicolas Sicard, my Numerical Analysis professor, who inspired me to become a scientist, and with whom I remember the interesting, albeit long, discussions in his office.

Besides work, I am grateful to my friends: MaVi and Chloé, for the many laughs shared and your unshakeable enthusiasm, Antonio, Florent, Pierre, Pascal, Olivier, Joris, Cyril, Bruno, Nadim from the Efrei gang, for all the projects, drinks and talks we had, as well as Cécile, Megan, Helen, Harmonie, Maëva, Julien, Marcela, Marie-Laure, for the numerous good times shared together. Not to forget Tigrou, who despite being just a cat, has demonstrated furry support when (he) needed.

My family, who has supported me and given me the opportunity of an education from the best institutions; there are undoubtedly keys to all my achievements.

Fanie, it is impossible for me to conceive what my life would be without your impact. There is no key moment in my life in the last 10 years that we have not discussed together, on which you gave your advice, your support. I am immensely happy we have met.

Mel, you have been a faithful source of kindness and support ever since I met you, including during the roughest times. My thoughts of you can hardly be expressed in a few lines, yet I would like to voice confident I am in the little world we are building together.

Introduction

Interactions are everywhere: in the contexts of face-to-face contacts, emails, phone calls, IP traffic, online purchases, running code, and many others. Interactions may be directed, weighted, enriched with supplementary information, yet the baseline remains: in all cases, an interaction means that two entities u and v interact together from time b to time e: for instance, two individuals u and v meet from time b to time e, two machines on a network start an IP session from time b to time e [1], two persons u and v phone each other from time b to time e, and so on.

1. Or exchange a packet at time $b = e$.

Sequences of interactions have strong temporal and structural features, and have been widely studied in the past years. The current approach mostly consists in focusing on their structural [2] or temporal properties [3], but combining both aspects remains challenging.

2. Which entities interact with each other.
3. When and how frequently do interactions occur.

If one wants to focus on the structure of interactions, graph theory is the natural framework to do so; it provides notions such as node degree (their number of links), density (the extent at which all nodes are connected together), paths (series of links going from a node to another one), etc. This language forms the basis of network science.

If one wants to focus on the dynamics of interactions, signal processing captures time features perfectly well; one typically computes properties that may be quantified in consecutive time windows, such as the number of interactions that occur every day, the number of involved nodes, the duration of interactions, etc.

These approaches give much insight on the considered objects, and have a key advantage: graph theory and signal processing offer a huge number of advanced tools; yet, they do not capture the both structural and temporal nature of interactions over time, and induce important information losses.

Much effort has therefore been devoted to upgrade these approaches; however, these extensions are often intricate, address only one aspect of the question, still induce important losses of information.

In this thesis, we explore a new approach consisting in modelling interactions directly as link streams, *i.e.* series of quadruplets (b, e, u, v) meaning that u and v interacted from time b to time e. We will develop the basis of the corresponding formalism in Part 1.

In order to guide and assess this fundamental work, we focus on the analysis of IP traffic. It is particularly important to us that we make both fundamental and applied progress: application cases should feed our theoretical thoughts, and formal tools are designed to have meaning on application cases in the most general way.

Attacks against online services, networks, information systems, as well as identity thefts, have annual costs estimated in billions of euros. These attacks also have dramatic consequences on user trust and the reliability of services. The techniques developed in the context of these malicious activities are of ever-growing complexity, and frequently rely on subtle malware (viruses or worms). Given this context, there is a critical need of methods and tools to fight against attacks and malware diffusion, and to give an appropriate answer to the major societal questions raised.

IP traffic is typically collected at one or many intermediary points in the network. A router is set up for capture, and will keep a record of all packets going through it. One obtains a sequence of packets, each of these packets meaning that machines u and v interacted through the router at a time t.

In Part 2 we apply our framework to the analysis of IP traffic, with the aim of assessing the relevance of link streams for describing IP traffic as well as finding events inside the traffic. We devise a method to identify events at different scales, and apply it to a trace of traffic from the MAWI dataset. The high volume of traffic [4], even over short periods of time, also makes it an ideal case of application to test the scalability of our approach.

4. Many thousands of packets each second.

The work presented in this manuscript opens numerous perspectives; we summarize our contributions and present some perspectives for future work in Part 3.

Part I

A basic language for link streams

Introduction

We model sequences of interactions as sequences of quadruplets (b, e, u, v), meaning that entities u and v have interacted from time b to time e.

To study such interactions, one typically studies the properties of graphs $G_{t..t+\Delta}$ for a given Δ and for $t = 0, \Delta, 2\Delta$, etc. In other words, one splits time into consecutive slices of duration Δ and then studies the sequence of graphs obtained for each time slice. This leads to studies of how graph properties evolve with time. Others study the sequence of graphs $G_{0..t}$ for $t = \Delta, 2\Delta, \ldots$ for a given Δ. Such studies explore how graph properties evolve when the observation duration grows.

However, sequences of interactions are fundamentally different from dynamic graphs: if one were to stop time right now, some data retain meaning: for instance, the Web graph [5] can be frozen and studied at a given time t. In the case of interactions, there are typically few interactions at a given time [6], making the analysis irrelevant.

5. A graph where nodes are web pages and one places a link between two pages if one cites the other.

6. The graph induced by individuals sending an email exactly at time t is likely to have limited interest.

Many works propose to model interactions over time. Time-Varying Graphs are model-oriented: they aim at providing a model to unify these different approaches. Others, like temporal networks, study the interactions with a data-oriented point of view: they aim at answering to precise questions on datasets. Those approaches bring much progress on both fundamental and applied aspects, and we briefly review the main works in the field in Chapter 1. Progress in the area however remains limited and there is still no consensus on an appropriate approach to handle interactions over time.

Our goal in this part of the thesis is language-oriented: we aim at defining a language to deal directly with link streams, in a way similar to what graph theory does for networks, or to what signal processing does for time series; we present this framework in Chapters 2 to 6. A key goal is to make our framework as simple and intuitive as possible. We also meant it to be an extension of graph theory and signal processing for the study of interactions: a link stream that has no dynamics [7] should be exactly equivalent to a graph. Similarly, a link stream that has no structure [8] should be exactly equivalent to a time series.

7. *i.e.* where all links last all the time.

8. *i.e.* where there are only two nodes.

Finally, it is of importance that we cover many case studies and incorporate the existing state of the art as much as possible.

Chapter 1
Context

The goal of this chapter is to briefly review the main works modelling and studying interactions over time. We first quickly review methods with loss of information, and then methods that induce no loss of information.

1.1 Models inducing information loss

The simplest and most appealing way to model a sequence of interactions in order to study their structure is to resort to a graph where the nodes are the entities, and one puts a link between two nodes if they have interacted together, regardless of the number of interactions. For example, [Strogatz, 2001] explores the different ways to model complex networks of dynamical entities.

However, we focus in this short review of the literature on the ways of extending graph theory to handle interactions over time, and will not develop the rich body of work devoted to the study of interactions as static complex networks.

Given the descriptive power of graph theory, it is tempting to model sequences of interactions as sequences of graphs over time. Much work has been done in this direction, and we review the main works in this section.

It is common to study the graph $G_{t..t+\Delta}$ from time α, for a given Δ and for $t = \alpha,\ \alpha + \Delta,\ \alpha + 2\Delta$, etc. One then ends up with a sequence of graphs $\{G_i\}_{i=1..k}$, also called snapshots depending on the context. Each element G_i contains all interactions that happened between time $\alpha + i$ and time $\alpha + i + \Delta$. In those cases, one is typically interested in the statistics on a given snapshot G_i, or in comparing snapshots G_i and G_{i+1}. See Figure 1.1 for an illustration.

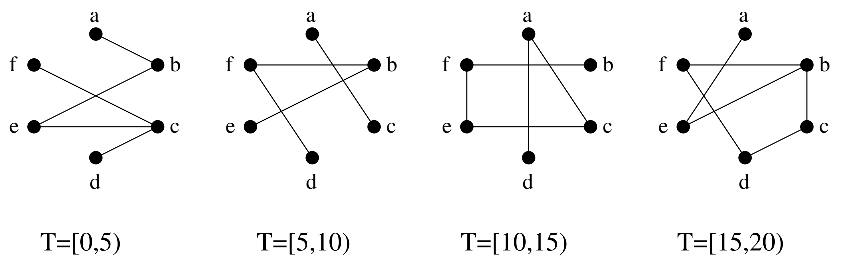


Figure 1.1: A sequence of graphs $\{G_{t..t+\Delta}\}_{t=0,\ 5,\ 10,\ 15}$ for $\Delta = 5$. The first graph of the sequence is the graph of all interactions that happened between time 0 and time 5; the second graph of the sequence is the graph of all interactions between time 5 and time 10, and so on.

Other works study the graph $G_{\alpha..t}$ at a given starting time α, for $t = \alpha + \Delta,\ \alpha + 2\Delta,\ \ldots$ for a given Δ. In other words, it is the aggregated graph of all interactions that happened over time. See Figure 1.2 for an example.

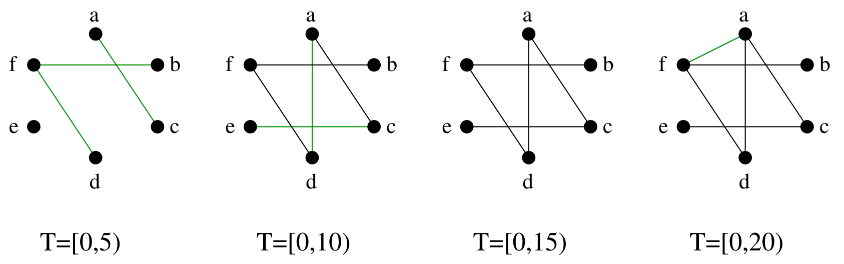


Figure 1.2: Study of the aggregated graph $G_{0..t}$ for $t = 5,\ 10,\ 15,\ 20$. The first graph of the sequence, $G_{0..5}$, contains all interactions between time 0 and time 5, the second graph of the sequence, $G_{0..10}$ contains all interactions between time 0 and time 10, and so on.

Notice that since a graph can be represented by an adjacency matrix, one can also study a sequence of matrices $\{A(t)\}_t$, and study this sequence. This approach, called 3D tensors, is used by [Gauvin et al., 2014] to track groups of nodes over time.

Sequences of graphs have been much studied: [Tang et al., 2010] extend concepts of distance and paths in sequences of graphs, and end with the conclusion that the small-world effect known for graphs also applies in time to sequences of graphs. Before them, [Leskovec et al., 2005] had studied the densification and shrinking over time in sequences of graphs.

Obviously, a key problem with these approaches is that one must choose appropriate values for Δ: too small ones lead to trivial snapshots, while too large ones lead to important losses of information about the dynamics. As a consequence, much work has been done to design methods for choosing and assessing choices in the value of Δ.

[Sulo et al., 2010] observe the evolution with Δ of common graph statistics such as the diameter or the density, and use time series tools to identify relevant time scales. [Benamara and Magnien, 2010a] produce a general methodology to assess whether a time window is large enough to characterize a given property.

In the context of mining high speed data streams, [Hulten et al., 2001a] point out the fact that the law governing the properties of data might change over time, especially over long durations. They propose a method for choosing a time window at a size where it is possible to detect such changes for given properties.

Recent work by [Léo et al., 2015] study the impact of a given time window on the resulting sequence of graphs, and show the existence of a *saturation scale*, a value of Δ such that for all $\Delta' < \Delta$, the dynamics are mostly preserved, and for all $\Delta'' > \Delta$, the properties are altered. They design an automatic method to find the saturation scale in real-world datasets.

However, in all cases, the authors merge all interactions occurring in the same slice. [Lee and Maggioni, 2011] instead consider that there is not one scale that is adapted to study a sequence of graphs, but many. They devise a framework to study a sequence of graphs at many different scales simultaneously.

Independently from the choice of appropriate values of Δ, the sequence of graphs itself is a complex object that is hard to manipulate.

1.2 Models inducing no information loss

We now turn our attention to the transformations and models of sequences of interaction that induce no loss of information.

1.2.1 *Static graphs from sequences of interactions*

In order to benefit from the tools of graph theory when studying interaction sequences, one can build a graph where the edges are labeled with their time of existence. The first work on the subject appears to be by [Berman, 1996], and [Kempe et al., 2000] are responsible for coining the term *temporal networks*. However, while this representation induces no information loss, the object itself is complicated to manipulate, and extending definitions to such labelled graphs is hard.

Instead of resorting to sequences of graphs, [Kostakos, 2009] propose to create a graph where each entity at each time is a node, and one puts a link between two nodes if they are linked in the graph, or is they are contiguous in time. See Figure 1.3 for an illustration. The links in the graph are of two natures: some of them are temporal links, and some of them are structural links. This approach, however, assumes that structural and temporal links carry similar meaning.

There are other variants, such as the one by [Queyroi, 2014], where having two nodes (u, t) and (u, t'), $t' > t$, means that $t' - t \geq \Delta$, for a given Δ.

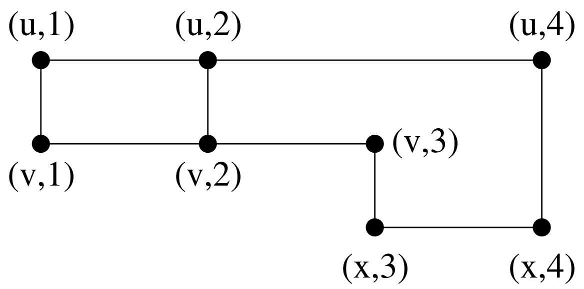


Figure 1.3: A temporal graph $G = (V, E)$, with $V = \{(u,1), (u,2), (v,3), \dots\}$, and $E = \{((u,1),(u,2)), ((u,1),(v,1)), \dots\})$. $((u,1),(u,2))$ is a *temporal* edge, and $((u,4),(x,4))$ is a *structural* edge. An interaction in this graph is a link at time i between two nodes (u,i) and (v,i).

1.2.2 *Temporal networks*

A *temporal network* is simply a sequence of interactions. Originally defined by [Holme and Saramäki, 2012], a variety of works fall under this denomination, and we summarize the main ones below. See Figure 1.4 for an example.

The definition itself of a temporal network is not formalized [Holme and Saramäki, 2012], and is simply a sequence of interactions. A consequence of this is the multiplication of works under other names, such as time varying networks, etc.

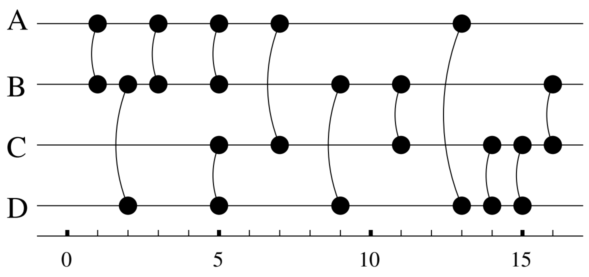


Figure 1.4: Representation of a temporal network, from [Holme and Saramäki, 2012]. Nodes (A,B,C,D) are represented horizontally with their names on the left of the diagram, and a link between two nodes *A* and *B* at abscissa 1 means that *A* and *B* interacted together at time 1.

The community of researchers working on temporal networks has mainly focused on questions related to paths in temporal networks: diffusion, reachability, and so on.

Paths in temporal networks have been extensively studied by [Pan and Saramäki, 2011]. The authors define different notions of paths that have no counterpart in graphs, and study the correlations between temporal paths and paths in the static graph induced by the temporal network; finally, they define an extension of closeness centrality to temporal networks.

Working upon these definitions, [Nicosia et al., 2013] proposes extensions of connectivity, closeness, betweenness, and spectral centrality, as well as studying temporal motifs.

Random walks are extended to temporal networks by [Starnini et al., 2012]; the same authors present different randomizing strategies, allowing one to single out the role of different temporal properties on empirical networks. More recently, [Saramäki and Holme, 2015] investigate the correlations between link activations in temporal networks with greedy walks.

Diffusion has been thoroughly studied on temporal networks. [Redmond and Cunningham, 2016] formulates the problem of identifying sequences of link activations that lead to the spread of a disease in terms of a time-respecting subgraph isomorphism problem.

Identifying the epidemic threshold [1] has attracted a lot of attention. [Valdano et al., 2015] analytically compute the epidemic threshold on temporal networks, and recent work by [Vestergaard et al., 2016] studies the minimum number of sensors that should be distributed in a population of n people to minimize the epidemic risk. Recently, [Holme, 2016] released a preprint studying disease spreading on 8 datasets of human contacts.

1. According to the World Health Organization, the epidemic threshold is "the critical number or density of susceptible hosts required for an epidemic to occur".

[Takaguchi et al., 2012] focuses on the identification of interactions in time that ease diffusion, and that are of importance to other nodes than just the two nodes involved by an interaction. They assess the importance of such interactions in temporal networks, and come to the conclusion that the diffusion of information is eased in real-world temporal networks by the bursty nature of their interactions.

Temporal networks had no formal basis until the recent work of [Batagelj and Praprotnik, 2016]. The authors provide algebraic definitions of temporal networks. The authors define notions of neighborhood, degree, and spectral centralities as algebraic operations on a semiring.

A recent review of the main works in the field of temporal networks has been published by [Holme, 2015].

1.2.3 *Time-Varying Graphs*

Time-Varying Graphs model evolving graphs, graphs where links and nodes can appear and disappear at arbitrary instances of time, see Figure 1.5 for an example.

[Casteigts et al., 2012] associate functions of presence $\rho(u,v)$ to edges: $\rho \mapsto \{0,1\}$ indicates whether a given link is available at a given time t, which corresponds to labelling links with their presence times. The au-

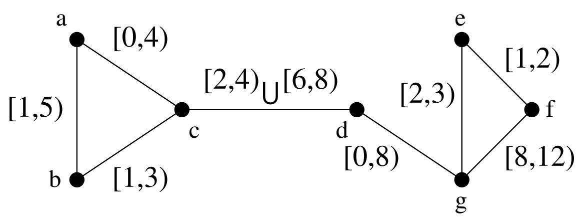


Figure 1.5: A Time-Varying Graph. Each link of the graph is labelled with the intervals where it is available: $[x, y)$ means that edge (u, v) is available from time x (included) to time y (excluded). For instance, the link (a, b) is available from time 1 to time 5; the link (c, d) is available from time 2 to time 4, and then from time 6 to time 8.

thors then formally define concepts of subgraphs, of journeys [2], and finally define a hierarchy of classes of Time-Varying Graphs based on their temporal properties. In [Casteigts et al., 2015], the authors define the language of feasible journeys given waiting time constraints, and study the expressivity of these languages.

2. A journey is a time-respecting path.

[Santoro et al., 2011] study the evolution of temporal and atemporal [3] statistics over time. This is interesting to study the convergence in time of statistics, for instance in a distributed system.

3. Defined on a Time-Varying Graph versus defined on a sequence of static graphs.

More recently, [Braud-Santoni et al., 2016] has published proofs of impossibility results, to prove for instance the convergence of deterministic algorithms on Time-Varying Graphs.

[Wehmuth et al., 2015b] aim at proposing a unique framework to unify evolving graphs representations. They rely on a reduction of the Multi-Aspect Graph model introduced by [Wehmuth et al., 2015a], which is a general model for representing multilayer graphs [4] varying over time.

4. In a multiplex graph, nodes are distributed in *layers*, and links can exist between two nodes of the same layer or two nodes of different layers.

1.3 Conclusion

We briefly presented the existing works aiming at describing interactions over time. We have structured this presentation in two parts: on the one hand, representations that induce a loss of information, and on the other hand, representations that do not induce such loss.

Since no information is lost for the models presented in Section 1.2, these models are equivalent for representing a given data: given a representation of a finite sequence of interactions as a Time-Varying Graph, for instance, one can readily define the associated temporal graph, or temporal network.

However, the different representations of each of these models is a good illustration of the fact that they correspond to different perspectives of the same underlying object: while in (evolving) graphs, focus is set on the graph structure at each instant, in the case of temporal networks, nodes are represented as horizontal lines, and focus is set on the activations of links through time.

In the link stream framework, we set focus on the fact that links arrive as a stream, like in temporal networks; however, unlike temporal networks, our goal is to define a comprehensive and consistent language to describe sequences of interactions.

Chapter 2 Basics

The goal of this chapter is to lay the foundations of our framework for describing sequences of interactions. In each section, we first recall the definition of the corresponding concept in graph theory, and then extend it to link streams.

2.1 Link streams

In a graph $G = (V, E)$, V is the set of nodes and $E \subseteq V \times V$ is the set of links.

A link (u, v) in E means that the two nodes u and v are in relation. In the example of Figure 2.1, nodes g and e are in relation. Links are undirected: no distinction is made between (u, v) and (v, u).

We say that G is simple if for all (u, v) in E, $u \neq v$, and there are no multiple links between two nodes u and v.

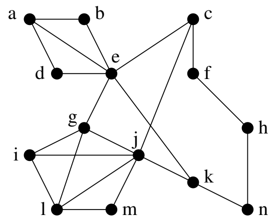


Figure 2.1: A graph $G = (V, E)$, with $V = \{a, b, c, \dots, m\}$ and $E = \{(a, b), (b, e), (c, f) \dots\}$. G has $n = 14$ nodes and $m = 22$ links.

In a link stream $L = (T, V, E)$, $T = [\alpha, \omega]$ is the time span of the stream, V is the set of nodes, and $E \subseteq T \times T \times V \times V$ is the set of links. We call $|V|$ the order of L and we denote it by n; we call $|E|$ its size and we denote it by $|L| = m$; we call $|T| = \omega - \alpha$ its duration and denote it by $\overline{L}$. In the example of Figure 2.2, the link stream has order $n = 4$, size $m = 9$ and duration $|T| = 20$ units of time.

A link $l = (b, e, u, v)$ in E means that nodes u and v interacted from time b to time e, and so $e \geq b$. We call

$e - b$ the duration of l and we denote it by $\bar{l}$. Links are undirected: we make no distinction between (b, e, u, v) and (b, e, v, u). In the example of Figure 2.2, nodes b and c interact from time 1 to time 7, nodes a and b interact from time 2 to time 9, and so on.

We say that L is simple if for all $l = (b, e, u, v)$ in E we have $u \neq v$ and $e > b$, and for all $l = (b, e, u, v)$ and $l' = (b', e', u, v)$ we have $[b, e] \cap [b', e'] = \emptyset$. In the remainder of this thesis, all link streams are considered to be simple if not stated otherwise.

For any u, v in V and t in T, we say that u and v interact at time t in L if there is a link (b, e, u, v) in E with $t \in [b, e]$. We denote by $\tau(u, v) = \cup_{(b,e,u,v) \in E}[b, e]$ the set of times at which u and v interact.

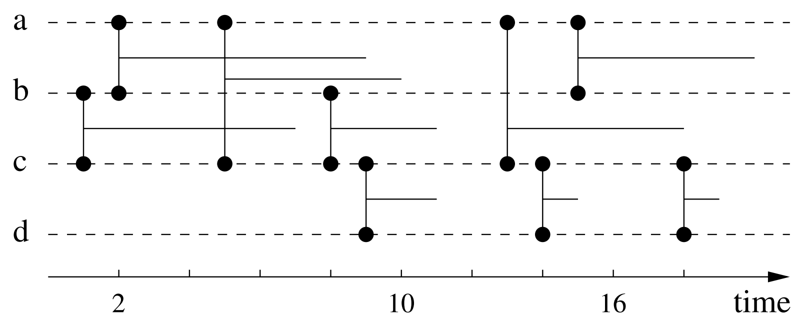


Figure 2.2: An example of link stream: $L = (T, V, E)$ with $T = [0, 20]$, $V = \{a, b, c, d\}$, and $E = \{(1, 7, b, c), (2, 9, a, b), (5, 10, a, c), \dots\}$. Each node is represented as a dotted horizontal line. Each link is represented by a line between the two horizontal lines representing involved nodes with dots at its extremities. The horizontal line attached to the link represents its duration.

2.2 Sub-links and sub-streams

Given two graphs $G = (V, E)$ and $G' = (V', E')$, G' is a subgraph of G if $V' \subseteq V$ and $E' \subseteq E$. This is denoted by $G' \subseteq G$. If $G' \subseteq G$ and $G \subseteq G'$ then $G = G'$.

Given a graph $G = (V, E)$ and a set of nodes $S \subseteq V$, the sub-graph induced by S is $G(S) = (S, \{(u, v) \in E : u \in S, v \in S\})$. Given a set of links $S \subseteq E$, the sub-graph induced by S is $G(S) = (\{u : (u, v) \in S\}, S)$. See Figure 2.3 for an example.

The *null graph* $G_\emptyset = (\emptyset, \emptyset)$ is the graph with no nodes and no edges, and the edgeless graph of order n, $G_{\not E} = (V, \emptyset)$ with $|V| = n$ is the graph with n nodes and no edges.

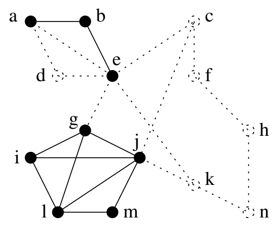


Figure 2.3: The two graphs in solid lines $G = (\{g, i, j, l, m\}, \{(g, i), (l, j), \dots\})$ and $G' = (\{a, b, e\}, \{(a, b), (b, e)\})$ are two subgraphs of the graph of Figure 2.1. G is the sub-graph induced by the set of nodes $\{g, i, j, l, m\}$ and G' is the sub-graph induced by the set of links $\{(a, b), (b, e)\}$.

The intersection of two graphs G and G' is their largest common sub-graph, and is denoted by $G \cap G'$.

The union of two graphs $G = (V, E)$ and $G' = (V', E')$ is the graph $(V \cup V', E \cup E')$, and is denoted by $G \cup G'$.

Given two link streams $L = (T, V, E)$ and $L' = (T', V', E')$, we say that L' is a substream of L if $V' \subseteq V$, $T' \subseteq T$, and for all u, v in V' and t in T', if u and v interact at time t in L' then they also interact at time t in L [1]. We denote this by $L' \subseteq L$. If $L' \subseteq L$ and $L \subseteq L'$ then $L' = L$.

1. Notice however that E' is *not* included in E.

Given a link $l = (b, e, u, v)$, we say that $l' = (b', e', u', v')$ is a sub-link of l if $u' = u$, $v' = v$, and $[b', e'] \subseteq [b, e]$; we denote this by $l' \subseteq l$. Notice that, if $L = (T, V, E)$ and $L' = (T', V', E')$ are simple link streams, then $L' \subseteq L$ if and only if for all l' in E' there is a l in E such that $l' \subseteq l$. In Figure 2.2, $L' = ([0, 10], \{a, b, c\}, \{(2, 5, b, c), (2, 7, a, b)\})$ is a sub-stream of L. See Figure 2.4 for an example.

Given a link stream $L = (T, V, E)$ and a set of pairs of nodes $S \subseteq V \times V$, let us denote by $V(S)$ the set of nodes involved in S: $V(S) = \{v \in V : \exists (v, u) \in S\}$, and let us denote by $E(S)$ the set of links involved in S: $E(S) = \{(b, e, u, v) \in E : (u, v) \in S\}$. We define the sub-stream of L induced by S as $L(S) = (T, V(S), E(S))$. By extension, if S is a set of nodes then we define the sub-stream of L induced by S as $L(S) = L(S \times S)$. We denote by $L(u, v)$ the sub-stream $L(\{(u, v)\})$ and by $L(v)$ the sub-stream $L(\{v\} \times V)$. In the example of Figure 2.2, the sub-stream of L induced by nodes a, b is $L(\{a, b\}) = (T, \{a, b\}, \{(2, 9, a, b), (15, 20, a, b)\})$, and the sub-stream induced by the pair of nodes (c, d) is $L(\{(c, d)\}) = (T, \{c, d\}, \{(9, 11, c, d), (14, 15, c, d), (18, 19, c, d)\})$.

Given a time interval $T' = [\alpha', \omega'] \subseteq T$, let us denote by $E_{T'} = E_{\alpha'..\omega'}$ the set of quadruplets (b', e', u, v) such that there exists (b, e, u, v) in E with $[b', e'] = [b, e] \cap T'$. We define the sub-stream $L_{T'} = L_{\alpha'..\omega'}$ of L induced by T' as $(T', V, E_{T'})$. By extension, we denote by L_t the sub-stream $L_{t..t}$ of duration 0.

The null link stream $L_\emptyset = (\emptyset, \emptyset, \emptyset)$ is the stream with no time, no nodes and no links. For any $T \subseteq \mathbb{R}$, the order

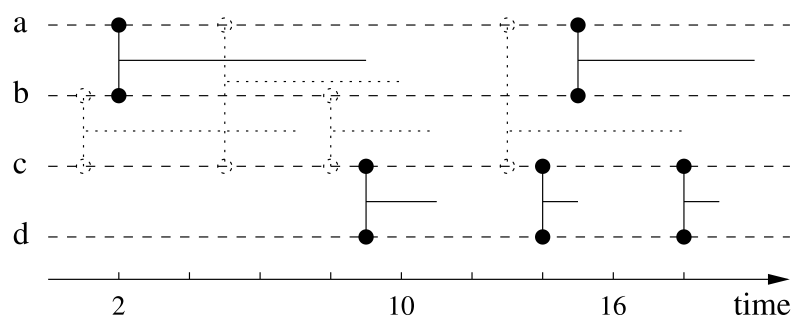


Figure 2.4: The two streams in solid lines $L = ([0,20],\{a,b\},\{(2,9,a,b),(15,20,a,b)\})$ and $L' = ([0,20],\{c,d\},\{(9,11,c,d),(14,15,c,d),\dots\})$ are two substreams of the stream of Figure 2.2. L is the substream induced by the set of nodes $\{a,b\}$ and L' is the substream induced by the pair of nodes (c,d).

0 stream, *i.e.* the stream with no nodes and no edges, is $L_{\not V,\not E} = (T,\emptyset,\emptyset)$. For any $T \subseteq \mathbb{R}$ and for any $n \in \mathbb{N}$, the edgeless stream of order n is the stream with n nodes and no edges, *i.e.* $L_{\not E} = (T,V,\emptyset)$ with $|V| = n$.

We define the intersection of two link streams L and L' as their largest common sub-stream, and we denote it by $L \cap L'$.

We define the union of two link streams $L = (T,V,E)$ and $L' = (T',V',E')$ as the stream $(T'',V \cup V',E \cup E')$, where T'' is the smallest interval such that $T \subseteq T''$ and $T' \subseteq T''$. We denote it by $L \cup L'$. Notice that even if L and L' are simple, $L \cup L'$ is not necessarily simple.

Finally, we define $\sigma(L)$ the simplification of L as the smallest sub-stream of L (in terms of number of links) such that $\sigma(L) = L$ and for all (b,e,u,v) in $\sigma(L)$, $e \geq b$. This necessarily is a simple link stream.

2.3 Line stream

The line graph $\hat{G}$ of $G = (V,E)$ is the graph $\hat{G} = (E,\hat{E})$ in which each node is a link of G and two nodes are linked if they have an extremity in common: $((u,v),(x,y))$ is in $\hat{E}$ if $\{u,v\} \cap \{x,y\} \neq \emptyset$.

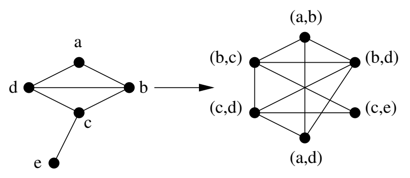


Figure 2.5: A graph $G = (V,E)$ and its line graph $\hat{G}$. In the line graph, there is a link between (a,b) and (b,d) because these two edges have node b in common.

The line stream $\hat{L}$ of $L = (T,V,E)$ is the link stream $\hat{L} = (T,\hat{V},\hat{E})$, with $\hat{V} = \{(u,v) : \exists(b,e,u,v) \in E\}$ and two nodes are linked if they share a node and are linked at the same time: $(b,e,(u,v),(x,y))$ is in $\hat{E}$ if $\{u,v\} \cap \{x,y\} \neq \emptyset$ and there exists two links (b',e',u,v) and (b'',e'',x,y) in E such that $[b',e'] \cap [b'',e''] = [b,e]$. See Figure 2.6 for an illustration.

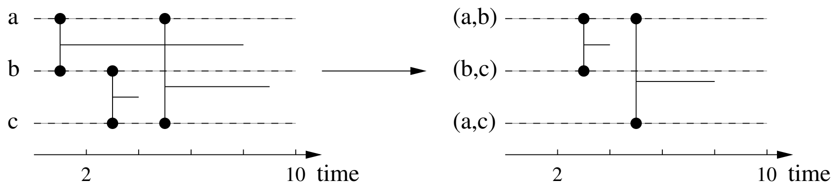


Figure 2.6: A link stream $L = ([0,10], \{a,b,c\}, \{(1,8,a,b),(3,4,b,c),(5,9,a,c)\}$ and its corresponding line stream $\hat{L} = ([0,10], \{(a,b),(b,c),(a,c)\}, \hat{E})$, with $\hat{E} = \{((3,4,(a,b),(b,c)),((5,8,(a,b),(a,c))\}$. There is no link between (b,c) and (a,c) because these two pairs of nodes do not have links at the same time in L.

Just like for graphs, the line stream of the line stream, $\hat{\hat{L}}$, is not equal to L.

2.4 Induced graphs

Given a link stream $L = (T,V,E)$, we define its induced graph $G(L) = (V,E')$ where $(u,v) \in E'$ if and only if there exist b and e such that $(b,e,u,v) \in E$. In other words, it is the graph in which there is a link between two nodes if they interacted at least once in L. One may in addition associate to each link (u,v) in E' a weight $w(u,v) = |L(u,v)|$ capturing the number of interactions, $w(u,v) = \sum_{l \in L(u,v)} \bar{l}$ capturing their total duration, or other quantities of interest.

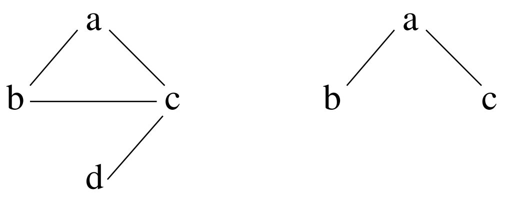


Figure 2.7: The graph $G = (\{a,b,c,d\}, \{(a,b),(a,c),(b,c),(c,d)\})$ induced by the link stream L of Figure 2.2 (left), and the graph $G_{12..20}(\{a,b,c\}) = (\{a,b,c\}, \{(a,b),(a,c)\})$ induced by the sub-stream of L induced by nodes a,b and c over interval $[12,20]$, $L_{12..20}(\{a,b,c\})$.

Given L, we denote by $G_{t..t'}$ the graph $G(L_{t..t'})$, by G_t the graph $G(L_t) = G_{t..t}$, and by $G(S)$ the graph $G(L(S))$. Figure 2.7 displays $G(L)$ for the example of Figure 2.2, and for the graph induced by nodes a, b and c from time 12 to time 20, $G_{12..20}(\{a,b,c\})$.

2.5 Conclusion

We presented in this first chapter the basis of our formalism. We defined link streams and substreams, line streams, and graphs induced by streams. We focused on simple, undirected and unweighted streams.

Adapting these definitions to directed link streams is just a matter of making a difference between (b, e, u, v) and (b, e, v, u), and defining notions of substreams, line streams and so on subsequently. Similarly, loops can be taken into account by authorizing links (b, e, u, v) in E such that $u = v$.

Using these definitions, one could easily define a weighted link stream as $L = (T, V, E)$, where $E \subseteq T \times T \times V \times V \times \mathbb{R}$. Elements of E would then be (b, e, u, v, w), where w is the weight associated to the link.

Chapter 3

Density-based notions

After having defined the very basis of our formalism, we move on to define more subtle notions. Since our goal is to make a language for link streams similar to how graph theory describes networks, we first focus on one of the most fundamental notion in graph theory: *density*. The concept of density leads to the ones of neighborhood and degrees of the graph.

In graph theory, density-based notions are of paramount importance to study the structure of the network; dense groups are typical indicators of groups of friends in social networks, or botnets in IP traffic, for instance. The degree distribution of a graph has a strong descriptive power, and studying it is classical to gain insight on a graph.

3.1 Density, neighborhood and degree

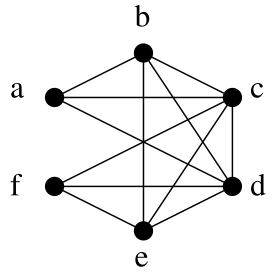


Figure 3.1: A graph G with 6 nodes and 12 links. Its density $\delta(G)$ is $\frac{2\cdot 12}{6\cdot 5} = 0.8$. The neighborhood of node b, $N(b)$, is the set $\{a, c, d, e\}$, and its degree $d(b)$ is 4.

In a simple graph $G = (V, E)$, the density is the probability when one takes two random nodes u and v that there is a link (u, v) in E: $\delta(G) = \frac{2m}{n(n-1)}$ where $n = |V|$ denotes the number of nodes and $m = |E|$ the number of links. The neighborhood $N(v)$ of $v \in V$ is the set of nodes linked to v: $N(v) = \{u \in V,\ \exists (u, v) \in E\}$. The degree $d(v)$ of v is the size of its neighborhood: $d(v) = |N(v)|$ and the average degree in G is $d(G) = \frac{\sum_{v \in V} d(v)}{n}$. Figure 3.1 illustrates these notions. Notice that the following relation between density and average degree holds: $\delta(G) = \frac{d(G)}{n-1}$.

The goal of this section is to extend these notions to a simple link stream $L = (T, V, E)$ with $n = |V|$ nodes, $m = |E|$ links, and $T = [\alpha, \omega]$.

Density

We define the density $\delta(L)$ of L as the probability when one takes two random nodes u and v and a random time instant t that they are in interaction (*i.e.* there is a link (b, e, u, v) in E such that $t \in [b, e]$):

$$\delta(L) = \frac{2 \sum_{l \in E} \bar{l}}{n(n-1)(\omega - \alpha)} \tag{3.1}$$

where $n = |V|$ denotes the number of nodes. The density is defined only for link streams such that for all $n \geq 2$ and $\omega > \alpha$ [1]. See Figure 3.2 for an illustration, and Figure 3.3 for a more complex example.

1. By convention, if the duration of the stream is 0 then its density is nothing but the density of its induced graph.

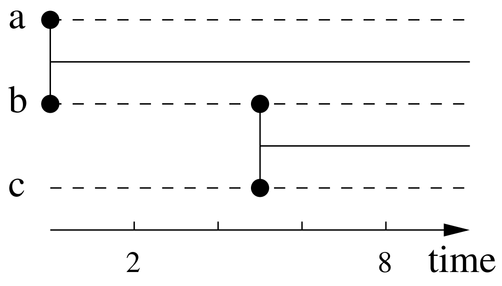


Figure 3.2: A link stream L, with $|V| = 3$ and $|T| = 10$ units of time. Link $(0, 10, a, b)$ accounts for 10 units of time in the numerator, and link $(5, 10, b, c)$ for 5 units of time. The density of L is $\delta(L) = \frac{2\cdot(10+5)}{3\cdot 2\cdot 10} = 0.5$.

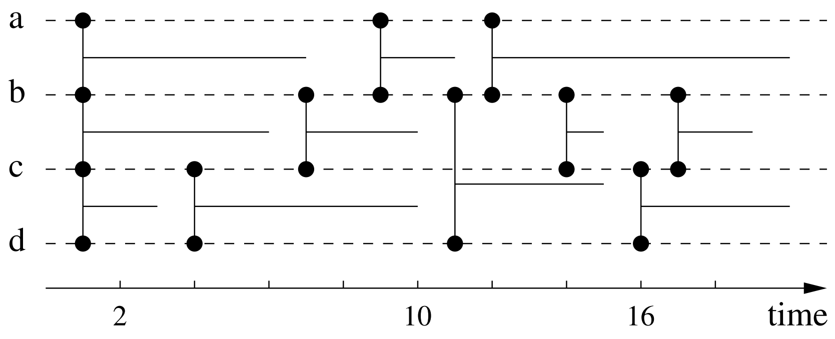


Figure 3.3: A link stream L, with $|V| = 4$ and $|T| = 20$ units of time. Its density is $\delta(L) = \frac{2\cdot 45}{4\cdot 3\cdot 20} = 0.375$.

Notice that when $n = 2$, *i.e.* $V = \{u, v\}$, the density is nothing but the frequency of occurrences of (u, v). If all links have duration $\omega - \alpha$ then the density of L is nothing but the density of its induced graph $G(L)$. Therefore, density of link streams generalize both frequency and graph density, by combining their temporal and structural natures into a same notion.

The density of L is also the average density of $G(L_t)$, the graph induced by L at time t, for all t in T:

$$\delta(L) = \frac{1}{\omega - \alpha} \int_{\alpha}^{\omega} \delta(G(L_t))dt.$$

Neighborhood and degree

The neighborhood $N_{x..y}(v)$ of node $v \in V$ from time $x \in T$ to time $y \in T$, $y > x$ is the set of pairs (u, t) such that $u \in V$ is linked to v at time $t \in [x, y]$: $N_{x..y}(v) = \{(u, t) : \exists(b, e, u, v) \in E,\ t \in [b, e] \cap [x, y]\}$.

We define the degree of v from x to y as:

$$d_{x..y}(v) = \sum_{(b,e,u,v) \in E} \frac{|[b, e] \cap [x, y]|}{y - x} = \sum_{l \in L_{x..y}(v)} \frac{\bar{l}}{y - x}$$

In this way, the contribution of each neighbor u to the degree of v is weighted by the duration of its links with v: $d_{x..y}(v) = \sum_{u \in N_{x..y}(v)} \sum_{(b,e,u,v) \in E} \frac{|[b,e] \cap [x,y]|}{y-x}$.

Notice also that $d_{x..y}(v)$ is the expected degree $d_t(v)$ at time t when t is chosen randomly in $[x, y]$. The neighborhood $N_t(v)$ of node $v \in V$ at time $t \in T$ is the set of nodes linked to v at time t: $N_t(v) = \{u \in V,\ \exists(b, e, u, v) \in E,\ t \in [b, e]\}$. The degree $d_t(v)$ of v at time t is the size of its neighborhood: $d_t(v) = |N_t(v)|$.

By extension, we define the degree of v in L as $d(v) = d_{\alpha..\omega}(v)$:

$$d(v) = \sum_{l \in L(v)} \frac{\bar{l}}{\omega - \alpha} \tag{3.2}$$

and the average degree in L is $d(L) = \frac{\sum_{v \in V} d(v)}{n}$. Just like for density, the degree of v in L is the average value of $d_t(v)$ for all $t \in T$. Degrees in the link stream are illustrated in Figure 3.4.

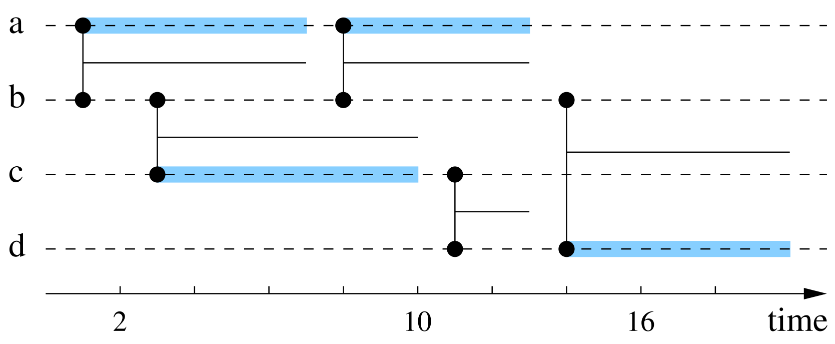


Figure 3.4: Illustration of neighborhoods and degrees in link streams. The neighborhood of node b is depicted in blue. Node b has node a in its neighborhood from time 1 to time 7 and from time 8 to time 13, node c from time 3 to time 10 and node d from time 14 to time 20. The degree of b is $d(b) = \frac{23}{20} = 1.15$. The average degree of L is $d(L) = \frac{\frac{11}{20}+\frac{23}{20}+\frac{9}{20}+\frac{8}{20}}{4} = 0.6375$.

Like in graphs, the following relation holds between the density and average degree of a link stream:

$$\begin{aligned}\delta(L) &= \frac{2\sum_{l\in E}\bar{l}}{n(n-1)(\omega-\alpha)} = \frac{2\cdot\sum_{l\in E}\frac{\bar{l}}{\omega-\alpha}}{n(n-1)} \\ &= \frac{2\cdot\frac{1}{2}\cdot\sum_{v\in V}\sum_{l\in L(v)}\frac{\bar{l}}{\omega-\alpha}}{n(n-1)} = \frac{\sum_{v\in V}d(v)}{n(n-1)} = \frac{d(L)}{n-1}.\end{aligned}$$

3.2 Node clusters

In a graph $G = (V, E)$, a node cluster is a subset C of V. We denote by $E(C) = E \cap (C \times C)$ the set of links between nodes in C, by $n_C = |C|$ the number of nodes in C, by $m_C = |E(C)|$ the number of links involved in C, and by $G(C) = (C, E(C))$ the subgraph of G induced by C.

Given two node clusters C and C' of G, their inter-cluster density is defined as:

$$\delta(C,C') = \frac{|E\cap(C\times C')|}{|C\setminus C'|\cdot|C'| + |C'\setminus C|\cdot|C\cap C'| + \frac{|C\cap C'|\cdot(|C\cap C'|-1)}{2}} \tag{3.3}$$

It is the number of links between a node in C and a node in C' divided by the number of possible such links: $|C\setminus C'|\cdot|C'|$ is the number of links (u,v) such that $u \in C$ and $v \in C'$, $|C'\setminus C|\cdot|C\cap C'|$ is the number of links (u,v) such that $u \in C\cap C'$ and $v \in C'$, and $\frac{|C\cap C'|\cdot(|C\cap C'|-1)}{2}$ is the number of links (u,v) such that $u,v \in C\cap C'$. For any node cluster C, the internal density of C is defined as

$\delta(C,C)$; it is nothing but the density of $G(C)$, which we denote by $\delta(C)$. The external density of C is defined as $\delta(C,\overline{C})$, denoted by $\overline{\delta}(C)$; it is nothing but the density of the subgraph induced by pairs of nodes (u,v) where $u \in C$ and $v \notin C$. See Figure 3.5. The internal degree of a node v in C is $d(v,C) = |\{(u,v) \in E : u \in C\}|$, and its external degree $\overline{d}(v,C) = |\{(u,v) \in E : u \notin C\}|$. The internal average degree of a node cluster C is $d(C) = \frac{1}{|C|}\sum_{v \in C} d(v,C)$, and the external average degree of a node cluster C is $\overline{d}(C) = \frac{1}{|C|}\sum_{v \in C} \overline{d}(v,C)$ [2].

2. Notice that, for all $v \in V$, $d(v,C) + \overline{d}(v,C) = d(v)$.

For any given set S, a family $P = \{S_i\}_{i=1..k}$ is a partition of S if $\bigcup_i S_i = S$, and for all i,j in $1..k$, $i \neq j$, $S_i \cap S_j = \emptyset$. For any interval $T = [x,y] \subseteq \mathbb{R}$, a partition of T is a finite sequence $x = (x_i)_{i=1..k}$ of real numbers such that $x = x_0 < x_1 < x_2 < ... < x_k = y$.

A partition of G in clusters is a family $C = \{C_i\}_{i=1..k}$ partitioning V into k groups of nodes: for all $i \neq j$, $C_i \cap C_j = \emptyset$ and $\cup_i C_i = V$. The intra-cluster density of P is the average internal density of all clusters C_i, weighted by the number of nodes in C_i. The inter-cluster density of P is the average external density of all clusters C_i, weighted by the number of nodes in C_i.

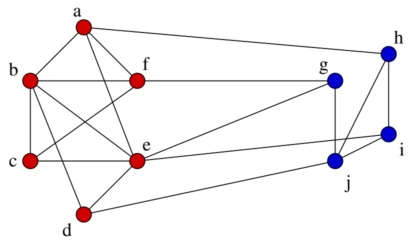


Figure 3.5: A graph G with two node clusters $C = \{a,c,d,e,f\}$ and $C' = \{g,h,i,j\}$.

In a link stream, a node cluster is a subset C of $V \times T$, and $(v,t) \in C$ means that v belongs to C at time t. We denote by $V(C)$ the set of nodes involved in C: $V(C) = \{v : \exists(v,t) \in C\}$. We denote by $T(C)$ the time interval over which nodes are involved in C: $T(C) = [\min(t : \exists(v,t) \in C), \max(t : \exists(v,t) \in C)]$. Finally, we denote by $N(C)$ the neighborhood of cluster C: $N(C) = \{(v,t) \in C : \exists(b,e,u,v) \in E, t \in [b,e], u \in V(C), v \notin V(C)\}$.

For all $v \in V$, we denote by $C(v) = \{t : \exists(v,t) \in C\}$ the set of time instants at which v is involved in C, and by $\mathcal{C}(v)$ the set of all bounds of all maximal intervals included in $C(v)$. We denote by $C(u,v)$ the set $C(u) \cap C(v) \cap \tau(u,v)$, and by $E(C)$ the set of all (b,e,u,v) such that $[b,e]$ is a maximal interval in $C(u,v)$. In other words, $C(u,v)$ is the set of times such that u and v are in C and are interacting together.

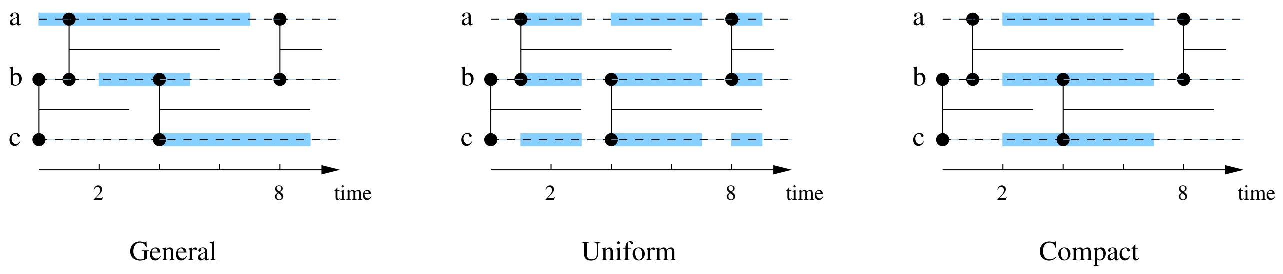


Figure 3.6: A link stream $L = ([0,10], \{a,b,c\}, \{\dots\})$ and three node clusters involving a, b and c, indicated in blue. A blue line from time x to time y over a node means that this node is in the node cluster from time x to time y. Left: a is in the node cluster from time 0 to time 7 ($C(a) = [0,7]$), b is in the cluster from time 2 to time 5, and c is in the cluster from time 4 to time 9. Middle: a *uniform* node cluster; $C(a) = C(b) = C(c) = [1,3] \cup [4,7] \cup [8,9]$. Right: a *compact* node cluster; $C(a) = C(b) = C(c) = [2,7]$.

Finally, we define $L(C) = (T(C), V(C), E(C))$, the sub-stream of L induced by C..

Notice that, if for all u and v in $V(C)$, $C(v) = C(u)$, we say that C is uniform. In the special case where $C(v) = T(C)$ for all v involved in C, then C is fully defined by $V(C)$ and $T(C)$, and we have $L(C) = L_{T(C)}(V(C))$. We then say that C is a compact cluster. See Figure 3.6 for an illustration of node clusters.

A partition of L into node clusters is a partition of $V \times T$, *i.e.* a family $P = \{C_1, C_2, \dots, C_k\}$ of k node clusters such that for all v in V, the union of all $\mathcal{C}_i(v)$ is a partition of T. See Figure 3.7.

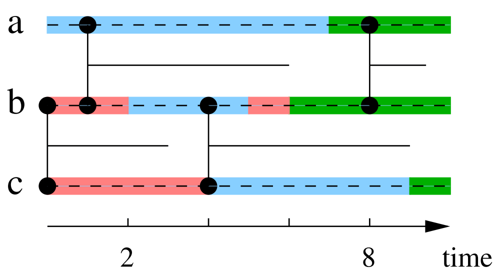


Figure 3.7: A partition in three node clusters C_1 (blue), C_2 (green), C_3 (red). C_1 involves a from time 0 to time 7, b from time 2 to time 5, and c from time 4 to time 9. C_2 involves b over times $[0,2]$ and $[5,6]$ and c from time 0 to time 4. C_3 involves a from 7 to 10, b from 6 to 10, and c from 9 to 10. For any node u and any time t, there exists a single $i \in 1,2,3$ such that node u is involved at time t in C_i. Notice that $\cup_i C_i(b) = [0,10]$.

We define the number of nodes $n(C)$ and links $m(C)$ in cluster C as follows:

$$n(C) = \sum_{v \in V} \frac{|C(v)|}{\omega - \alpha} \quad \text{and} \quad m(C) = \frac{1}{2} \sum_{u \in V} \sum_{v \in V} \frac{|C(u,v)|}{\omega - \alpha}$$

In other words, each node v accounts for $n(v,C) = \frac{|C(v)|}{\omega-\alpha}$ nodes in C and each pair of nodes u, v accounts for $m(u,v,C) = \frac{|C(u,v)|}{\omega-\alpha}$ links in C.

If P is a partition of L into node clusters then for all v, $\sum_C n(v,C) = 1$ and for all u, v, $\sum_C m(u,v,C) = \frac{|\tau(u,v)|}{\omega-\alpha}$. As a consequence, $\sum_C n(C) = n$.

We define the internal and external degree of node v in cluster C as follows:

$$d(C,v) = \sum_{u\in V} \frac{|C(v,u)|}{\omega-\alpha} \text{ and } \overline{d}(C,v) = \sum_{u\in V} \frac{|C(v,\overline{u})|}{\omega-\alpha}$$

where $C(v,\overline{u})$ denotes $\overline{C(u)} \cap C(v) \cap \tau(u,v)$.

For all v, we have $\sum_C d(C,v) + \overline{d}(C,v) = d(v)$:

$$\sum_C d(C,v) + \overline{d}(C,v) = \sum_C \sum_u \frac{|C(v,u)| + |C(v,\overline{u})|}{\omega-\alpha}$$
$$= \sum_u \sum_C \frac{|C(v,u) \cup C(v,\overline{u})|}{\omega-\alpha} \sum_u \sum_C \frac{|\tau(u,v) \cap C(v) \cap (C(u) \cup C(\overline{u}))|}{\omega-\alpha} = \sum_u \sum_C \frac{|C(v) \cap \tau(u,v)|}{\omega-\alpha}$$
$$= \sum_u \frac{|\tau(u,v)|}{\omega-\alpha} = d(v)$$

We define the internal and external degree of a cluster C:

$$d(C) = \frac{1}{n(C)} \sum_{v\in V} \frac{|C(v)|}{\omega-\alpha} d(C,v) = \frac{\sum_{v\in V} |C(v)| d(C,v)}{\sum_{v\in V} |C(v)|} \tag{3.4}$$

and

$$\overline{d}(C) = \frac{1}{n(C)} \sum_{v\in V} \frac{|C(v)|}{\omega-\alpha} \overline{d}(C,v) = \frac{\sum_{v\in V} |C(v)| \overline{d}(C,v)}{\sum_{v\in V} |C(v)|} \tag{3.5}$$

If C is a compact cluster, then $d(C) = \frac{|T(C)|}{\omega-\alpha} d(L(C))$. Indeed, for all $v \in V$, $C(v) = T(C)$:

$$\begin{aligned} d(C) &= \frac{\sum_{v\in V} |C(v)| d(C,v)}{\sum_{v\in V} |C(v)|} \\ &= \frac{\sum_{v\in V} |C(v)| \sum_{u\in V} \frac{|C(u,v)|}{\omega-\alpha}}{\sum_{v\in V} |C(v)|} \\ &= \frac{\sum_{v\in V} |T(C)| \sum_{u\in V} \frac{|T(C)\cap\tau(u,v)|}{\omega-\alpha}}{\sum_{v\in V(C)} |T(C)|} \\ &= \frac{\sum_{v\in V} \sum_{u\in V} \frac{|T(C)\cap\tau(u,v)|}{\omega-\alpha}}{|V(C)|} \\ &= \frac{\sum_{u,v\in V} |T(C) \cap \tau(u,v)|}{|V(C)|(\omega-\alpha)} \end{aligned}$$

and

$$\begin{aligned} d(L(C)) &= \frac{1}{|V(C)|} \sum_{v \in V(C)} \frac{\sum_u |\tau(u,v) \cap T(C)|}{|T(C)|} \\ &= \frac{1}{|V(C)| \cdot |T(C)|} \sum_{v \in V(C)} \sum_u |\tau(u,v) \cap T(C)| \end{aligned}$$

In the case of a link stream $L = (T, V, E)$ and a node cluster C that contains all L, *i.e.* for all $u \in V$ and for all $t \in T$, $(u,t) \in C$, the internal density of C is the density of L and the internal degree of C is the average degree of L, since for all $u \in V$, $C(u) = |T|$, and for all $u, v \in V \times V, C(u,v) = \tau(u,v)$.

We now define the density between two clusters C and C' as the fraction of possible links between them that do exist:

$$\begin{aligned} \delta(C, C') &= \frac{\sum_{u \in V(C), v \in V(C')} |C(u) \cap C'(v) \cap \tau(u,v)|}{\sum_{u \in V(C), v \in V(C')} |C(u) \cap C'(v)|} \\ &= \frac{\sum_{u \neq v} |C(u) \cap C'(v) \cap \tau(u,v)|}{\sum_{u \neq v} |C(u) \cap C'(v)|} \end{aligned} \tag{3.6}$$

In other words, it is the probability when one takes a random triplet u, v and t such that u is in C and v is in C' at time t, that there is a link between u and v at time t.

We obtain the internal and external densities of cluster C:

$$\delta(C) = \frac{\sum_{u,v} |C(u,v)|}{\sum_{u,v} |C(u) \cap C(v)|} = \delta(C, C) \tag{3.7}$$

and

$$\overline{\delta}(C) = \frac{\sum_u \sum_{\overline{v}} |C(u, \overline{v})|}{\sum_{u,v} |C(u) \cap C(\overline{v})|} = \delta(C, \overline{C}) \tag{3.8}$$

The internal density of cluster C is the probability when one takes a random triplet u, v and t such that u and v are in C at time t, that there is a link between u and v at time t. The external density of cluster C is the probability, when one takes a random triplet u, v and t such that u is in C at time t and v is not in C at time t, that there is a link between u and v at time t. For a numerical illustration of internal and external densities, see Figure 3.8.

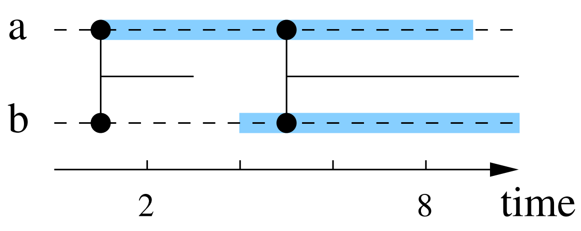


Figure 3.8: The link stream $L = ([0,10], \{a,b\}, \{(1,3,a,b),(5,10,a,b)\})$, and a node cluster C involving a from 1 to 9 and b from 4 to 10. $\tau(a,b) = [1,3] \cup [5,10]$. The internal density of C is $\delta(C) = \frac{||[5,9]||}{||[4,9]||} = \frac{4}{5}$, and the external density of C is $\overline{\delta}(C) = \frac{||[1,3]||+||[9,10]||}{||[1,4]||+||[9,10]||} = \frac{3}{4}$.

If C is a compact cluster, then $\delta(C) = \delta(L(C))$, and so $\delta(C) = \frac{d(C)}{|V(C)|-1} \frac{\omega-\alpha}{|T(C)|}$.

Notice that there is no general relation between the internal density and the degree of a cluster, see Figure 3.9. However, if C is uniform, then the following relation holds: $\delta(C) = \frac{d(C)}{|V(C)|-1} \frac{\omega-\alpha}{|T(C)|}$.

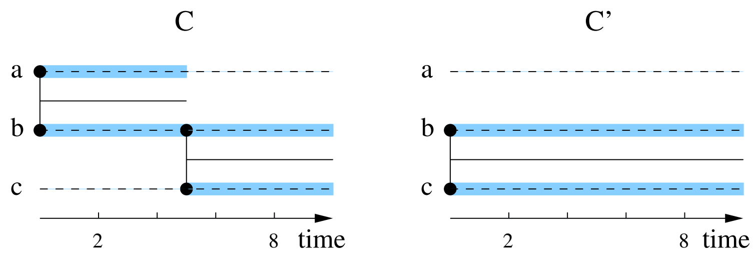


Figure 3.9: Example that there is no general relation between the degree of a node cluster C, $d(C)$, and its density $\delta(C)$. The two clusters have 2 nodes and 1 link. **Left:** The node cluster C has density $\delta(C) = \frac{0.5+0+0.5}{0.5+0+0.5} = 1$ and degree $d(C) = \frac{0.25+1+0.25}{2} = \frac{3}{4}$. **Right:** The node cluster C' has density $\delta(C') = \frac{1+1}{1+1} = 1 = \delta(C)$ and degree $d(C') = \frac{1+1}{1+1} \neq d(C)$.

Notice also that in the general case, the internal density of a cluster C is not the average density of the graph induced by C at time t for all $t \in T(C)$, *i.e.* $\delta(C) \neq \int_{T(C)} \delta(G_t(C))dt$. See Figure 3.10 for an example.

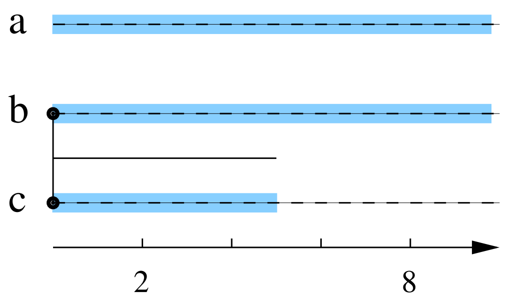


Figure 3.10: The link stream $L = ([0,10], \{a,b,c\}, \{(0,5,b,c)\}$, and node cluster C where a and b are in C over $[0,10]$, and c is in C over $[0,5]$. The internal density of C is $\delta(C) = \frac{5}{20} = \frac{1}{4}$, while the average density for all $t \in [0,10]$ of the graph G_t is $\frac{1}{10}\int_0^{10} \delta(G_t)dt = \frac{5}{3 \cdot 2 \cdot 10} = \frac{1}{12}$.

Given a partition P of link stream L into node clusters, we define the intra-cluster and inter-cluster densities of P as:

$$\delta(P) = \frac{\sum_{C \in P} \sum_{u,v \in V, u \neq v} |C(u,v)|}{\sum_{C \in P} \sum_{u,v \in V, u \neq v} |C(u) \cap C(v)|} \qquad (3.9)$$

and

$$\overline{\delta}(P) = \frac{\sum_{C\in P}\sum_{u,v\in V,u\neq v}|C(u,\overline{v})|}{\sum_{C\in P}\sum_{u,v\in V,u\neq v}|C(u)\cap C(\overline{v})|} \tag{3.10}$$

In other words, $\delta(P)$ is the probability when one takes a random triplet u, v and t such that u and v are in the same cluster at time t, that there is a link between u and v at time t. Likewise, $\overline{\delta}(P)$ is the probability when one takes a random triplet u, v and t such that u and v are in distinct clusters at time t, that there is a link between u and v at time t.

Finally, notice that the neighborhood $N(u)$ of a node $u \in V$ defined in Section 3.1 can be seen as a node cluster, and the size of this cluster is nothing but the degree of v.

3.3 Cliques

In a simple graph G, a clique is a set $C \subseteq V$ such that the sub-graph of G induced by C has density 1. In other words, for all u and v in C, the link (u,v) exists in E. Notice that if a sub-graph $G' = (V',E')$ of G has density 1, then V' necessarily is a clique of G. Therefore we make no distinction between a clique and the associated induced sub-graph. A clique C is maximal if there is no other clique C' such that $C \subset C'$.

In a simple link stream, a clique is a set of nodes $X \subseteq V$ and an interval of time $Y \subseteq T$ such that the sub-stream of L induced by X over Y has density 1. In other words, for all u and v in X, there is a link (b,e,u,v) in E such that $Y \subseteq [b,e]$. Notice that if a sub-stream $L' = (T',V',E')$ of L has density 1, then (V',T') necessarily is a clique of L. Therefore we make no distinction between a clique and the associated induced sub-stream.

A clique $C' = (X',Y')$ is included in another clique $C = (X,Y)$ if $X' \subseteq X$ and $Y' \subseteq Y$, *i.e.* the link stream induced by C' is included in the one induced by C. We denote this by $C' \subseteq C$. A clique C is maximal if there is no other clique C' such that $C \subset C'$.

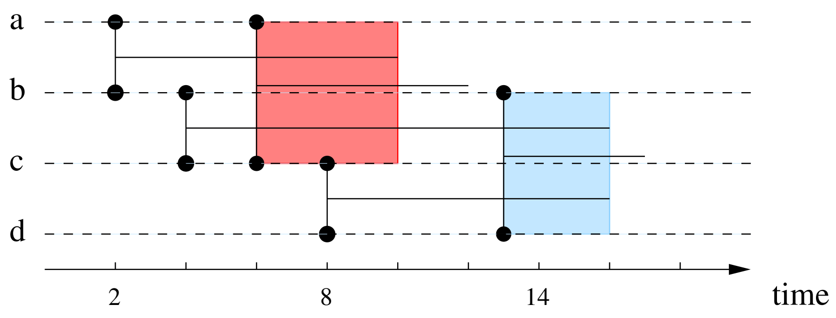


Figure 3.11: The link stream L and two maximal cliques $A = (\{a, b, c\}, [6, 10])$ and $B = (\{b, c, d\}, [13, 16])$. Saying that A is a clique means that from time 7 to time 9, all links between $\{a, b, c\} \times \{a, b, c\}$ exist, which is equivalent to saying that the substream of L induced by A, $L_{7..9}(\{a, b, c\})$, has density 1.

3.4 Clustering coefficient and transitivity ratio

In a graph $G = (V, E)$, the clustering coefficient of a given node v is the internal density of its neighborhood: $cc(v) = \delta(N(v))$. In other words, $cc(v)$ is the probability that two randomly chosen neighbors of v are linked together in G. Notice that the clustering coefficient of nodes of degree 1 is undefined. The clustering coefficient of G as a whole is the average clustering coefficient of all nodes v such that $d(v) > 1$: $cc(G) = \frac{\sum_{v \in V} cc(v)}{n}$. See Figure 3.12 for an example.

The transitivity ratio of G is the probability when one takes a random triplet of nodes u, v, w such that $(v, u) \in E$ and $(v, w) \in E$ that (u, w) is in E too: $tr(G) = \frac{\#_\Delta}{\#_\vee}$ where $\#_\Delta$ is the number of closed triplets (*i.e.*the triplets for which (u, w) is in E), and $\#_\vee$ is the number of connected triplets.

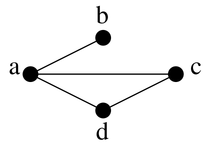


Figure 3.12: The graph $G = (\{a, b, c, d\}, \{(a, b), (a, c), (a, d), (c, d)\}$. The clustering coefficient of G is $cc(G) = \frac{\frac{1}{3}+1+1}{1} \approx 0.66$, and the transitivity ratio of G is $tr(G) = \frac{3}{5} = 0.6$.

In a link stream $L = (T, V, E)$, the clustering coefficient of v is the internal density of the node cluster $N_{>1}(v) = \{(u, t) : \exists x \in V, \exists (b, e, u, v) \in E, \exists (b', e', x, v) \in E, t \in [b, e] \cap [b', e']\}$:

$$cc(v) = \delta(N_{>1}(v))$$

where $N_{>1}(v)$ is the node cluster of the neighborhood of v when v has more than 1 neighbor.

In other words, $cc(v)$ is the probability when one takes a random triplet t, u and w such that u and w are linked to v at time t, that u is linked to w at time t.

Notice that if there is no t such that $d_t(v) = 1$, *i.e.* node

v has always more than 1 neighbor, then:

$$cc(v) = \delta(N(v)),$$

the clustering coefficient of v is nothing but the internal density of its neighborhood.

Notice that the clustering coefficient of v in L_t is nothing but the clustering coefficient $cc_t(v)$ of v in the graph G_t. However, the average over time of $cc_t(v)$ is the probability, when one takes a random time instant t, that two random neighbors u and w of v at t are linked together at time t. Therefore, it is different from $cc(v)$.

We define the clustering coefficient of L as a whole as the average clustering coefficient of all the nodes when they have a degree superior to 1: $cc(L) = \frac{\sum_v cc(v)}{n}$.

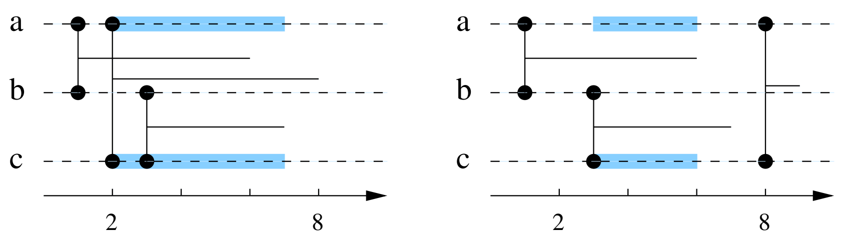


Figure 3.13: Two link streams L and L', where the neighborhoods of node b are depicted in blue. In L, the clustering coefficient of b is $cc(b) = \frac{[2,7]\cap[2,8]}{[2,7]} = 1$, since a and c always interact together when they interact with b. In L', $cc(b) = \frac{[3,6]\cap[8,9]}{[3,6]} = i|\emptyset| = 0$.

We define the transitivity ratio of L as the probability when one takes a random quadruplet t, u, v, w such that u and w are linked to v at t, that u and w are linked together at t. Notice that, for a given time instant t, the transitivity ratio of the stream induced by time t, $tr(L_t)$, is equal to the transitivity ratio of the graph induced by L at time t, $tr(G_t)$. However, $tr(L)$ is not the average over all t of $tr(L_t)$.

Table 3.1 summarizes the relations between the clustering coefficient and the transitivity ratio in link streams.

	Given t	Any t
Given v, any (u, w)	$cc_t(v) = cc(v)$ in G_t	$cc(v) \neq$ average of $cc_t(v)$ over t
Any (v, u, w)	$tr(L_t) = tr(G_t)$	$tr(L) \neq$ average of $tr_t(L_t)$ over $t \neq cc(L)$

Table 3.1: Relations between clustering coefficient and transitivity ratio in link streams. For example, if one is given a time $t \in T$ ("Given t") and a node $v \in V$ ("Given v"), one still needs to choose two random neighbors of v u and w ("any (u,w)"). This defines the clustering coefficient of v at time t, $cc_t(v)$, which is the clustering coefficient of v in the induced graph G_t (upper left cell).

3.5 Conclusion

In this chapter, we generalized notions from graph theory that stem from density. After extending density to link streams, we extend notions of neighborhood, degree, node clusters, cliques, clustering coefficient and transitivity ratio. Many concepts defined in this chapter are non trivial to compute. We illustrate this by studying in depth clique computation in Chapter 6.

Notice that a link stream may be seen as a node cluster where each node is in involved from α to ω. This paves the way for a more general definition of link streams, where nodes need *not* be present at all times in $[\alpha, \omega]$. This extension is more general than link streams, and is a promising perspective. It leads to more complex definitions, though, and do not allow the generalization of relations like the one between degree and density.

Chapter 4

Paths-based notions

In many contexts, graphs are used to study diffusion between nodes: delivering a message, or a file; assessing the resilience of a network to cable outages;predicting the spreading of a disease; etc. In all those cases, the underlying concept is the one of *path*, which is fundamental for defining notions of reachability and connectivity.

In many contexts, one is interested in assessing the importance of individual nodes or links: to identify key actors in social networks, or to target the weakest points of a network. Paths also lead to notions of centrality designed to capture such behaviour.

We discuss in this chapter these notions, and extend them to link streams.

4.1 Paths

In a simple graph $G = (V, E)$, a path P from $u \in V$ to $v \in V$ is a sequence (u_0, v_0), (u_1, v_1), $\dots$, (u_k, v_k) such that $u_0 = u$, $v_k = v$, and for all i, $v_{i-1} = u_i$ and (u_i, v_i) is in E. The integer k is the length of P. See Figure 4.1 for an illustration.

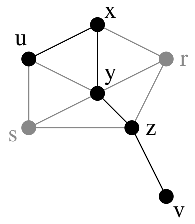


Figure 4.1: A graph $G = (\{u, x, y, \dots\}, \{(u, x), (s, y), \dots\})$ and a path P of length 4 from node u to node v; $P = ((u, x), (x, y), (y, z), (z, v))$.

In a simple link stream $L = (T, V, E)$, a γ-path P from node $u \in V$ to node $v \in V$ is a sequence $l_0 = (b_0, e_0, u_0, v_0)$, $l_1 = (b_1, e_1, u_1, v_1)$, $\dots$, $l_k = (b_k, e_k, u_k, v_k)$ such that $u_0 = u$,

$v_k = v$ and for all i, l_i is a sub-link of a link in E, $u_i = v_{i-1}$, $b_i \geq e_{i-1}$, and $e_i = b_i + \gamma$ [1]. We call γ the delay of links. For all i, we say that P involves u_i and v_i, and that it involves them at all t in $[b_i, e_i]$. See Figures 4.2 and 4.3 for examples.

1. Notice that (t, t, u, v) is a valid sub-link, if $\gamma = 0$.

We say that path P starts at t_0, arrives at t_k, has duration $t_k - t_0$ and length k. Given two nodes u and v, we say that a path from u to v of minimal length is a shortest path, that a path with minimal duration is a fastest path, and a path with minimal arrival time is a foremost path [2]. The length $d(u,v)$ of a shortest path from u to v is the distance from u to v, and the duration $\ell(u,v)$ of a fastest path is the latency from u to v. We denote by $\mathcal{T}_t(u,v)$ the time needed to reach v from u at time t, *i.e.* the difference between t and the minimal arrival time of a path from u to v starting after t. By extension, we denote by $\mathcal{T}_{b..e}(u,v)$ the time needed to reach v from u at time t with a path starting between b and e. In the remainder of this chapter, $\gamma = 0$ unless stated otherwise.

2. These properties can be cumulative: the shortest fastest path is, among all fastest paths, the shortest one; the fastest foremost is, among all foremost paths, the fastest one, and so on.

Notice that these notions have previously been defined by [Bui-Xuan et al., 2003], and we simply formulate them in the link stream framework.

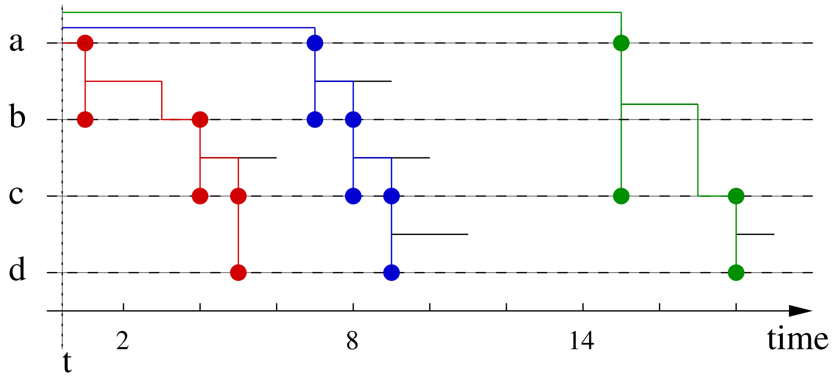


Figure 4.2: A link stream and 3 0-paths (*i.e.* $\gamma = 0$) from a to d from time t. Path $((1,3,a,b),(4,5,b,c),(5,5,c,d))$ (in red) arrives at time 5, has duration and length 3; it is a *foremost* path. Path $((7,8,a,b),(8,9,b,c),(9,9,c,d))$ (in blue) arrives at time 9 has duration 2 and length 3; it is a *fastest* path. Path $((15,17,a,c),(18,18,c,d))$ (in green) arrives at time 16, has duration 3 and length 2; it is a *shortest* path.

If $\gamma = 0$ then $b_i = e_i$ in the definition above, and a 0-path is equivalent to a sequence of triplets (t_0, u_0, v_0), (t_1, u_1, v_1), $\ldots$, (t_k, u_k, v_k) such that $u_0 = u$, $v_k = v$, and for all $i \in [1,k]$, $v_{i-1} = u_i$, $t_i \geq t_{i-1}$ and there is a link (b_i, e_i, u_i, v_i) in E such that $t_i \in [b_i, e_i]$.

Several variants of this notion of path in link streams

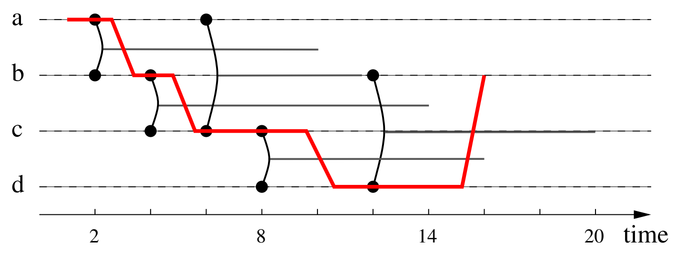


Figure 4.3: A link stream and a γ-path from a to b, with $\gamma = 1$ unit of time.

make sense. For instance, one may capture the fact that each node cannot forward information immediately after receiving it by adding the constraint $t_{i+1} \geq t_i + \gamma'$ in the definition, for a given γ'. Notice that this is *not* equivalent to $\gamma + \gamma'$-paths. One may also impose an overlap between successive links involved in a path: conditions above then become $b_{i+1} \leq e_i$.

Similarly, one may want to impose non-null delays on links and/or nodes but without bounds on these delays. Conditions above then become $e_i > b_i$ and $t_{i+1} > t_i$, respectively.

4.2 Connectivity

In a graph $G = (V, E)$, we say that node v is *reachable* from u if there is a path from u to v. Notice that, if G is undirected, reachability is symmetric: if u is reachable from v, then the reverse is true. If, for all $u, v \in V$, u is reachable from v, then G is connected. The largest subsets $V' \subseteq V$ such that the subgraph induced by V' is connected are called the connected components of G.

In link streams, we say that u is reachable from v if there is a path from v to u. Notice that reachability is not symmetric: u may be reachable from v while v is not reachable from u, see Figure 4.4. If for all u and v in V node u is reachable from node v then we say that L is connected.

Given $X \subseteq V$ and $[b, e] \subseteq T$, we say that $C = (X, b, e)$ is a connected component if the substream $L_{b..e}(X)$ is connected. In other words, for any two nodes u and v in X there is a path from u to v starting at b or later, arriving at e or earlier, and involving only nodes in X.

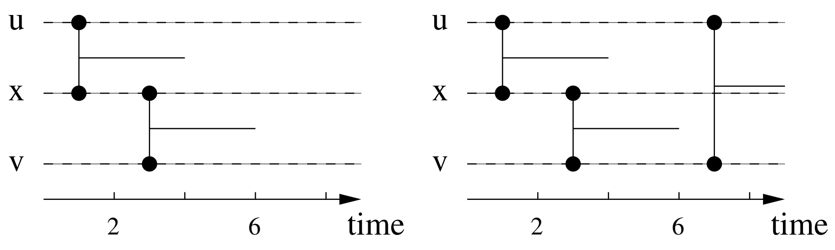


Figure 4.4: Two streams L and L'. In L, v is reachable from u, but the reverse is not true: there is no path from v to u. In L', u and v are both reachable from each other: v can reach u through the link $(7, 10, v, u)$.

Notice that, if $C = (X, b, e)$ is a connected component then $C' = (X, b', e')$ also is a connected component if $b' \leq b$ and $e' \geq e$. As a consequence, if (X, b, e) is a connected component then (X, α, ω) also is a connected component, and so maximal connected components are characterized by their set of nodes only. Conversely, we say that a connected component C is a connecting component if X is maximal and $[b, e]$ is minimal. See Figure 4.5 for an illustration.

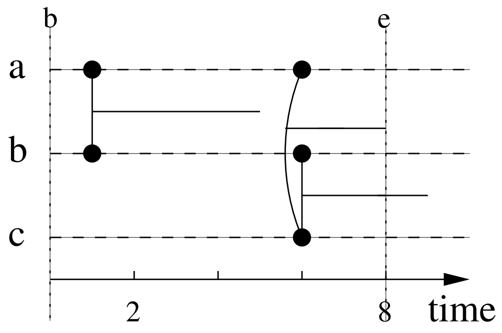


Figure 4.5: $(\{a, b, c\}, 0, 8)$ is a connected component: from time 0 to time 8, the three nodes are all reachable from each other. This is not true for any time $e' < 6$, as there is no path from c to a. We say that if the set of nodes is maximal and the time interval is minimal, the connected component is a connecting component: this is the case for $(\{a, b, c\}, 6, 6)$.

A connected part of L is a triplet (X, b, e) such that for any two nodes u and v in X there is a path from u to v starting at b or later and arriving at e or earlier in L. Notice that a connected component necessarily is a connected part. On the contrary, the paths between nodes in X may involve nodes outside X and so connected parts are not necessarily connected components, see Figure 4.6.

4.3 Centralities

It is usual to measure the importance of nodes and links in graphs using metrics called centralities. As there may be various criteria for this importance, several notions of

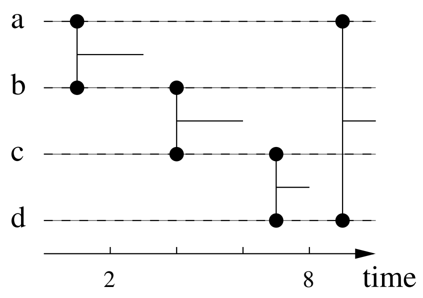


Figure 4.6: $A = (\{a,b,c\}, 0, 10)$ is a connected part: all nodes in A are reachable from each other. However, $B = (\{a,b,c,d\}, 0, 10)$ is not a connected part, since there is no path from d to b. Yet, if one considers the substream $L(\{a,b,c\})$, A is not a connected part. d is a connecting node.

centralities coexist. The most widely used ones probably are closeness and betweenness centralities. The closeness of a node v measures its proximity to other nodes: $\mathcal{C}(v) = \sum_{u \neq v} \frac{1}{d(v,u)}$, where $d(v,u)$ is the distance from v to u, *i.e.* the length of the shortest path from v to u. If there is no path from u to v, then the two nodes are infinitely far, and $d(u,v) = \infty$. The betweenness of a node v measures the number of shortest paths in the graph that involve v: $\mathcal{B}(v) = \sum_{x \neq y \neq v} \frac{\sigma(x,y,v)}{\sigma(x,y)}$ where $\sigma(x,y,v)$ is the number of shortest paths from x to y involving v and $\sigma(x,y)$ is the total number of shortest paths from x to y. Notice that the betweenness centrality is only defined for connected graphs. See Figure 4.7 for an illustration.

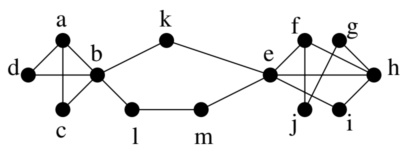


Figure 4.7: In this toy graph; nodes k, l and m have similar values of closeness centrality: they are the closest to every node in the graph. k has a high betweenness centrality: it is on all shortest paths between $\{a,b,c,d\}$ and $\{e,f,g,h,i,j\}$, whereas l and m have a low betweenness centrality.

4.3.1 Closeness centrality

In a link stream, we want the closeness of a node to capture how fast it may reach other nodes. We therefore define the closeness of node v at time t, for all $\gamma > 0$, as:

$$\mathcal{C}_t(v) = \sum_{u \neq v} \frac{1}{\mathcal{T}_t(v,u)}$$

and the closeness of node v in L as its average closeness over all the duration of L:

$$\mathcal{C}(v) = \frac{1}{\omega - \alpha} \int_{\alpha}^{\omega} \mathcal{C}_t(v) dt$$

In other words, the closeness of v is the average of the inverse of the time needed to reach a randomly chosen node from v at a randomly chosen instant. See Figure 4.8. If there is no path from u to v in the stream, then the two nodes are infinitely far, and $\mathcal{T}_t(u,v) = \infty$.

By convention, we consider that $\frac{1}{\infty} = 0$. If there is a link at time t between u and v, then $\mathcal{T}_t(u,v) = \gamma$.

By extension, we define the closeness of v in an interval $[b,e] \subset [\alpha,\omega]$:

$$\mathcal{C}_{b..e}(v) = \frac{1}{e-b}\int_b^e \mathcal{C}_t(v)dt$$

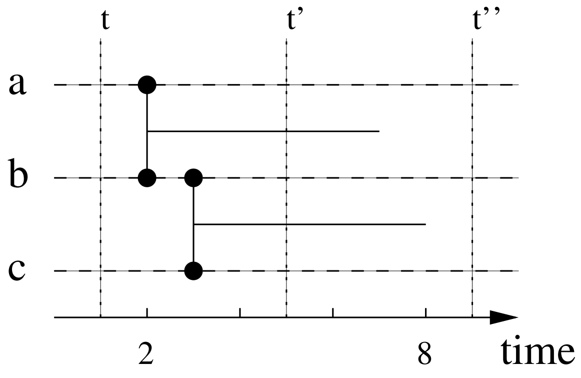


Figure 4.8: A link stream $L = ([0,10],\{a,b,c\},\{(2,7,a,b),(3,8,b,c)\})$. From time $t=1$, node a reaches node b at time 2, and node c at time 3. The closeness centrality of node a at time 1 is then $\mathcal{C}_t(a) = \frac{1}{2}+\frac{1}{3} = \frac{1}{6}$. At time $t' = 5$, there is a path from a to b and from a to c, and $\mathcal{C}_{t'}(a) = 2$. At time $t'' = 9$, there is no path from a to b nor from a to c, hence $\mathcal{C}_{t''}(a) = 0$, by convention.

4.3.2 *Betweenness centrality*

We then define the betweenness of node v in L as:

$$\mathcal{B}(v) = \sum_{x\neq y\neq v}\frac{\varphi(x,y,v)}{\varphi(x,y)} \tag{4.1}$$

where $\varphi(x,y,v)$ is the number of shortest fastest paths from x to y involving v.

We define the betweenness of node v from time b to time e as: $\mathcal{B}_{b..e}(v) = \sum_{x\neq y\neq v}\frac{\varphi_{b..e}(x,y,v)}{\varphi(x,y)}$ where $\varphi_{b..e}(x,y,v)$ is the number of shortest fastest paths from x to y involving v starting at b or before and arriving at e or later. Notice that $\mathcal{B}(v) = \mathcal{B}_{\alpha..\omega}(v)$.

Then, the betweenness of node v at time t is the fraction of shortest fastest paths between any two pairs of nodes involving v at time t :

$$\mathcal{B}_t(v) = \sum_{x\neq y\neq v}\frac{\varphi_t(x,y,v)}{\varphi(x,y)}$$

where $\varphi_t(x,y,v)$ is the number of shortest fastest paths from x to y involving v at time t and $\varphi(x,y)$ is the total number of shortest fastest paths from x to y in L.

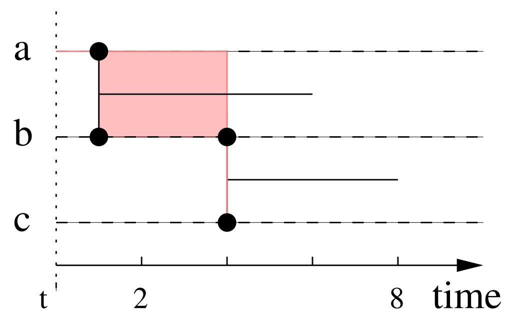


Figure 4.9: A link stream L. The number of shortest fastest paths from a to c from time t is depicted in red: it is all the paths $((x, 4, a, b), (4, 4, b, c))$ for $x \in [1, 4]$. We say that there are 3 units of time paths.

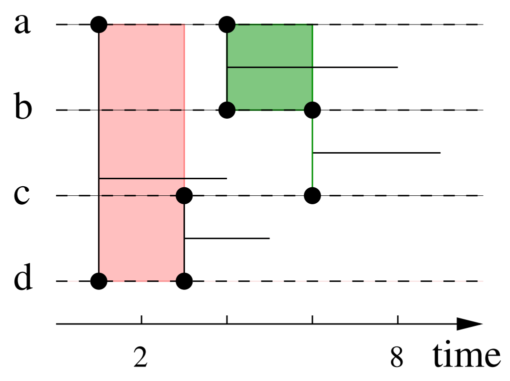


Figure 4.10: A link stream L. The number of shortest fastest paths from a to c involving b is depicted in green, and represents 2 units of time (from time 4 to time 6. The number of shortest fastest paths from a to c also contains the paths depicted in red, representing 2 units of time (from time 1 to time 3). Hence, $\frac{\varphi_t(a,c,b)}{\varphi_t(a,c)} = \frac{1}{2}$.

When all links have duration $\omega - \alpha$, the betweenness in the link stream is equal to the betweenness in the induced graph.

Notice that $\mathcal{B}(v)$ is *not* the average value of $\mathcal{B}_t(v)$ over $[\alpha, \omega]$: we do not take into account the time during which v is involved in considered paths. Indeed, we are interested in measuring the number of paths involving v; measuring this number weighted by the duration of the involvement of v may be done by computing $\frac{1}{\omega-\alpha} \int_\alpha^\omega \mathcal{B}_t(v)dt$.

The betweenness centrality we just defined evaluates the importance of nodes and time periods in function of very global objects: shortest fastest paths in the global stream, and the number of such paths in the whole stream. One may obtain a much more local view by studying betweenness in a substream $L_{b..e}$ of L. In between, one may study the fraction of all shortest fastest paths that begin and/or end between b and e, which is different from studying the substream $L_{b..e}$, see Figure 4.11.

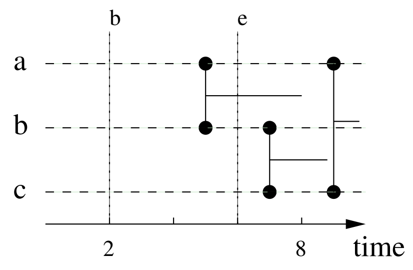


Figure 4.11: The fraction of all shortest fastest paths that begin and/or end between b and e is different from studying the substream $L_{2..6}$: indeed, $((5, 6, a, b), (7, 7, b, c))$ is a shortest fastest path from a to c starting before b, but is not contained in $L_{2..6}$.

4.4 k-closure

Given two nodes $u, v \in V$ and a time $t \in T$, we define the k-closure of (u, v) at time t, as the difference between the maximum time $t' < t$ such that there exists a path $P = ((t_i, u_i, v_i))_{i=0..n}$ of length $n \leq k$ with $t_0 \geq t'$, $u_0 = u$ and $v_n = v$ and $t_n \leq t$. We denote it by $\perp_k(u, v, t) = t - t'$ [3]. In other words, the k-closure of (u, v) is the minimum time one has to go in the past before t to find a path of length at most k from u to v in the stream. See example in Figure 4.12.

3. Notice that $\perp_k(u, v, t) \neq \perp_k(v, u, t)$.

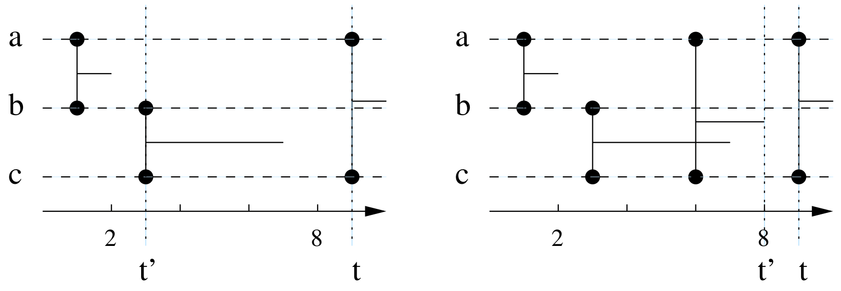


Figure 4.12: Two streams L and L'. In L, the 2-closure of (a, c) at time t is at time $t' = 3$: indeed, there is a path of length 2 from a to c at t'. In L', one finds $t' = 8$, since there is a path of length 1 from a to c at time 8.

4.5 Conclusion

In this chapter, we define notions of paths in link streams. Unlike graphs, where one is typically interested in shortest paths, we describe three types of interesting paths between two nodes: the *foremost* path, which is the first to arrive from a given time t, the *fastest*, which is the one that has the smallest duration, and the *shortest*, which minimizes the number of hops.

We define notions of reachability and connectivity, as well as metrics of centrality to assess the importance of nodes in the stream.

As we saw in Chapter 1, many works study path-related concepts in the literature. Studying how these notions can be unified with the ones presented in this chapter is a prospective work.

A particularly interesting future work is to generalize more complex notions of graph theory based on paths: for example cycles, trees (and spanning trees), as well as random walks, greedy walks or traversals of the link stream.

Chapter 5

Δ-analysis of link streams

In many contexts, the data is a sequence of instantaneous contacts (t, u, v) rather than of contacts with duration (b, e, u, v). This is the case, for example, of face-to-face contacts captured by sensors, or IP traffic studied as a sequence of packets.

However, the notions we have defined in Chapter 3 do not take into account such instantaneous links. The density, for instance, sums the durations of links: if links have duration 0, they account for nothing in the density. To avoid this, one generally considers a time scale Δ and defines Δ-related notions like the Δ-density, the Δ-degree, and so on. In this chapter, we define a transformation of a link stream to study it at a given scale Δ, and show the notions defined in Chapter 3 are equivalent to Δ-related notions.

5.1 Δ-analysis and instantaneous links

When studying a link stream L, one often wants to consider a time scale Δ coarser than the exact times of links in L. For instance, if L represents contacts between individuals, one may be interested in knowing if these individuals generally meet every day, every week or every year.

In particular, some data are typically in the form of instantaneous links, for example IP traffic: a packet exchanged between two machines u and v occurs exactly at a given time t.

One may see the parameter Δ as capturing the following intuition: if a link exists between two nodes within a time interval of duration at most Δ then one considers that the two nodes are continuously linked together during this time interval.

Given a link stream $L = (T, V, E)$ and a value $\Delta \in [0, \omega - \alpha]$, we define $L_\Delta = (T_\Delta, V, E_\Delta)$ as the link stream such that $T_\Delta = [\alpha + \frac{\Delta}{2}, \omega - \frac{\Delta}{2}]$ and $E_\Delta = (\{(b', e', u, v) : \exists(b, e, u, v) \in E, [b', e'] = [b - \frac{\Delta}{2}, e + \frac{\Delta}{2}] \cap T_\Delta\})$. In other words, any two nodes are linked together at time t in L_Δ if and only if they are linked together in L at a time t' such that $t \in [t' - \frac{\Delta}{2}, t' + \frac{\Delta}{2}] \cap T_\Delta$. This operation is depicted in Figure 5.1.

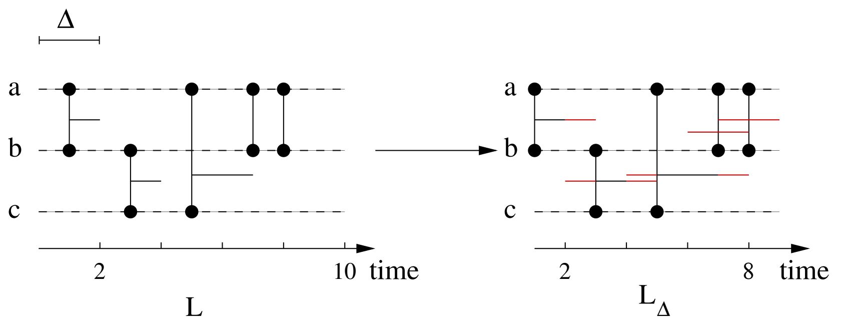


Figure 5.1: Δ-transformation of the link stream $L = ([0, 10], \{a, b, c\}, \{(1, 2, a, b), (3, 4, b, c), \dots\}$ for $\Delta = 2$. $L_\Delta = ([1, 9], \{a, b, c\}, \{(1, 3, a, b), (2, 5, b, c), \dots\})$.

5.2 Density

[Viard and Latapy, 2014a] define $\delta_\Delta(L)$ as the probability, when one takes a random time interval of size Δ in T and two random nodes u and v in V that these two nodes are linked at some time during this time interval. This is equivalent to the probability, when one takes a random time instant t in $[\alpha + \frac{\Delta}{2}, \omega - \frac{\Delta}{2}]$, that the two nodes are linked at some time in $[t - \frac{\Delta}{2}, t + \frac{\Delta}{2}]$ in L. By definition of L_Δ, this is equivalent to the probability, when one takes a time t in T_Δ, that the two nodes are linked at time t in L_Δ: if the nodes are linked together at some time in $[t - \frac{\Delta}{2}, t + \frac{\Delta}{2}]$ in L then they are linked together at time instant t in L_Δ, and conversely. The probability that this occurs is nothing but the density of L_Δ. Then, we have $\delta_\Delta(L) = \delta(L_\Delta)$.

5.3 Degree

We define the Δ-degree of a stream $d_\Delta(L)$ as the number of expected neighbors of a node u chosen at random when one takes a random interval of size Δ in T.

We then have the following relation between Δ-degree and Δ-density: $\delta_\Delta(L) = \frac{d_\Delta(L)}{n-1}$. As we have shown in Chapter 3, for any link stream L', $\delta(L') = \frac{d(L')}{n-1}$. Since the order of L is equal to the order of L_Δ, we conclude that $d_\Delta(L) = d(L_\Delta)$.

5.4 Clusters

One can associate to any cluster C a cluster C_Δ, which is the result of the same operation as the one transforming a stream L into L_Δ. However, by doing so the clusters of L_Δ do not form a partition of the stream, even if they formed a partition of L, and density and degree are not equivalent to the Δ-density and the Δ-degree.

To circumvent this, one can define Δ-clusters: C is a Δ-cluster if, for all $u \in V(C)$, and for all maximal intervals $[b,e]$ of $C(u)$, there is no (x,y,u,v) in E such that $b \leq x < \frac{\Delta}{2}$ or $e \geq y > e - \frac{\Delta}{2}$.

In other words, for each node in the cluster, there is at least $\frac{\Delta}{2}$ time between two implications of the node in two different clusters.

In the case of Δ-clusters, density and degree are respectively equivalent to Δ-density and Δ-degree.

5.5 Conclusion

In this chapter, we show that our formalism makes it possible to study link streams at a given scale Δ; we define a transformation of a link stream, and show that the notions of density, degree and clusters defined in Chapter 3 remain relevant on the transformed stream.

In our definition, Δ is a constant for all links in L, but the way we transform the stream is flexible: it is possible to set a value of Δ that is different for each (u, v), at different times, or that is a fraction of the duration of the link, for instance; Δ can also be fixed for each link (b, e, u, v) using external knowledge.

Chapter 6

Computing cliques in link streams

Enumerating maximal cliques of a graph is one of the most fundamental problems in computer science, and it has many applications [Rowe et al., 2007, Samudrala and Moult, 1998].

In Chapter 3, we have extended density and cliques to link streams. Enumerating the maximal cliques of a link stream raises algorithmic challenges.

As we have seen in Chapter 5, in many contexts, interactions are captured as sequences of (t, u, v). However, in this case, one needs to resort to a parameter Δ to study such interactions.

In this chapter, we present an algorithm to enumerate all maximal Δ-cliques of a link stream, implement it and apply it to a real-world contact trace.

6.1 Definitions

Let us consider a (simple) link stream $L = (T, V, E)$ with $T = [\alpha, \omega]$ and $E \subseteq T \times V \times V$: $l = (t, u, v)$ in E means that an interaction occurred between $u \in V$ and $v \in V$ at time $t \in T$.

For a given duration Δ, a Δ-clique C of L is a pair $C = (X, [b, e])$ with $X \subseteq V$ and $[b, e] \subseteq T$ such that $|X| \geq 2$, and for all $\{u, v\} \subseteq X$ and $\tau \in [b, \max(e - \Delta, b)]$ there is a link (t, u, v) in E with $t \in [\tau, \min(\tau + \Delta, e)]$ [1].

1. Notice that Δ-cliques necessarily have at least two nodes.

More intuitively, $(X, [b, e])$ is a Δ-clique if all nodes in X interact at least once with all others at least every Δ from time b to time e. Δ-clique C is maximal if it is included in no other Δ-clique, (*i.e.* there exists no Δ-clique $C' = (X', [b', e'])$ such that $C' \neq C$, $X \subseteq X'$ and $[b, e] \subseteq [b', e']$). See Figure 6.1 for an example.

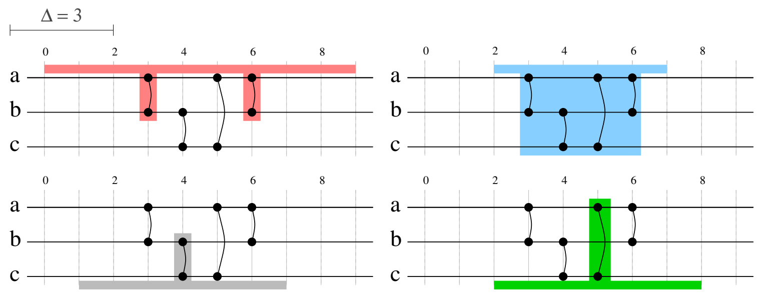


Figure 6.1: Examples of Δ-cliques. We consider the link stream $L = ([0,9], \{a,b,c\}, E)$ with $E = \{(3,a,b), (4,b,c), (5,a,c), (6,a,b)\}$ and $\Delta = 3$. There are four maximal 3-cliques in L: $(\{a,b\}, [0,9])$ (top left), $(\{a,b,c\}, [2,7])$ (top right), $(\{b,c\}, [1,7])$ (bottom left), and $(\{a,c\}, [2,8])$ (bottom right). Notice that $(\{a,b,c\}, [1,7])$ is not a Δ-clique since during time interval $[1,4]$ of duration $\Delta = 3$ there is no interaction between a and c. Notice also that $(\{a,b\}, [1,9])$, for instance, is not maximal: it is included in $(\{a,b\}, [0,9])$.

In real-world situations like the ones cited above, Δ-cliques are signatures of meetings, discussions, or distributed applications for instance. Moreover, just like cliques in a graph correspond to its subgraphs of density 1, Δ-cliques in a link stream correspond to its substreams of Δ-density 1, as defined in Chapter 5. Therefore, Δ-cliques in link streams are natural generalizations of cliques in graphs.

We propose an algorithm for listing all maximal Δ-cliques of a given link stream. We illustrate the relevance of the concept and algorithm by computing maximal Δ-cliques of a real-world dataset.

We define the first occurrence time of (u, v) after b as the smallest time $t \geq b$ such that $(t, u, v) \in E$, and we denote it by f_{buv}. Conversely we denote the last occurrence time of (u, v) before e by l_{euv}. We say that a link (t, u, v) is in $C = (X, [b, e])$ if $u \in X$, $v \in X$ and $t \in [b, e]$.

6.2 Algorithm

6.2.1 *Description of the algorithm*

One may trivially enumerate all maximal cliques in a graph as follows. One maintains a set M of previously found cliques (maximal or not), as well as a set S of candidate cliques. Then for each clique C in S, one removes C from S and searches for nodes outside C connected to all nodes in clique C, thus obtaining new cliques (one for each such node) larger than C. If one finds no such node, then clique C is maximal and it is part of the output. Otherwise, if the newly found cliques have not already been found (*i.e.*, they do not belong to M), then one adds them to S and M. The set S is initialized with the trivial cliques containing only one node, and all maximal cliques have been found when S is empty. The set M is used for memorization, and ensures that one does not examine the same clique more than once. In [Johnson et al., 1988] the authors use this framework to enumerate all maximal cliques of a graph in lexicographic order.

Our algorithm for finding Δ-cliques in a link stream $L = (T, V, E)$ (Algorithm 1) relies on the same scheme. We initialize the set S of candidate Δ-cliques and the set M of all found Δ-cliques with the trivial Δ-cliques $(\{a, b\}, [t, t])$ for all (t, a, b) in E (Line 2). Then, until S is empty (*while* loop of Lines 3 to 24), we pick an element $(X, [b, e])$ in S (Line 4) and search for nodes v outside X such that $(X \cup \{v\}, [b, e])$ is a Δ-clique (Lines 6 to 10). We also look for a value $b' < b$ such that $(X, [b', e])$ is a Δ-clique (Lines 11 to 16), and likewise a value $e' > e$ such that $(X, [b, e'])$ is a Δ-clique (Lines 17 to 22). If we find such a node, such a b' or such an e', then Δ-clique C is not maximal and we add to S and M the new Δ-cliques larger than C we just found (Lines 10, 16 and 22), on the condition that they had not already been seen (*i.e.*, they do not belong to M). Otherwise, C is maximal and is part of the output (Line 24).

Let us explain the choice of b' (Lines 11 to 16) in details, the choice of e' (Lines 19 to 22) being symmetrical. For a given Δ-clique $(X, [b, e])$, we set b' to $f - \Delta$, which is the smallest time such that we are sure that $(X, [b', e])$ is a Δ-clique without inspecting any link outside of $(X, [b, e])$. Indeed, all links in $X \times X$ appear at least once in the interval $[f - \Delta, f]$: f is the latest of the first occurrence times of all links in this Δ-clique, and so all links appear at least once

Algorithm 1: Maximal Δ-cliques of a link stream

```
input: a link stream $L = (T, V, E)$ and a duration $\Delta$
output: the set of all maximal $\Delta$-cliques in $L$
 1: $S \leftarrow \emptyset$, $R \leftarrow \emptyset$, $M \leftarrow \emptyset$
 2: for $(t, u, v) \in E$: add $(\{u, v\}, [t, t])$ to $S$ and to $M$
 3: while $S \neq \emptyset$ do
 4:     take and remove $(X, [b, e])$ from $S$
 5:     set isMax to True
 6:     for $v$ in $V \setminus X$ do
 7:         if $(X \cup \{v\}, [b, e])$ is a $\Delta$-clique then
 8:             set isMax to False
 9:             if $(X \cup \{v\}, [b, e])$ not in $M$ then
10:                 add $(X \cup \{v\}, [b, e])$ to $S$ and $M$
11:     $f \leftarrow \max_{u,v \in X} f_{buv}$        ▷ latest first occurrence time of a link in $(X, [b, e])$
12:     set $b'$ to $f - \Delta$
13:     if $b \neq b'$ then
14:         set isMax to False
15:         if $(X, [b', e])$ not in $M$ then
16:             add $(X, [b', e])$ to $S$ and $M$
17:     $l \leftarrow \min_{u,v \in X} l_{euv}$        ▷ earliest last occurrence time of a link in $(X, [b, e])$
18:     set $e'$ to $l + \Delta$
19:     if $e \neq e'$ then
20:         set isMax to False
21:         if $(X, [b, e'])$ not in $M$ then
22:             add $(X, [b, e'])$ to $S$ and $M$
23:     if isMax then
24:         add $(X, [b, e])$ to $R$
25: return $R$
```

in $[b, f] \subseteq [f - \Delta, f]$. If $b' \neq b$, then the Δ-clique $(X, [b', e])$ is added to S (Line 13).

We display in Figure 6.2 an example of a sequence of such operations from an initial trivial Δ-clique to a maximal Δ-clique in an illustrative link stream. The algorithm builds this way a set of Δ-cliques of L, which we call the *configuration space*; we display the configuration space for this simple example in Figure 6.3 together with the relations induced by the algorithm between these Δ-cliques.

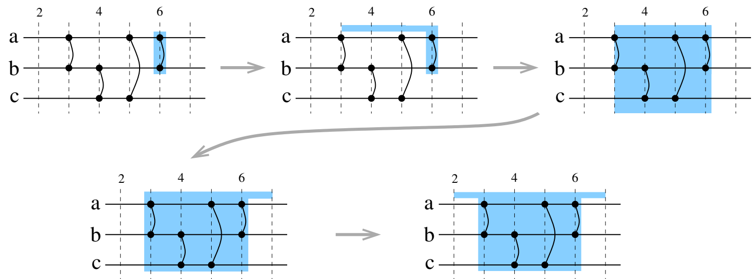


Figure 6.2: A sequence of Δ-cliques built by our algorithm to find a maximal Δ-clique (bottom-right) from an initial trivial Δ-clique (top-left) in the link stream of Figure 6.1 when $\Delta = 3$. From left to right and top to bottom: the algorithm starts with $(\{a, b\}, [6, 6])$, and finds $(\{a, b\}, [3, 6])$ thanks to Lines 11 to 16 of the algorithm. It then finds $(\{a, b, c\}, [3, 6])$ thanks to Lines 6 to 10. It finds $(\{a, b, c\}, [3, 7])$ from Lines 17 to 22, and finally $(\{a, b, c\}, [2, 7])$ from Lines 11 to 16.

6.2.2 *Proof of correctness*

To prove the validity of Algorithm 1, we must show that all the elements it outputs are Δ-cliques, that they are maximal, and that all maximal Δ-cliques are in its output.

Lemma 1. *In Algorithm 1, all elements of S are Δ-cliques of L.*

Proof. We prove the claim by induction on the iterations of the *while* loop (Lines 3 to 24).

Initially, all elements of S are Δ-cliques (Line 2). Let us assume that all the elements of S are Δ-cliques at the i-th iteration of the loop (induction hypothesis). The loop may add new elements to S at Lines 10, 16 and 22. In all cases, the added element is built from an element $C = (X, [b, e])$ of S (Line 4), which is a Δ-clique by induction hypothesis.

It is trivial (from the test at Line 7) that Line 10 only adds Δ-cliques.

Let us show that $(X, [b, l + \Delta])$, where l is computed in Line 17, necessarily is a Δ-clique. As $(X, [b, e])$ is a Δ-clique all links in $X \times X$ appear at least once every Δ from b to $l \leq e$. Moreover, since l is the earliest last occurrence time

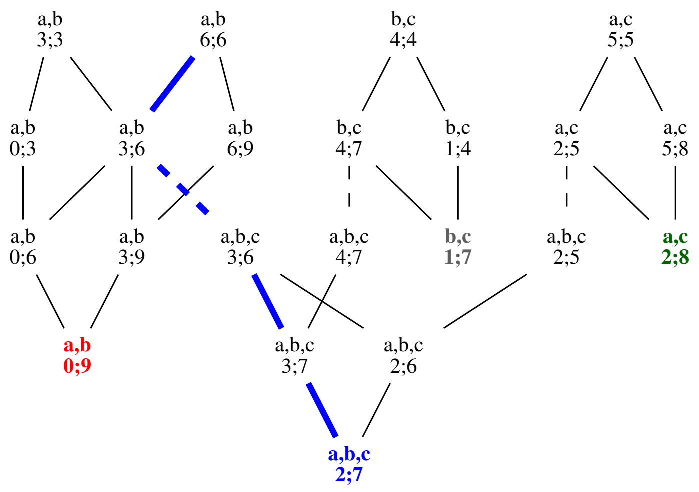


Figure 6.3: The configuration space built by our algorithm from the link stream of Figures 6.1 and 6.2 when $\Delta = 3$. Each element is a Δ-clique and it is linked to the Δ-cliques the algorithm builds from it (links are implicitly directed from top to bottom). Plain links indicate Δ-cliques discovered by Lines 11 to 16 or Lines 17 to 22 of the algorithm, which change the time span of the clique. Dotted links indicate Δ-cliques discovered by Lines 6 to 10, which change the set of nodes involved in the clique. The bold path is the one detailed in Figure 6.2. Colors correspond to the maximal Δ-cliques displayed in Figure 6.1.

of a link in C, for all u and v in X there is necessarily a link (t, u, v) in E with $l \leq t \leq e$. Notice also that $l \geq e - \Delta$, otherwise $(X, [b, e])$ would not be a Δ-clique. Therefore a link between u and v occurs at least once between l and $l + \Delta$ for all u and v in X. Finally, $(X, [b, l + \Delta])$ is a Δ-clique.

The same arguments hold for Line 11.

Finally, at the end of the $(i + 1)$-th iteration of the loop, all the elements of S are Δ-cliques, which ends the proof.

□

Lemma 2. *All the elements of the set returned by Algorithm 1 are maximal Δ-cliques of L.*

Proof. Let $C = (X, [b, e])$ be an element of R returned by the algorithm. Only elements of S are added to R (at Line 24), and so according to Lemma 1 C is a Δ-clique. Assume it is not maximal; then we are in one of the three following situations.

There exists v in $V \setminus X$ such that $(X \cup \{v\}, [b, e])$ is a Δ-clique. Then v is found at Lines 6–7, and Line 8 sets the boolean *isMax* to *false*. Therefore, Line 23 ensures that $C = (X, [b, e])$ is not added to R, and we reach a contradiction.

There exists $e' > e$ such that $(X, [b, e'])$ is a Δ-clique and we assume without loss of generality that there is no link between nodes in X from e to e'. Then, let us consider

$l \in [b,e]$, computed in Line 17, which is the earliest last occurrence time of a link in C. We necessarily have $l \geq e' - \Delta$ because $(X,[b,e'])$ is a Δ-clique. Since $e' > e$, this implies $l > e - \Delta$. As a consequence, the test in Line 19 of the algorithm is satisfied, and Line 20 sets the boolean *isMax* to *false*. Like above, we reach a contradiction.

If there exists $b' < b$ such that $(X,[b',e])$ is a Δ-clique, then similarly to the previous case we reach a contradiction.

Finally, C necessarily is maximal, which proves the claim. □

Before proving our main result, which is that all maximal Δ-cliques are returned by the algorithm, we need the following two intermediate results.

Lemma 3. *Let $C = (X,[b,e])$ be a maximal Δ-clique of L, and let s be the earliest occurrence time of a link in C. Then $e \geq s + \Delta$.*

Proof. Since C is a Δ-clique and by definition of s, for all u, v in X there exists at least one link (t,u,v) such that $s \leq t \leq e$. Assume $e < s + \Delta$; then for all u, v in X there also exists a link (t,u,v) such that $s \leq t \leq e < s + \Delta$. Therefore $(X,[b,s+\Delta])$ is a Δ-clique and C is included in it, which means that C is not maximal and we reach a contradiction. □

Lemma 4. *Let $C = (X,[b,e])$ be a maximal Δ-clique of L and let s be the earliest occurrence time of a link in C. If $(X,[s,s+\Delta])$ is in S at some stage of Algorithm 1, then C is in the set returned by the algorithm.*

Proof. Assume $C_0 = (X,[s,s+\Delta])$ is in S and consider the longest sequence of steps of Algorithm 1 of the form: $C_0 \to C_1 \to \cdots \to C_k$ such that for all i $C_i = (X,[s,e_i])$ with $e_{i+1} > e_i$. In other words, the algorithm builds C_{i+1} from C_i in Lines 17 to 22 (notice that $e \geq s + \Delta$ from Lemma 3 and so C_0 is included in C).

We prove that $C_k = (X,[s,e])$ by contradiction. Assume this is false, and so that $e_k \neq e$. As C is maximal, we then necessarily have $e_k < e$. In addition, $e_k = l + \Delta$ where l is the earliest last occurrence time of a link in C_{k-1} computed at Line 17. Since C_k is the last Δ-clique in the sequence, l is also the earliest last occurrence time of a link in C_k (otherwise there would be a clique C_{k+1} satisfying the constraints of the sequence above). Therefore there exist $u, v \in X$ such that $(l, u, v) \in E$ and such that there is no occurrence of a link (u, v) between l and $e_k = l + \Delta$. This ensures that there exists an ϵ such that $l + \Delta + \epsilon < e$ and such that there is no link between u and v from $l + \epsilon$ to $l + \Delta + \epsilon$, which contradicts the assumption that C is a Δ-clique.

We now show that the algorithm builds C from C_k to end the proof. Since C is maximal, there exists $u, v \in X$ such that $(b + \Delta, u, v) \in E$ and such that there is no other link between u and v from b to $b + \Delta$. By definition of s, $b + \Delta \geq s$. Therefore the latest first occurrence time of a link in C_k, f, is equal to $b + \Delta$ and Lines 11 to 16 build C from C_k. □

Lemma 5. *All maximal Δ-cliques of L are in the set returned by Algorithm 1.*

Proof. It is easy to check that if S contains a maximal Δ-clique then it is added to the set R returned by the algorithm, and only these Δ-cliques are added to R. We therefore show that all maximal Δ-cliques are in S at some stage.

Let $C = (X, [b, e])$ be a maximal Δ-clique of L, let s be the earliest occurrence time of a link in C, and let $u, v \in X$ be two nodes such that there exists a link between them at s (*i.e.*, $(s, u, v) \in E$). We show that there is a sequence of steps of the algorithm that builds C from Δ-clique $C_0 = (\{u, v\}, [s, s])$ (which is placed in S at the beginning of the algorithm, Line 2).

Lines 17 to 22 builds $C_1 = (\{u, v\}, [s, s + \Delta])$ from C_0.

Notice that for all subset Y of X, $(Y, [s, s + \Delta])$ is a Δ-clique. Therefore the algorithm iteratively adds all elements of X at Lines 6 to 10, finally obtaining $C' = (X, [s, s + \Delta])$ from C_1.

We finally apply Lemma 4 to conclude that the algorithm builds C from C'.

□

From these lemmas, we finally obtain the following result.

Theorem 1. *Given a link stream L and a duration Δ, Algorithm 1 computes the set of all maximal Δ-cliques of L.*

6.2.3 *Computational complexity*

In order to investigate the complexity of our algorithm, let us denote by $n = |V|$ the number of nodes and $m = |E|$ the number of links in L. First notice that the number of elements in the configuration space built by the algorithm is bounded by the number of subsets of V times the number of sub-intervals of T.

Moreover, for all Δ-clique $C = (X, [b, e])$ in the configuration space, there exists a link (b, u, v) or a link $(b + \Delta, u, v)$ in E. Indeed, the initial trivial Δ-cliques are in the first case, and all Δ-cliques obtained from them are also in this case until Line 16 is applied. The Δ-cliques built after this are in the second case. Likewise, there exists a link (e, u, v) or a link $(e - \Delta, u, v)$ in E. Therefore, the number of possible values for b and e for any Δ-clique in the configuration space is proportional to the number of time instants at which a link occurs, which is bounded by the number of links m. The number of sub-intervals of T corresponding to a Δ-clique in the configuration space is therefore in $\mathcal{O}(m^2)$. This bound is reached in the worst case, for instance if the stream is a sequence of links occurring once every Δ time interval.

The trivial bound $\mathcal{O}(2^n)$ for the number of subsets of V is also reached in the worst case, for instance if there is a link between all pairs of nodes at the same time: the algorithm will enumerate all subsets of V.

Therefore, the number of elements in the configuration space is in $\mathcal{O}(2^n m^2)$. This leads to the space complexity of our algorithm: it is proportional to the space needed to store the configuration space, which is in $\mathcal{O}(2^n n m^2)$ since each element may be stored in $\mathcal{O}(n)$ space.

We estimate the time complexity by studying the complexity of operations performed on each element of the configuration space, (*i.e.*, the complexity of each iteration of the *while* loop at Lines 3 to 24). Let us consider a Δ-clique $C = (X, [b, e])$ picked from S by the algorithm at Line 4.

The *while* loop is composed of three blocks: (1) searching for Δ-cliques of the form $(X \cup \{v\}, [b, e])$ larger than C (Lines 6 to 10); (2) searching for a Δ-clique $(X, [b', e])$ larger than C (Lines 11 to 16); and (3) searching for a Δ-clique $(X, [b, e'])$ larger than C (Lines 17 to 22). The third block has the same complexity as the second one, so we focus on the time complexity of the two first blocks.

Given a node $v \notin X$, Line 7 tests whether for all nodes u in X there is a link $(t, v, u) \in E$ in each time interval of duration Δ. This requires at most $|X| \cdot m$ tests, and so it is in $\mathcal{O}(nm)$. Then, Line 9 searches for the found Δ-clique in M, which has a size in $\mathcal{O}(2^n m^2)$. Since the comparison between two Δ-cliques can be performed in $\mathcal{O}(n)$ time, this search therefore is in $\mathcal{O}(n \log(2^n m^2)) = \mathcal{O}(n^2 + n \log m)$ time. The algorithm repeats these operations for all nodes $v \in V \setminus X$, and thus less than n times, hence the complexity of Lines 6 to 10 is in $\mathcal{O}(n(nm + n^2 + n \log m)) = \mathcal{O}(n^2 m + n^3)$.

Computing f in Line 11 may clearly be done with at most m tests. Lines 13 and 14 are all trivial computations. Lines 15 and Line 16 are in $\mathcal{O}(n^2 + n \log m)$. The complexity of Lines 11 to 16 is therefore in $\mathcal{O}(m + n^2 + n \log m)$.

Finally, each iteration of the while loop costs at most $\mathcal{O}(n^2 m + n^3 + m + n^2 + n \log m) = \mathcal{O}(n^2 m + n^3)$ time. We bound the overall time complexity of the algorithm by multiplying this by the number of iterations of the while loop, which is the number of elements in the configuration space. It is therefore in $\mathcal{O}(2^n m^2 (n^2 m + n^3)) = \mathcal{O}(2^n n^2 m^3 + 2^n n^3 m^2)$.

From this analysis, we obtain the following result:

Theorem 2. *Let $L = (T, V, E)$ be a link stream with $|V| = n$ and $|E| = m$, and let Δ be a duration, then Algorithm 1* *computes the set of all maximal* Δ*-cliques of* L *in* $\mathcal{O}(2^n nm^2)$ *space and* $\mathcal{O}(2^n n^2 m^3 + 2^n n^3 m^2)$ *time.*

Notice that enumerating the maximal cliques in a graph $G = (V, E)$ is equivalent to enumerating the maximal Δ-cliques in $L = ([0,0], V, E')$ where $(0, u, v) \in E'$ if and only if $(u, v) \in E$. The problem of enumerating maximal Δ-cliques in a link stream is therefore at least as difficult as enumerating maximal cliques in a graph, which has an exponential time complexity (in particular, there can be an exponential number of maximal cliques). Therefore any algorithm for enumerating maximal Δ-cliques in a link stream is at least exponential in the number of nodes.

Notice also that several optimizations may speed up our algorithm (without changing its worst-case complexity). In particular, f and l, computed in Lines 11 and 17, are necessarily in $[b, \min(e, b + \Delta)]$ and $[\max(b, e - \Delta), e]$, respectively. One may therefore focus the search on these intervals rather than $[b, e]$. Likewise, if $V(C)$ is the set of nodes satisfying condition of Line 7, then the set $V(C')$ of nodes satisfying this condition for the Δ-cliques C' added to S at Lines 10, 16 and 22 is included in $V(C)$. One may therefore associate to each element of S a set of candidate nodes to be considered at Line 6 in place of $V \setminus X$, thus drastically reducing the number of iterations of this loop.

6.3 Experiments

We implemented Algorithm 1 with the optimizations discussed above in Python (2.7) and provide the source code at [Viard and Latapy, 2014c]. We illustrate here its practical relevance by computing maximal Δ-cliques of a link stream representing real-world contacts between individuals, captured with RFID sensors.

6.3.1 Dataset

The trace Thiers-Highschool was collected at a French high school in 2012, see [Fournet and Barrat, 2014] for full details. It induces a link stream of 181 nodes and 45,047 links, connecting 2,220 distinct pairs of nodes over a period of 729,500 seconds (approximately 8 days). Each link (t, u, v) means that the sensor carried by individual u or v detected the sensor carried by the other individual at time t, which means in turn that these two individuals were close enough from each other at time t for the detection to happen. We call this a contact between individuals u and v. We also have the information of the class to which students belong.

6.3.2 *Impact of data structure*

Notice that Algorithm 1 makes no assumption on the order in which elements of S are processed, which corresponds to the way we explore the configuration space. In particular, if S is a first-in-first-out structure (a queue), the algorithm performs a BFS of the configuration space; if it is a last-in-first-out structure (a stack) then it performs a DFS. The execution time is essentially the same in all cases. The size of S may vary, but the space complexity of the algorithm is dominated by the size of M, that does not change. Still, the data structure impacts the order in which Δ-cliques are found.

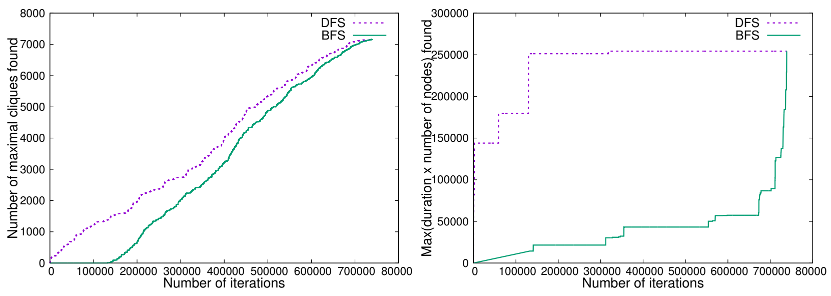


Figure 6.4: Behavior of our algorithm depending on the way it explores its configuration space (DFS or BFS). **Left**: number of maximal cliques discovered as a function of the number of iterations of the main loop of the algorithm. **Right**: maximal size of discovered cliques as a function of the number of iterations of the main loop of the algorithm. A clique size is estimated here by its number of nodes times its duration (in seconds).

We illustrate this in the practical case where $\Delta = 3600$ seconds (1 hour), see Figure 6.4. It shows that DFS rapidly discovers many cliques, and that those cliques are non-trivial cliques (cliques involving more than 2 nodes or lasting a substantial amount of time). In this case, using a DFS is therefore more interesting than a BFS, as it outputs results and exhibits non-trivial Δ-cliques faster. However, this behavior is dependent on the dataset, and deciding on the most appropriate exploration strategy in a given case remains an open question.

6.3.3 *Examining Δ-cliques*

We computed all maximal Δ-cliques for $\Delta = 60$ seconds, $\Delta = 900$ seconds (15 minutes), $\Delta = 3,600$ seconds (1 hour), and $\Delta = 10,800$ seconds (3 hours). We handpicked these values because of the rhythm of school day: on a typical day, courses usually last roughly two hours, with two 15 minutes breaks during the day, and a longer 1 hour lunch break. Our Python implementation took an hour on a standard server [2] to obtain the results. Although many discovered Δ-cliques are very small, we also found rather large and long ones. See Table 6.1 for a summary of these computations.

2. A Debian machine with a 2.9 GHz CPU and 64 GB of RAM.

Δ **(s)**	$\lvert R \rvert$	**Max** $\lvert X \rvert$	**Max** $e - b$ **(s)**	**Running time (s)**	**Memory (MB)**
60	14 664	5	6 820	150	537
900	8 214	7	17 420	555	4 755
3 600	7 170	7	36 340	1 080	23 186
10 800	7 416	7	59 560	3 100	30 453

Table 6.1: Experimental results for computing all maximal Δ-cliques on the THIERS-HIGHSCHOOL dataset. $|R|$ is the size of the set returned by our algorithm, (*i.e.*, the number of Δ-cliques found). For information, storing the dataset in RAM requires 51 MB.

We present in Figure 6.5, for each value of Δ, the complementary cumulative distributions for the size $|X|$ and duration $e - b$ of all maximal Δ-cliques $(X, [b, e])$. By definition, larger values of Δ trivially induce larger and longer Δ-cliques. Indeed, if $\Delta' > \Delta$ then every (maximal) Δ-clique also is a Δ'-clique (not maximal in general). More intuitively, small values of Δ detect local bursts, but are unable to find periodic behaviors if the period is larger than Δ. This is because for a larger value $\Delta' > \Delta$, small Δ-cliques tend to merge together into one larger Δ'-clique. Notice that when Δ grows the number of maximal Δ-cliques generally decreases, but this is not always true, as seen in Table 6.1. For an example of how the impact of Δ on the number of maximal Δ-cliques is not trivial, consider the stream presented in Figure 6.1: it contains four maximal 1-cliques, six maximal 2-cliques, and four maximal 3-cliques.

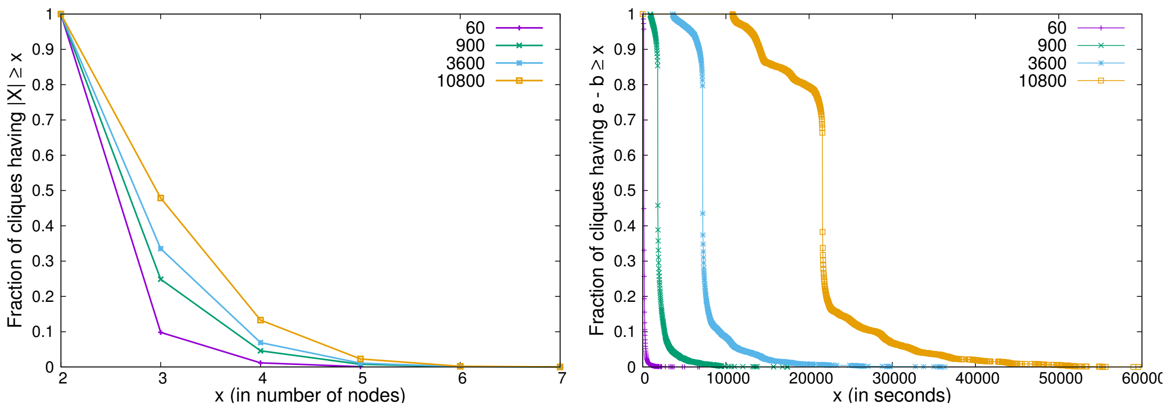


Figure 6.5: **Left:** complementary cumulative distribution of Δ-clique sizes for different values of Δ. **Right:** complementary cumulative distribution of Δ-clique durations for different values of Δ. The sharp drop at $2 \cdot \Delta$ is due to Δ-cliques involving only one link.

For every Δ, a sharp drop can be observed in the distribution of the durations (Figure 6.4, right); this corresponds to the value $2 \cdot \Delta$. This behaviour is typical of Δ-cliques involving only two nodes ($|X| = 2$). When Δ is larger, the proportion of such Δ-cliques drops, as there are less Δ-cliques of size 2, which is confirmed in Figure 6.5 (left). Conversely, the tails of the distributions get longer, as there are more larger and longer Δ-cliques.

Let us define the surrounding stream of a Δ-clique. For a maximal Δ-clique $(X, [b, e])$ and a stream $L = (T, V, E)$, the surrounding stream is the substream $L' = (T, V, E')$, with $E' = \{(t, u, v) \in E : t \in [b, e], u \in V, v \in V\}$. Intuitively, the surrounding stream contains all the links (t, u, v) having $t \in [b, e]$. We show in Figure 6.6 two typical Δ-cliques found in the contact trace, along with their surrounding stream. The tool for drawing link streams is available online [3]. It gives us an intuitive lookup onto the behaviours of the nodes involved in the Δ-cliques we found.

3. http://github.com/TiphaineV/LinkStreamViz

The 60-clique exhibited in Figure 6.6 (left) is a clique between 5 students of different classrooms, the largest size found for this Δ. The duration of this clique is very small (~3 minutes), and converting the timestamps to dates show these students were in contact during the lunch break (from 12:47 to 12:50). It is likely to be a group of friends that gathered before the beginning of the afternoon courses, or a meeting between the classroom's representatives.

The 60-clique shown in Figure 6.6 (right) is the longest one found by our algorithm: it lasts for 1080 seconds (18 minutes), and, surprisingly, involves two students from different classrooms. Considering the short range of the sensors, it is unlikely that this contact is due to students being close to each other, but in different rooms. The timestamp of the 60-clique, around 10am, likely points to an in-between courses discussion between two students.

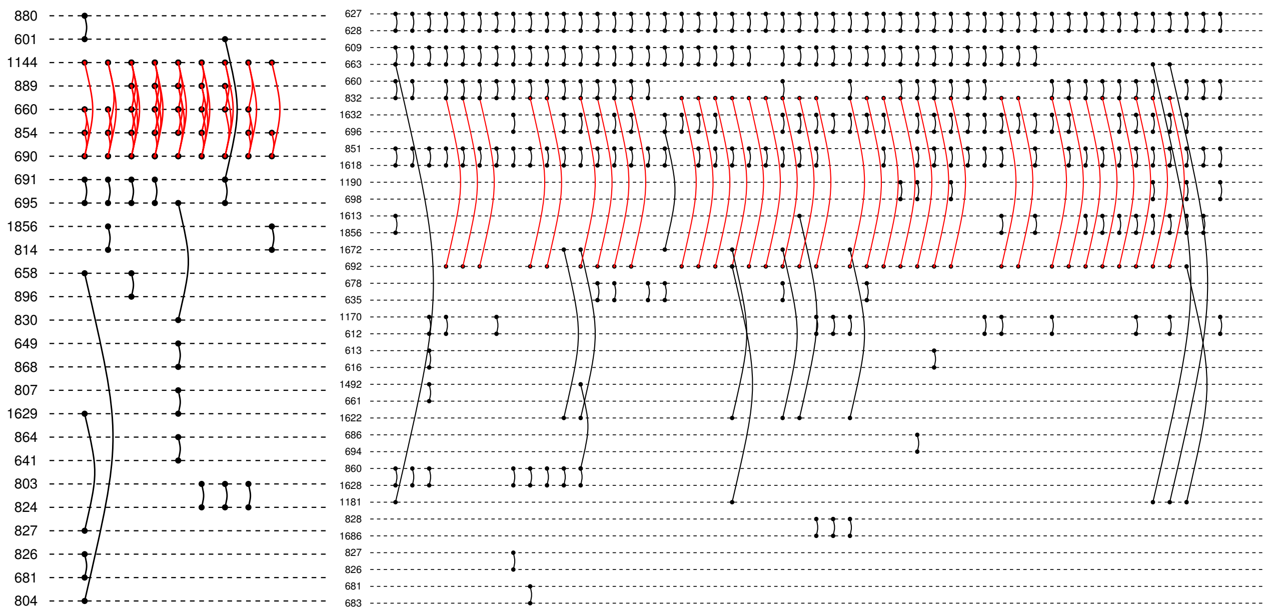


Figure 6.6: Two 60-cliques (in red) in their surrounding stream. **Left:** The 60-clique $(\{1144, 889, 660, 854, 690\}, 1353325660, 1353325820)$. For scale, the time between the two first occurrences of the link $(1144, 690)$ corresponds to 20 seconds. **Right:** The 60-clique $(\{832, 692\}, 1353920500, 1353921480)$. For scale, the time between the two first occurrences of the link $(832, 692)$ corresponds to 20 seconds.

We further investigate the descriptive role of Δ-cliques as compared to cliques in the aggregated graph. For a link stream $L = (T, V, E)$, as defined in Section 6.1, we define the aggregated graph $G(L) = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V} = V$ and $\mathcal{E} = \{(u, v) : \exists (t, u, v) \in E\}$. Intuitively, this is the graph obtained by putting a link between two nodes u and v if and only if there is at least one contact between u and v in the link stream.

The graph $G(L)$ contains 1742 cliques, the largest one involving 14 nodes.

Remember that in the aggregated graph, two nodes are linked if there is at least one contact between these nodes, regardless of the time and frequency of their interaction. As such, cliques in $G(L)$ are typically linking students in the same classroom. Most of the cliques (~70%) involve students of the same class.

If $(X, [b, e])$ is a (maximal) Δ-clique of L, then by definition X is a clique of $G(L)$ (in general not maximal). Considering Δ-cliques instead sheds light on different patterns.

6.4 Conclusion

We studied in this chapter the notion of Δ-clique in link streams, and proposed a first algorithm to compute the maximal such Δ-cliques. We have proved the correctness of our algorithm, and estimated its computational complexity: it runs in $\mathcal{O}(2^n nm^2)$ space and $\mathcal{O}(2^n n^2 m^3 + 2^n n^3 m^2)$ time.

We implemented this algorithm, released the source code online, and ran it on a real-world dataset of contacts between individuals of hundreds of nodes and thousands of links. The code runs in less than an hour, and detects interesting Δ-cliques that shed light on patterns of interaction between students, that cannot be found by studying the cliques in the aggregated graph.

Our algorithm may be significantly improved. Adapting the Bron-Kerbosch algorithm [Bron and Kerbosch, 1973] and some of its variants [Tomita et al., 2006, Koch, 2001, Cazals and Karande, 2008, Eppstein et al., 2013], the most widely used algorithms for computing cliques in graphs, is particularly appealing. Indeed, the configuration spaces built by these algorithms are trees, which avoids redundant computations. Notice that recent work by [Himmel et al., 2016] has already adapted the Bron-Kerbosch algorithm to link streams, significantly improving the performance clique computation.

Studying the influence of Δ on maximal Δ-cliques is an interesting perspective; given a link stream L and a clique $C = (X, [b, e])$ of L, finding the range $[\Delta_{min}, \Delta_{max}]$ such that, for all $\Delta \in [\Delta_{min}, \Delta_{max}]$, C is a maximal Δ-clique of L may give significant insight. For instance, if the range is small then it means that the cliques quickly merge into larger ones; on the contrary, if the range is large then it means cliques tend to be stable over time.

We also consider the case of links with duration as a promising perspective: each link (b, e, u, v) means that u and v interact from time b to e. In this case there is no need for a Δ anymore, and our algorithm may be directly adapted to compute cliques in duration link streams; however, the absence of Δ might simplifies the problem, and there may be a more straightforward algorithm.

CHAPTER 7

Conclusion and perspectives

In the first part of this thesis, we explored a new approach to model interactions over time with link streams. We extended fundamental notions from graph theory, and we devised a consistent framework, keeping the existing relations between notions. Our framework is distributed along two axes: one defines the notions derived from density, and another the notions derived from the notion of path.

Density is a fundamental notion in graph theory, that we extended to link streams. We also extended notions of neighborhood, degree, as well as clusters of nodes. These definitions give strong insights to describe and understand a link stream. Moreover, they lay foundations for defining more subtle notions defined on complex networks, such as communities, dense groups, and functions to evaluate partitions of the stream.

Studying paths is a key concept to anyone studying the connectivity of a link stream. Moreover, we defined notions of centrality to assess the importance of nodes and links at a given time in the link stream.

These new concepts raise rich algorithmic questions. We focused on clique computation, and presented a greedy algorithm that, given a duration Δ, finds all the maximal Δ-cliques in a link stream. We proved our algorithm's correctness, and used it to find maximal Δ-cliques in a real-world trace of contacts between individuals.

Some of the notions presented in Chapters 2 to 5 already exist in the literature; this is the case for instance of paths [Bui-Xuan et al., 2003], or closeness centrality [Pan and Saramäki, 2011]. Others, like density, or clusters, are completely new. To the best of our knowledge, however, it is the first time all these notions are embedded into one comprehensive and consistent framework.

An interesting perspective is to study the distributions of the notions we define on real-world link streams; this may enable to derive general properties of real-world link streams, and bring a better understanding of those objects.

A key goal of the link stream formalism is to extend both graph theory and signal processing into a single framework to describe sequences of interactions. In our work, emphasis has been put on notions from graph theory, but not from signal processing. An important perspective of this work is to bring notions from signal processing, and assess their meaning in link streams.

Fuzzy links, links that have a presence function over $[\alpha, \omega]$ are interesting from a theoretical point of view, yet few data embed such information. A fuzzy link can either be a tuple (b, e, u, v, x), where $x \in (0, 1]$ represents the presence of the link, of even a tuple (b, e, u, v, f), where f is a function of time.

Part II

Application to IP traffic

Introduction

Attacks against online services, networks, and information systems, as well as identity thefts, have annual costs estimated in billions of euros. These attacks also have dramatic consequences on the reliability of services and user trust. The techniques developed in the context of these malicious activities are of ever-growing complexity, and frequently rely on subtle malware (viruses or worms). Given this context, there is a critical need of methods and tools to fight against attacks and malware diffusion, and to give an appropriate answer to the major societal questions raised.

In the case of network traffic analysis, the data available is typically a capture of the traffic at one or many intermediary points in the network. A router is set up for capture, and will keep a record of all packets going through it. One obtains a stored sequence of packets, each indicating that two machines u and v interacted through the router at a time t. Typical captures of traffic also contain information about the ports and protocol used, the physical addresses, and so on.

Naturally, analysis of network traffic and more specifically event detection in such traffic has received a lot of attention. Studying the structure of interactions may be done through the lens of a graph, where one places a link (weighted or not) between two nodes if they have exchanged a packet [1]; another common approach consists in computing statistics on consecutive time windows; for example, the volume of traffic per second, or minute. From here, it is possible to resort to the powerful set of tools offered by signal processing. However, choosing an appropriate size for the time window is difficult, and the structure of interactions is completely lost.

1. The weight can then represent the number, or the frequency of interactions.

As we have seen in Part 1, however, these two approaches lose part of the information encoded in the interactions, and current approaches to extend them still call for much improvement. In this part of the thesis, our primary goal is to study traces of IP traffic as a link stream, and to assess the relevance of this model to detect and explain events in the traffic.

Recently, a substantial amount of research has been focusing on the design and evaluation of features describing the traffic, with the goal of providing these features to machine learning algorithms. The field calls for new features able to describe both structural and temporal aspects of IP traffic.

We briefly review the main works on the analysis of IP traffic in Chapter 8.

Given the nature of the data, the rich body of work developed in the past years and the important challenges remaining, it seems particularly appealing to model IP traffic as link streams. We discuss ways of modelling IP traffic in Chapter 9. It is worth noticing that to the best of our knowledge, this is the first work attempting to study IP traffic directly as a sequence of interactions.

We devise a method to study IP traffic and apply it to a trace of traffic in Chapter 10.

Chapter 8 Context

IP traffic has been widely studied from many different fields of research, including graph theory, information theory, signal processing, or, more recently, machine learning. The goal of this chapter is to describe some well-known sources of IP traffic, briefly review the main works in IP traffic analysis, and finally describe in depth the dataset we use in the remainder of this thesis.

8.1 Sources of data

8.1.1 CAIDA

The main goal of CAIDA [CAI, 2001] is to "investigate practical and theoretical aspects of the Internet", in order to provide insights on Internet infrastructures, behaviour, usage and evolution.

CAIDA provides a variety of datasets, resulting from active and passive measurement of the Internet. Available traffic traces range from 2001 to 2014, and their duration ranges from a few hours to a few days. While CAIDA provides unfiltered traffic, some traces focus on specific events and are curated to keep only the interactions relevant to this event [1]. All traffic is anonymized.

1. For instance, a 3-day trace covers the onset of the Confincker A infection.

8.1.2 MAWI

The WIDE (Widely Integrated Distributed Environment) project [wid, 1984] is one of the two main academic networks in Japan (with SciNet). It is responsible for the .jp top level domain, hosts the M root DNS server, and aims to integrate academia and industry into a single group. One of the realizations of this project is a testbed spanning across Japan.

It is a large network gathering thousands of users and a diversity of behaviours. Its academic nature also explains the presence of experimental technologies, such as IPv6 or DiffServ in their time.

The MAWI group project (Measurement and Analysis on the WIDE Internet) has been collecting traffic at 6 sample points since 1999. Everyday, 15 minutes of traffic are collected, anonymized and made public. Some longer traces (ranging from 24 hours to 83 hours) have also been collected as part of a Day in the Life of the Internet project.

8.1.3 USC ANT

The ANT Lab [ANT, 2002] is a research group at the University of Southern California, aiming at understanding the network topology and traffic, as well as its uses and misuses.

The traffic data provided by the ANT Lab is of three kinds: some traces are general traffic, collected at regional network access links, some traffic traces specifically contain known attacks on the network, and, finally, some traffic traces contain artificial attacks over real background traffic.

The ANT Lab offers traffic traces since 2002, and is still actively capturing traffic today. The traffic data contains packet headers, and IP addresses are anonymized. Duration of captures range from a few minutes to many years.

8.1.4 KDD Cup

Since 1999, KDD'99 [kdd, 1999] has been the most widely used dataset for evaluating anomaly detection methods. This dataset was prepared by [Stolfo et al., 2000] and built based on the data captured in DARPA'98 evaluation program.

The data contains 7 weeks of traffic, and 2 of them make up the test data. The remaining 5 weeks, used as training data, is given with the result (*e.g.* scalars, or vectors) of 41 features computed on the traffic, and each connection is labeled as "normal" or "attack". It is important to notice that the test data is not from the same probability distribution as the training data, and that specific attacks types are included in the test data but not in the training data, making the task more realistic.

The major drawback of this dataset is the fact that is is now 16 years old; however, this claim can be mitigated by knowing that some experts believe that most novel attacks are variants of known attacks.

8.2 Analysis of IP traffic

We now present the main works studying IP traffic. We breakdown the papers in three categories: papers studying the traffic as a (dynamic) graph, papers resorting to signal processing to analyze time series extracted from the traffic, and finally machine learning approaches.

8.2.1 Graph-based

Given that the traffic encodes important structural information, it is natural to model it as a graph, where the nodes are typically IP addresses, or couples constituted of the IP address and the port number. The general scheme is that one places a link between two nodes if they have interacted together at one point during the capture.

[Xu et al., 2014] model the traffic as a bipartite graph and study the one-mode projection, and apply graph clustering methods to differentiate end-host (*i.e.* clients) from application (*i.e.* server) behaviour. The core idea is that one-mode projections reveal relationships between nodes of the same vertex set.

[Iliofotou et al., 2007] have introduced Traffic Dispersion Graphs (TDGs) as a flexible graph model for IP traffic. A TDG is a graph $G = (V, E)$ where nodes $u, v \in V$ are distinct IP addresses, and there is a link $(u, v) \in E$ if the interaction between u and v match a given rule. The rule can be as simple as "there is at least 1 packet between u and v", but also more complex, "at least 3 TCP packets were exchanged at port 53". In further work, [Iliofotou et al., 2009a] extend TDGs to create sequences of TDGs snapshots in an attempt to study the dynamics of the network.

Work by [Latapy et al., 2013] has applied statistical anomaly detection methods to internet topology measurements. The methodology is based on the assumption that for some statistics computed on the data (for instance, the number of distinct IP addresses present between t and $t + \Delta$), the distribution is either homogeneous (*i.e.* there are no events), heterogeneous (*i.e.* the notion of event is irrelevant), or homogeneous with outliers. Though the data is very different from IP traffic, the methodology designed can be applied, which is why we cite it here.

8.2.2 Time-series analysis

A signal provides an coarse view of the traffic, by focusing on the evolution over time of one of its characteristics.

Entropy is probably the most common metric in this research domain, since it helps quantify how traffic is distributed in a specific feature space. For instance, entropy can measure whether the traffic is concentrated on a single IP address, or fairly distributed across the IP addressing space.The work by [Nychis et al., 2008] has studied the entropy of a large number of traffic features, and assessed their relevance for the analysis of the traffic. Using information-theoretic techniques, [Xu et al., 2005] define a methodology to extract relevant clusters of IP addresses and classify the IP addresses according to their behaviour.

In [Abry and Veitch, 1998] that wavelets efficiently model the long-range dependence [2] present in IP traffic. [Abry et al., 1998] propose an aggregation method that keeps some properties of the wavelet regardless of the size of the aggregation window. In later work, [Abry et al., 2002a] demonstrates the multiscale nature of IP traffic.

2. A phenomenon is usually considered to have long-range dependence if the dependence decays more slowly than an exponential decay, typically a power-like decay.

[Barford et al., 2002] and [Kim and Reddy, 2008] make use of wavelet filters to expose the details of both normal and anomalous traffic, and detect statistical anomalies by studying correlations in the packet header.

[Heymann et al., 2012] study the impact of removing points on the skewness of a distribution, and consider that a value is an outlier if removing it leads to a more symmetric distribution. [Brauckhoff et al., 2012] study the distributions of traffic-related features for anomaly detection.

In order to cluster the IP space, sketches are commonly used. The core idea is to use a hashing key such as source IP or destination IP, dividing a set of traffic data into subgroups, or sketches. In [Dewaele et al., 2007], the authors devise an anomaly detection method relying on sketching and multi-resolution gamma modelling. The traffic is split into sketches, and then modelled with gamma distributions. The traffic that is distant from a computed reference is reported as anomalous. [Li et al., 2006] apply the subspace method defined by [Lakhina et al., 2005] to sketches to precisely identify anomalous IP flows rather than IP addresses.

Other approaches rely on finer-grained signal processing techniques [Guralnik and Srivastava, 1999a], or on equilibrium properties [Silveira et al., 2010].

8.2.3 Machine-learning

Machine learning is a field exploring the study and construction of algorithms that can learn from and make predictions on the data. These algorithms are typically classified in 3 kinds: supervised learning, semi-supervised learning and unsupervised learning.

The lack of labelled datasets, and the prohibitive cost of building one makes unsupervised techniques particularly appealing. Yet, hunting anomalies "in the dark" is much more challenging, and semi-supervised are a good trade-off between cost and feasibility.

Machine learning algorithms rely on features (*i.e.* metrics computed on the considered dataset) to classify elements of a given dataset. These algorithms work on the vectorial space induced by the different features, and attempt to find clusters, or deviating values in this vectorial space.

Notice that the works presented in sections 8.2.1 and 8.2.2 provide metrics that could be considered as features of machine learning algorithms.

Typical features extracted from the traffic are the number of packets, of bytes exchanged, of distinct IP addresses. Some more subtle features are imported from signal processing [Dewaele et al., 2007], or even image pattern recognition [Fontugne and Fukuda, 2011].

Multiplying the number of features is tempting; however, features can be seen from a geometrical perspectives as dimensions, and one is then faced with high dimensional data. [Ankerst et al., 1999] points out 4 major challenges caused by high dimensionality:

1. The number of dimensions make it hard-to-impossible to visualize, or enumerate the space of features.

2. Notions of neighborhood and distance lose meaning as the number of dimensions grows. Indeed, the relative distance between the nearest and farthest point converges to 0 when the number of dimensions increases.

3. Highly dimensional data means that a high number of features were used. This implies that features might be correlated to each other, and that clusters might exist in arbitrarily oriented subspaces.

4. The high number of features enhances a problem known as "local feature relevance": different clusters might be

found in different subspaces, reducing the efficiency of global attribute filtering.

Instead, works have attempted to reduce the number of dimensions, and use the fact that a cluster of dimension n is also a cluster in all subspaces of dimension $k < n$.

For example, work by [Lakhina et al., 2004] is based on the use of Principal Component Analysis (PCA) to identify anomalies in large traces of network traffic. The authors represent the traffic as a matrix, where each cell (i, j) of the matrix contains the volume of traffic exchanged between machines i and j over a given time interval. PCA is then used to extract the main components from the matrix, and anomalies are detected in the residual traffic. This seminal paper has drawn attention on the use of PCA in the context of anomaly detection in IP traffic: its main shortcomings have been identified, and considerable improvements have been published [Kanda et al., 2010, Rubinstein et al., 2009].

The Hough transform is a pattern recognition technique aiming at finding specific shapes in pictures. In their paper, [Fontugne and Fukuda, 2011] models the traffic as a 2D scatter plot where anomalous traffic appears as "lines", and then uses the Hough transform to identify the anomalies.

[Brauckhoff et al., 2012] computes the Kullback-Leibler divergence to several histograms monitoring distinct traffic features. Then, association rule mining permits the extraction of the set of traffic features associated to the anomalies detected in the histograms.

[Zander et al., 2005] propose an unsupervised method to classify the traffic according to its statistical characteristics, and devise a systematic approach to identify ideal sets of traffic features to classify IP traffic. [Dainotti et al., 2011, Nguyen et al., 2012] use similar approaches, again with the goal of classifying the traffic.

[Bhuyan et al., 2014, Fernandes and Owezarski, 2009] and [Mazel et al., 2015] devise unsupervised methods clustering the data to identify anomalies without any previous knowledge of the data.

Finally, [Marnerides et al., 2014] published a survey focused on anomaly detection in IP traffic.

Providing relevant features to machine learning algorithms is a vivid area of research, and one of the applicative goals of the link stream framework is to provide new features that induce no loss of information.

8.3 The MAWI dataset

In this section, we present in details the MAWI dataset, which we chose to study in the remainder of this thesis.

8.3.1 *Traffic captures*

We use the data gathered on June 25, 2013 as part of the experiment "A Day in the Life of the Internet" (DITL). The data was collected (using port mirroring on a router) on a transit link between WIDE and the upstream ISP, see Figure 8.1. Due to the nature of the capture, the resulting network is intrinsically bipartite: traffic between two WIDE machines (or between two "Internet" machines) cannot be seen at this router, and is not captured. Thanks to expert knowledge from the maintainers of the MAWI repository, we can precisely know which machines are in and out of WIDE.

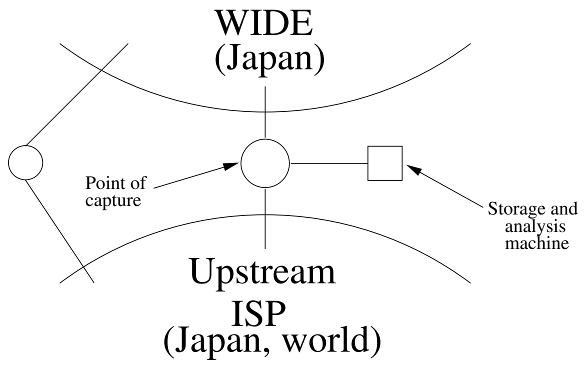


Figure 8.1: Capture of the traffic between WIDE and the upstream ISP. Packets circulate through the point of capture, and port mirroring copies them to a dedicated machine where the capture is stored.

The original data comes in the form of raw *tcpdump* files of 15 minutes each, that we merge to obtain a dump file containing one hour of traffic, from 00 : 00 to 01 : 00 JST (Japan Standard Time). For an example of IP traffic, see Figure 8.3.1. Choosing to keep 1 hour of traffic out of the 72 available in the capture was a trade-off between having enough data to obtain significant results, and keeping computation times to a few hours at most.

Example of IP traffic from the MAWI dataset

```
07:00:00.915688 IP 197.247.88.98.25197 > 149.173.188.128.6881: Flags [.], seq 1495724409:1495725789, ack 1528736985, win 64522, length 1380
07:00:00.915816 IP 149.53.11.191.1025 > 59.201.100.146.53: 38455
07:00:00.915938 IP 197.247.88.98.25197 > 149.173.188.128.6881: Flags [.], seq 1380:2760, ack 1, win 64522, length 1380
07:00:00.915944 IP 202.117.132.136.16809 > 214.2.100.173.53: 62301
07:00:00.916065 IP 80.58.88.3.53984 > 164.89.55.232.80: Flags [.], ack 2869520795, win 33210, length 0
07:00:00.916313 IP 197.247.88.98.25197 > 149.173.188.128.6881: Flags [.], seq 2760:4140, ack 1, win 64522, length 1380
07:00:00.916438 IP 215.13.122.90 > 214.130.120.245: GREv5 ERROR: unknown-version
07:00:00.916687 IP 197.247.88.98.25197 > 149.173.188.128.6881: Flags [.], seq 4140:5520, ack 1, win 64522, length 1380
07:00:00.917062 IP 197.247.88.98.25197 > 149.173.188.128.6881: Flags [.], seq 5520:6900, ack 1, win 64522, length 1380
07:00:00.917186 IP 214.2.97.241.53 > 192.168.96.4.11831: 31095-
07:00:00.917437 IP 197.247.88.98.25197 > 149.173.188.128.6881: Flags [.], seq 6900:8280, ack 1, win 64522, length 1380
07:00:00.917689 IP 215.37.127.61.80 > 219.217.167.126.46261: Flags [.], seq 1840757696:1840759144, ack 132375684, win 33304, length 1448
07:00:00.917696 IP 215.37.127.61.80 > 219.217.167.126.46261: Flags [.], seq 1448:2896, ack 1, win 33304, length 1448
07:00:00.917702 IP 215.37.127.61.80 > 219.217.167.126.46261: Flags [.], seq 2896:4344, ack 1, win 33304, length 1448
07:00:00.917708 IP 197.247.88.98.25197 > 149.173.188.128.6881: Flags [.], seq 8280:9660, ack 1, win 64522, length 1380
07:00:00.917714 IP 200.79.252.53.3730 > 215.37.113.218.3124: Flags [.], ack 3098387854, win 65535, length 0
07:00:00.917816 IP 215.37.124.41.28344 > 59.32.86.117.6972: UDP, length 3
07:00:00.917937 IP 133.208.154.172.19101 > 59.13.59.106.4095: Flags [.], seq 1638582278:1638583738, ack 1061949299, win 64667, length 1460
07:00:00.917943 IP 215.37.124.41.28344 > 59.32.86.117.6972: UDP, length 1017
```

Figure 8.3.1: Example of IP traffic; Each line represents a packet. The first field is the time of the capture, with microsecond precision, followed by the layer 3 protocol . The next fields show the IP addresses involved and the port number. The ">" shows the direction of the packet, and the rest is network-specific information.

As stated on the MAWI website, the captures contain a lot of ICMP traffic. This is due to the USC ANT project, that probes the entire IPv4 space. Since this traffic has a clear origin, we discard it by filtering the two IP addresses associated with the ANT project.

Moreover, the data is bipartite by nature, but a few packets break this bipartite nature, due to measurement errors [3]. We discard them too.

3. They represent less than 0.001% of the traffic

8.3.2 MAWILab event database

Besides from the MAWI dataset, an initiative called *MAWILab* [Fontugne et al., 2010a] aims at providing a public repository of events detected in each MAWI trace.

MAWILab reports events by combining the results of 4 different unsupervised detectors presented in section 8.2 of this chapter: Hough, Gamma, PCA and Kullback-Leibler.

Since these detectors output scores on different ranges, further work by [Fontugne et al., 2010b] is made to obtain comparable scores.

Moreover, [Mazel et al., 2014] designed a taxonomy of more than a 100 categories revealing the nature of backbone traffic anomalies; this taxonomy has been applied to the *MAWILab* events.

The *MAWILab* event database facilitates the work of anyone wishing to compare results from an anomaly detector to an established truth. Nevertheless, notice that the events on *MAWILab* are not a ground truth on the traffic: some events in the traffic may have been missed by all detectors (*false negatives*), and some events reported on *MAWILab* may not be events at all (*false positives*).

Chapter 9
MAWI traffic as a link stream

IP traffic, and in particular traffic from the MAWI dataset, has idiosyncrasies; for instance, it comes as a collection of IP packets, and is bipartite by nature.

The goal of this chapter is to adapt the link stream formalism introduced in part 1 to the modelling of collections of IP packets from the MAWI dataset.

9.1 Traffic modelling

The data from MAWI comes under the form of a collection of packets. One can then either study this sequence directly, or rebuild the IP flows from the original data and study them. We will now detail those two possibilities.

Studying the sequence of packets is appealing, since it is the raw form of the data. We remove all port, protocol, flag and direction information from the data, and extract from the collection of packets a sequence of triplets (t, u, v) meaning that machines u and v exchanged a packet at time t. Keeping more information, such as the ports numbers, protocol information, etc. is also interesting, yet in our work we focus on being as generic as possible. We discuss this in Section 9.3 of this chapter.

However, recall that in order to study instantaneous interactions, one has first to choose a value of Δ in order to transform the sequence of interactions into a link stream, as we have discussed in Chapter 5.

From a collection of packets $\mathcal{D} = \{(t,u,v)\}$ and a duration Δ, we obtain the link stream $L'_\Delta = (T,V,E_\Delta) = \{(t-\frac{\Delta}{2}, t+\frac{\Delta}{2}, u, v) : \exists(t,u,v) \in \mathcal{D}\}$; we then simplify L'_Δ to obtain the link stream L_Δ, as explained in Chapter 5. See Figure 9.1 for an illustration.

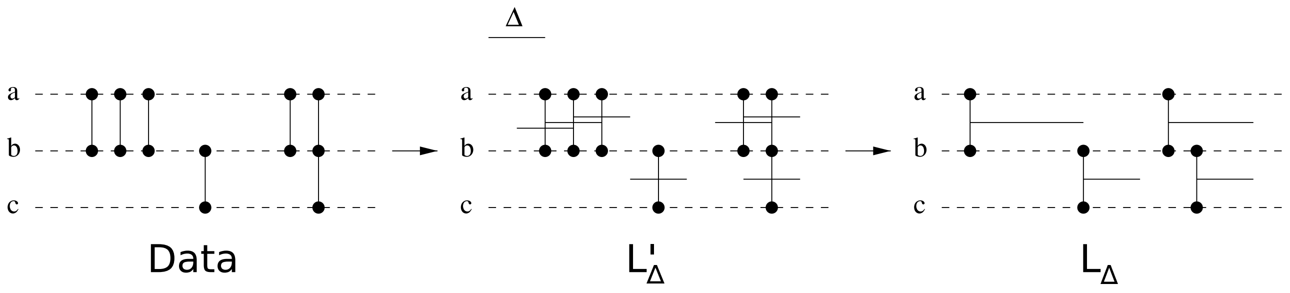


Figure 9.1: Obtaining a link stream from a collection of (t,u,v) (left). Given a Δ, one creates links from $t-\frac{\Delta}{2}$ to $t+\frac{\Delta}{2}$ (middle). However, the obtained link stream is not simple, as there might be overlaps between links, and so it is made simple (right).

Studying IP flows, one can naturally build a link stream in which a link (b,e,u,v) means that there is an IP flow from time b to time e between machines u and v [1]. Several methods and tools are used to build flows from IP packets, NetFlow and sFlow being the most common ones.

Besides its relevance, studying IP flows instead of packets is particularly interesting when considering high-speed traffic, since it significantly reduces the size of the data [2].

However, most flows contain less than 2 packets [3]. Nevertheless, many patterns of interest are constituted of 1 or 2 packet flows: a scan machine typically sends one packet to all the IP addresses in a subnetwork, and waits for replies from active machines in this subnetwork. In this case, exchanges between that machine and any other in the network are flows of 1 (if the machine does not reply) or 2 packets (if the machine replies).

A first analysis of the duration link stream induced by IP flows showed us that it was preferable to resort to packet analysis with a Δ. Indeed, in a duration link stream, all links (b,e,u,v) with $b = e$, *i.e.* flows consisting of one packet, account for nothing in the density, degree, clustering and so on.

1. Flows are unidirectional, so any TCP connection will create at least 2 flows.

2. From 88,266,535 packets (600MB, gzipped) to 2,504,106 flows (47MB, gzipped) for the 1 hour traffic trace we consider.

3. Flows with less than two packets account for 96% of flows in the dataset we use.

As we discussed in part 1, Chapter 1, choosing an appropriate value for Δ is an open problem [Abry et al., 2002a]. Adequately identifying relevant values for Δ is a direction of work that [Viard and Latapy, 2014a] has only superficially explored, and that is available in Appendix B of this thesis; in this exploratory work, we fix $\Delta = 2$ seconds.

9.2 Bipartite link streams

As explained in section 8.3 of this chapter, the data collected on a single router is intrinsically bipartite (*i.e.* nodes can be separated into two disjoint sets $\top$ and $\bot$ such that there is no link in E between two nodes of $\top$ or two nodes of $\bot$). This bipartite nature has huge implications on the underlying structure of the interactions. Indeed, some links cannot exist, there are no triangles, etc. We therefore extend some of the notions defined in Part 1 of this thesis to take this bipartite nature into account.

A bipartite graph is a tuple $G = (\top, \bot, E)$, with $\top$ and $\bot$ sets of nodes such that $\top \cap \bot = \emptyset$, and $E \subseteq \top \times \bot$. In other words, links can only exist between one node of $\top$ and one node of $\bot$.

The density is the probability when one takes two random nodes $u \in \top$ and $v \in \bot$ that there is a link (u, v) in E: $\delta(G) = \frac{|E|}{|\top| \cdot |\bot|}$.

The clustering coefficient of a node $v \in \bot$ is the probability, when one takes three nodes $x \in \top, y \in \bot$ and $z \in \top$ such that $(v, x), (x, y)$ and (y, z) are in E, that (v, z) is in E. This probability is nothing but the density of the graph induced by the neighbors of v and the neighbors of these neighbors, *i.e.* the density of the graph induced by $N(v) \cup N(N(v) \setminus \{v\})$. See Figure 9.2. Notice that $cc(v)$ is undefined if v has degree 1 or if $N(N(v)) = \emptyset$.

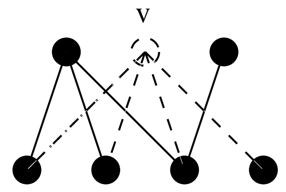


Figure 9.2: Illustration of the clustering coefficient in bipartite graphs. The clustering coefficient of node v is the density of the graph induced by the neighbors of v and the neighbors of these neighbors, deprived of v. It is the density of the graph in solid lines in this figure (links between v and its neighbors are represented in dashed lines).

The specific structure of bipartite graphs calls for new metrics that have no counterpart in non-bipartite graphs. Work by [Latapy et al., 2008a] surveys key metrics for bipartite graphs. We adapt two of them to link streams: projection and redundancy.

In order to benefit from the large number of works on non-bipartite graphs, the projection of a bipartite graph is defined as follows: given a bipartite graph $G = (\top, \bot, E)$, its $\bot$-projection [4] is the graph $G_\bot = (\bot, E_\bot)$ with $E_\bot = \{(u,v) : \exists x \in \top, (u,x) \in E, (x,v) \in E\}$. There is a link between two nodes of $\bot$ in the $\bot$-projection if these two nodes have at least one neighbor in common in $\top$. See Figure 9.3. Notice that each node in $\top$ induces a clique among its neighbors in the $\bot$-projection.

4. The $\top$-projection can be defined symmetrically.

In the bipartite projection, there is a link between two nodes u and v if they have a neighbor in common. Notice though that even if u and v have more than one neighbor in common, there is still only one link in the projection between u and v; it is a loss of information induced by the projection.

In order to avoid this loss, one could resort to weighted links (*i.e.* link (u,v) has weight w if u and v have w neighbors in common in the bipartite graph); other relevant ways of setting weights can be devised, yet one then transforms the problem of analyzing the bipartite structure into analyzing a weighted one, which remains difficult in spite of recent works.

Instead, the *redundancy coefficient* $rc(v)$ quantifies the extent to which a node v is the only node responsible for connecting its neighbors in the projection. In other words, $rc(v)$ measures the extent at which removing v from the network would affect the projection. It is defined as the probability, when one takes two neighbors u and w of a node v, that u and w are both linked to a node $v' \neq v$. One can also think of it as the density of the neighborhood of v in the projected graph induced by the bipartite graph deprived of v. See example in Figure 9.4.

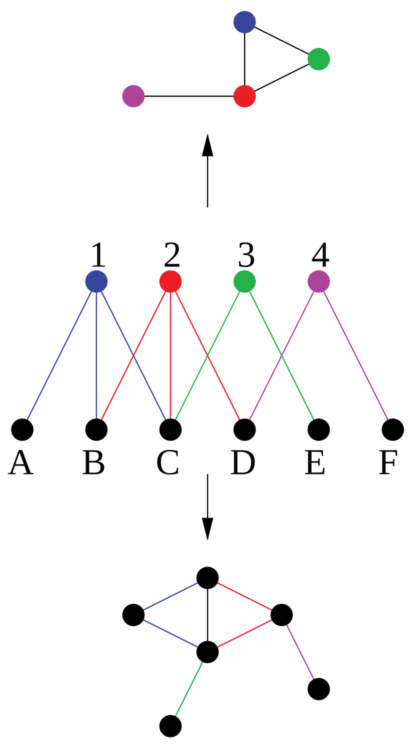


Figure 9.3: $\top$ and $\bot$ projections of a bipartite graph. Reproduced from [Latapy et al., 2008a].

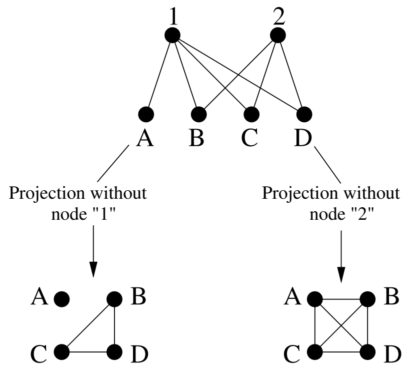


Figure 9.4: Redundancy of a node in the bipartite graph. Node 1 is connected to A, B, C and D, hence removing node 2 does not affect the projection. Node 2 has a redundancy of 1. However, removing node 1 disconnects A from the other nodes; 1 has a redundancy of $\frac{3}{4}$.

The goal of this section is to extend these notions to link streams.

A bipartite link stream is a tuple $L = (T, \top, \bot, E)$, with $T = [\alpha, \omega]$ a time interval, $\top$ and $\bot$ sets of nodes such that $\top \cap \bot = \emptyset$, and $E \subseteq T \times T \times \top \times \bot$. In other words, for all $(b, e, u, v) \in E$, either $u \in \top$ and $v \in \bot$, or the converse.

9.2.1 *Density and clustering coefficient*

The density is the probability, when one takes two random nodes $u \in \top$ and $v \in \bot$ and a random time $t \in T$, that there exists a link between u and v at time t in E:

$$\delta(L) = \frac{\sum_{(b,e,u,v)\in E} e - b}{|\top| \cdot |\bot| \cdot (\omega - \alpha)} \tag{9.1}$$

In a bipartite link stream, we define the clustering coefficient of a node $v \in \top$ as the probability, when one takes three nodes $x \in \bot, y \in \top, z \in \bot$ and a time $t \in T$ at random such that there are links $(t, v, x), (t, x, y), (t, y, z)$, that there is also a link between z and v at time t. This is the density of the cluster of nodes defined by the neighbors of v and the neighbors of these neighbors. Formally, it is the density of the cluster of nodes defined as follows: $\mathcal{C}(v) = \{(u,t) : \exists(b,e,u,v) \in E, t \in [b,e]\} \cup \{(u,t) : \{(b,e,v,x),(b',e',u,x)\} \subseteq E, t \in [b',e'] \cap [b,e]\}$, *i.e.* the cluster of nodes containing all neighbors of v and all neighbors of the neighbors of v. See Figure 9.5 for an illustration.

$$cc(v) = \delta(\mathcal{C}(v)) \tag{9.2}$$

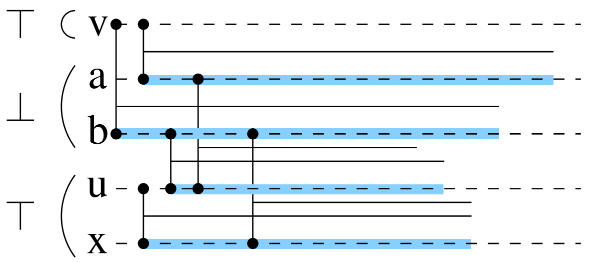


Figure 9.5: Clustering coefficient in link streams. Consider node v; the bipartite node cluster induced by its neighborhood (a and b) and the neighbors of those (u and x) is drawn in blue. The clustering coefficient of v is the density of this cluster.

9.2.2 *Projection*

Given a bipartite link stream $L = (T, \top, \bot, E)$, we define the $\bot$-projection of L as $L_\bot = (T, \bot, E_\bot)$, with $E_\bot = \{(b,e,u,v) : \exists x \in \top, \exists(b',e',u,x),(b'',e'',x,v) \in E, [b',e'] \cap [b'',e''] = [b,e], b > e\}$. In other words, there is link between u and v in $E_\bot$ if and only if these two nodes have a neighbor in common over the same period of time in $\top$. For an example, see Figure 9.6.

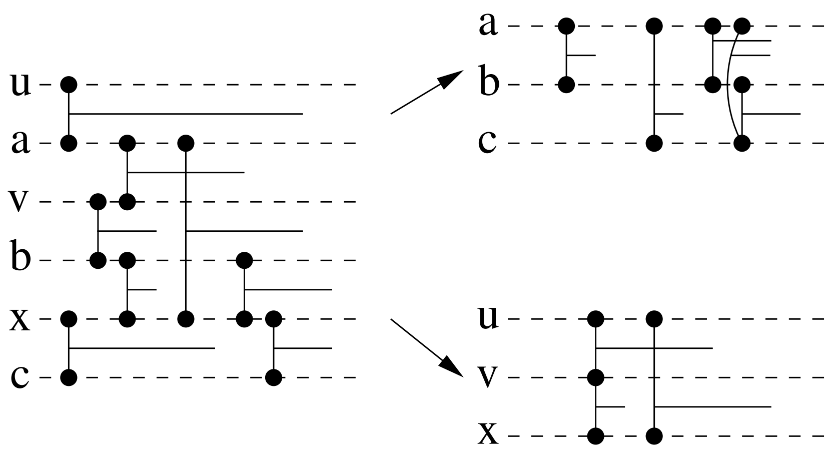


Figure 9.6: Example of a bipartite stream where $\top = \{a, b, c\}$ and $\perp = \{u, v, x\}$ (left). On the right, the $\top$-projection of L (top) and its $\perp$-projection (bottom). There is a link between u and x in the $\perp$-projection since the two nodes both interact with node a.

9.2.3 *Redundancy*

In link streams, the intuition is the same as for graphs; one wants to quantify the impact of the removal of a node v at a time t.

The redundancy of a node v is the probability that, for a random time t and two neighbors u and w of v at time t, that u and w have a neighbor $v' \neq v$ at time t. The redundancy of a node v at time t, $rc_t(v)$, is nothing but the redundancy of node v in the graph G_t. Notice that, for the same reasons as the ones exposed in Section 3.2 of Chapter 3, this is not equivalent to the average value of $rc_t(v)$ over t.

Just like for graphs, the notion of redundancy of a node is easier to express in terms of density in the bipartite projection. Given a bipartite stream $L = (T, \top\perp, E)$ and a node $v \in \top$ [5], let us consider the sub stream $L(\overline{v})$ of L, defined as $L(\overline{v}) = (T, \top \setminus \{v\}, \perp, \{(b, e, x, y) \in E : x \neq v, y \neq v\})$. In others words, it is the sub stream induced by L deprived of node v. Now consider the $\perp$-projection of $L(\overline{v})$, $L_\perp(\overline{v})$. The redundancy of v is the density of the sub stream induced by $N(v)$ in $L_\perp(\overline{v})$. See Figure 9.7 for an example.

5. The definition is symmetrical for a node $v \in \perp$.

Finally, we define the redundancy of the stream as the

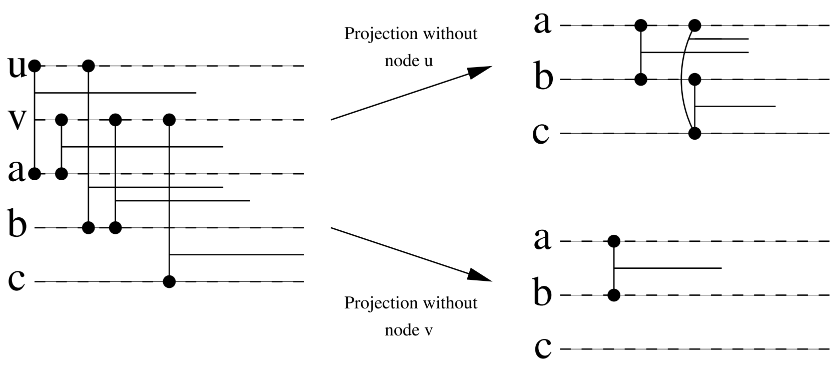


Figure 9.7: Consider a bipartite link stream L such that $\top = \{u, v\}$ and $\bot = \{a, b, c\}$. The redundancy of node u is close to maximal: indeed, removing u from L does not affect much the connectivity between nodes in the $\bot$-projection of L, since all the nodes in $\bot$ have v in common. However, v has a lower redundancy: indeed, v is the only common neighbor of a and c, as well as b and c.

average value for all nodes:

$$rc(L) = \frac{\sum_{v \in V} rc(v)}{|V|} \tag{9.3}$$

9.3 Conclusion

In this chapter, we have discussed the possible ways of modelling IP traffic as a link stream, as well as modelling choices we have made in this thesis. IP traffic typically comes as a sequence of (t, u, v), and we need to resort to a parameter Δ in order to use the link stream formalism defined in Part 1 of this manuscript. In the context of this exploratory work, we chose $\Delta = 2$ seconds.

The IP traffic in the MAWI dataset is intrinsically bipartite, and we adapted our work on link streams to bipartite link streams; we also have introduced new notions that have no counterpart in non-bipartite streams.

Notice that we decided to take the bipartite nature of single-router traffic captures into account, but not link direction, or weights on links, or protocol information, and so on. This is because in the context of link streams, undirected (or unweighted) links retain meaning on the original data, whilst the bipartite nature of traffic has so strong an impact on the underlying structure of the stream that non-bipartite notions would become irrelevant. For example, not taking this bipartite nature into account means that there are no dense groups, since not all links between two nodes can exist, or that the clustering coefficient is always 0.

Finally, discerning in metrics what is of interest in the data versus what is an artifact of the bipartite structure makes focusing on the features of the traffic more challenging.

Chapter 10
Analysis of IP traffic

Events happen at very different temporal and structural scales: denial-of-service attacks or network scans typically involve large number of machines on short periods of time, whereas Advanced Persistent Threats (APTs) are more subtle and last through a significant period of time. Moreover, it is expectable that large-scale events will have an impact on all kinds of metrics, whereas more subtle events will call for more precise statistics to be revealed. Large-scale events are typically many orders of magnitude over the expected value for a statistic, and they make smaller-scale events, such as data exfiltrations, or subtle APTs, imperceptible.

The goal of this chapter is to present a methodology to apply the link stream formalism to the study of IP traffic, and to apply it to real traffic from the MAWI dataset, with the aim of shedding light on events with simple statistics.

10.1 Our approach

In this section, we present the methodology we designed to study IP traffic, modelled as a link stream. Our primary goal is to assess the relevance of the link stream formalism for modelling IP traffic.

Our goal is to detect and identify events in IP traffic. We first define different types of events, then proceed to present a method for identifying events in IP traffic.

10.1.1 *Event characterization*

We define three nested types of events, from the more general to the more specific:

- A **detected event** is simply an outlier in a distribution, or an alarm from an anomaly detector. For example, a period of time where the number of packets is significantly higher than the usual value is a *detected event*;
- An **identified event** is a detected event for which we are able to identify (at least) one cause in the data. For example, if the detected event is caused by only two machines exchanging a lot of packets, it is an *identified event*;
- An **explained event** is one for which we found an explanation, eventually outside of the data. For example, two users transferring a large file explains the rise in the number of packets, and is an *explained event*.

This taxonomy of events is of course not the only possible one; signature-based event matching [1], where one tries to detect all events matching a pattern, is a natural counterpart to our approach.

1. Which requires expert knowledge and frequent updating to avoid *concept drift* [Gama et al., 2014].

Instead of targeting specific events with expert knowledge, we target here any kind of event, where *event* is then simply a label for statistical significance. In the remainder of this work, an *event* is a statistically abnormal value in a set of data points.

10.1.2 *Event identification through manual inspection*

In this exploratory work, we rely on manual inspection to find abnormal values. More precisely, following the work of [Latapy et al., 2013], we will consider three different situations:

1. The observed values are **homogeneous**, as in Figure 10.1, meaning they are all similar to the average value, and that no significant deviation from this value

is ever seen. In this situation, the considered property is not relevant for event detection. Bell-shaped distributions, visible in lin-lin scale, and exponential decreases (visible as straight lines in lin-log scale), are indicators of homogeneity.

2. The observed values are **heterogeneous**, as in Figure 10.2, meaning that the notions of *normality* or *event* are irrelevant. In this situation, the considered property is not relevant for event detection. Heterogeneity is indicated by polynomial decreases, visible as straight line in log-log scale.

3. The observed values are **homogeneous with outliers**, as in Figure 10.3, meaning that most values have a homogeneous nature but a few significantly deviate from them. In this case, the property may be used for event detection: statistically significant values are *outliers* indicating events, while most values correspond to a normal behaviour.

In the following, we use the term *outlier* as a synonym for *statistical outlier*, *i.e.* a data point that is abnormal in an otherwise homogeneous distribution. We use instead the less specific term of *event* for a data point that is clearly separated from the rest of the distribution, but that is not *per se* a statistical outlier [2]. Notice however that it may be possible to find events in heterogeneous distributions of values [3], even if they do not have as much significance as statistical outliers in an otherwise homogeneous distribution. Deciding which points of a distribution are outliers is an active area of research, and we discuss in conclusion some methods for automatically finding outliers.

2. Thus, outliers are events, but not the opposite.

3. For example, values that are separated by one or many orders of magnitude from the rest of the distribution.

10.1.3 *Hierarchy of features for the analysis of IP traffic*

One challenge of IP traffic is its volume, typically dozens of thousands of packets per second of traffic. This volume is a barrier blocking us from focusing on statistics with high computational complexity.

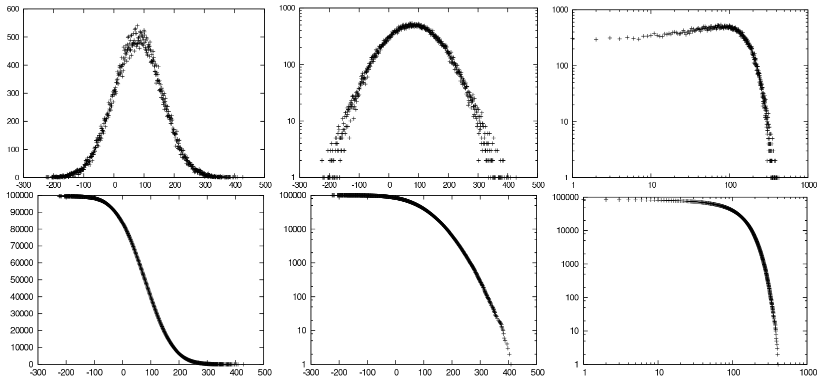


Figure 10.1: Typical homogeneous distribution. First row (left to right): the distribution in lin-lin, lin-log and log-log scales. Second row: the inverse cumulative of the distribution in the same scales. In such situations, all values are similar to the average value, and the considered property is not relevant for event detection.

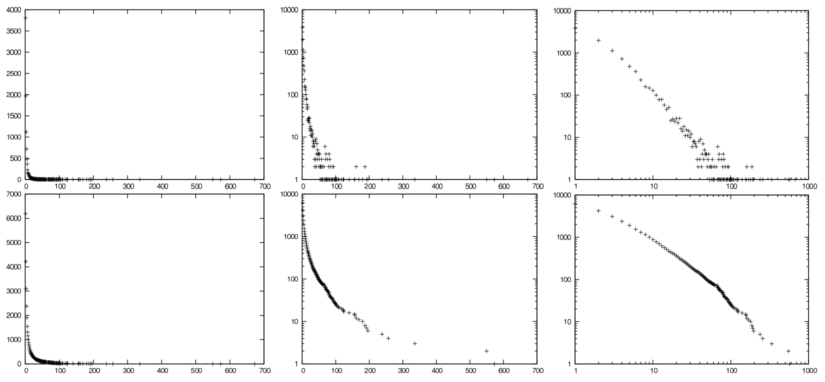


Figure 10.2: Typical heterogeneous distribution. First row (left to right): the distribution in lin-lin, lin-log and log-log scales. Second row: the inverse cumulative of the distribution in the same scales. In such situations, the considered property is not relevant for event detection.

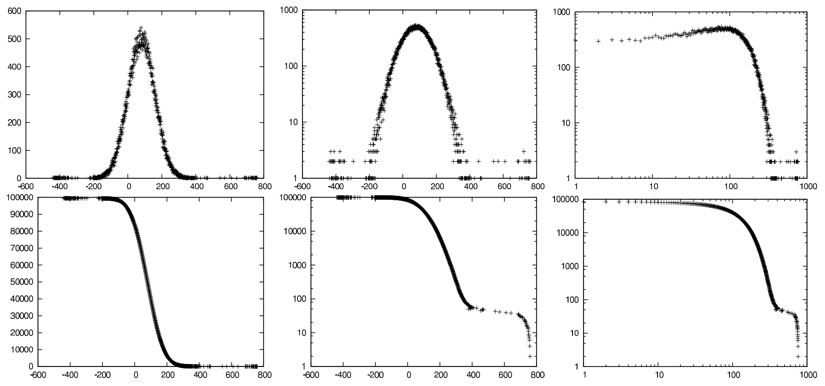


Figure 10.3: Typical homogeneous distribution with outliers. First row (left to right): the distribution in lin-lin, lin-log and log-log scales. Second row: the inverse cumulative of the distribution in the same scales. In such situations, the property may be used for event detection: statistically significant values are *outliers* indicating events, while most values correspond to a normal behaviour.

However, events do not involve all nodes all the time: one can use easily computable features to identify groups of nodes and periods of time that are of interest, and then study the substreams induced by these groups of nodes and periods of time. Since these substreams are smaller than the stream, one may then study features that are more computationally complex but that more descriptive power.

We propose a hierarchy of features, from the simplest ones to the most complex. The core idea is that the more computationally expensive a statistic is, the less data it is possible to handle with it. We use features with low computational complexity to detect and remove large scale events as well as identify relevant groups of nodes and periods of time in the data, in order to then be able to compute more costly features. The hierarchy we use is the following:

0 **Preliminary features**: number of packets per second, number of IP addresses per second;

1. **Basic features on the stream**: number of links per second, number of nodes per second, degrees;

2. **Bipartite features and dense groups**: clustering coefficient, redundancy coefficient, bipartite projection;

3. **Complex features**: cliques.

The preliminary features do not require transforming the data into a link stream, whilst the others do. For a sequence of m packets, creating the link stream can be done in $\mathcal{O}(m)$ time, but not in a single pass on the data. Notice that this hierarchy is not a computational complexity hierarchy; while it is true that computational complexity globally increases as one goes higher in the hierarchy, there is no equivalence between this hierarchy and computational complexity classes.

10.1.4 *Analysis of IP traffic process*

We proceed as follows: we start with the original dataset $\mathcal{D}_0$, and we set a counter i to 0, corresponding to the classes of our hierarchy. We initialize two sets: *detected* $= \emptyset$ will contain detected events, and *identified* $= \emptyset$ will contained identified events.

We compute the features of level i on the original dataset $\mathcal{D}_0$.

We detect events on the distributions of these features. All events detected are added to the set *detected*.

For all events in *detected*, we manually search for the cause of the event in the data. If we find such cause, we remove the event from *detected*, and place it in the set *identified*. If not, we leave the event in *detected*; our hope is that another feature will make it possible to identify it.

For all events in *identified*, we carefully remove the part of the data involved in the identified event, and we remove the event from *identified*. At the end of this step, the set *identified* is empty, but not the set *detected*.

When done with removing all identified events, we obtain a new dataset $\mathcal{D}_{i+1} \subseteq \mathcal{D}_i$. We repeat this process with features of the next class in our hierarchy. This process is summarized in Figure 10.4.

Notice that the set *detected* is never explicitly emptied. The rationale for this is that some events may be detectable with basic features, but those basic features might not be enough to identify them. We keep all detected events and expect that more complex features will bring identification.

Removing parts of the stream puts one at risk of removing events that are yet to be detected. No matter how precise the link stream statistics allow us to be, we will probably miss some events. However, the metrics we propose allow to remove a node at a single instant t if necessary, making it possible to remove parts of the data in a precise manner, thus this risk is reduced.

Moreover, keep in mind that our aim is *not* to explain *all* events in the traffic, but instead shed light on some of them happening at different scales. Even though we are at risk of removing a smaller-scale event happening at the same times and involving the same nodes as a large scale event, removing identified events allows us to notice

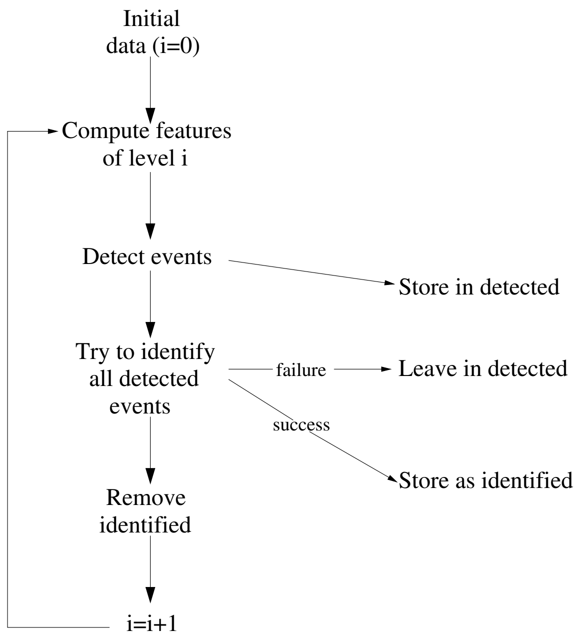


Figure 10.4: Our process for traffic analysis. We start with the original data and $i = 0$. At each step, we compute features of level i on the data. Then, we manually inspect the distributions of these features to detect events, and add them to a set of detected events. We then make attempts to identify all detected events. If an event can be identified, it is removed from the detected events, and is added as an identified event instead. We remove, as carefully as possible, the cause for all identified events. Finally, we set $i = i + 1$ and iterate the process with the data cleaned of its identified events.

smaller-scale events.

10.2 Results

We apply the methodology defined in section 10.1 to the data described in section 8.3 of Chapter 8. Our starting point is a collection of $\mathcal{D} = ((t, u, v))$, each element of the sequence meaning that IP addresses u and v have exchanged (at least) a packet at time t in the MAWI dataset.

10.2.1 Preliminary features

We first start by identifying events that do not require computing the link stream; we study the number of packets per second, and the number of distinct IP addresses per second.

Let us start with the number of packets per second. It is, for each second $s = 0..3599$, the number of (t, u, v) in $\mathcal{D}$ at s, *i.e.* $|\{(t, u, v) \in \mathcal{D}, s \leq t < s + 1\}|$. The number of packets oscillates around a mean value of $5 \cdot 10^4$ packets per second, with some sharp peaks (where there are nearly $140,000$ packets). Figure 10.5 shows the distributions of these values in all scales.

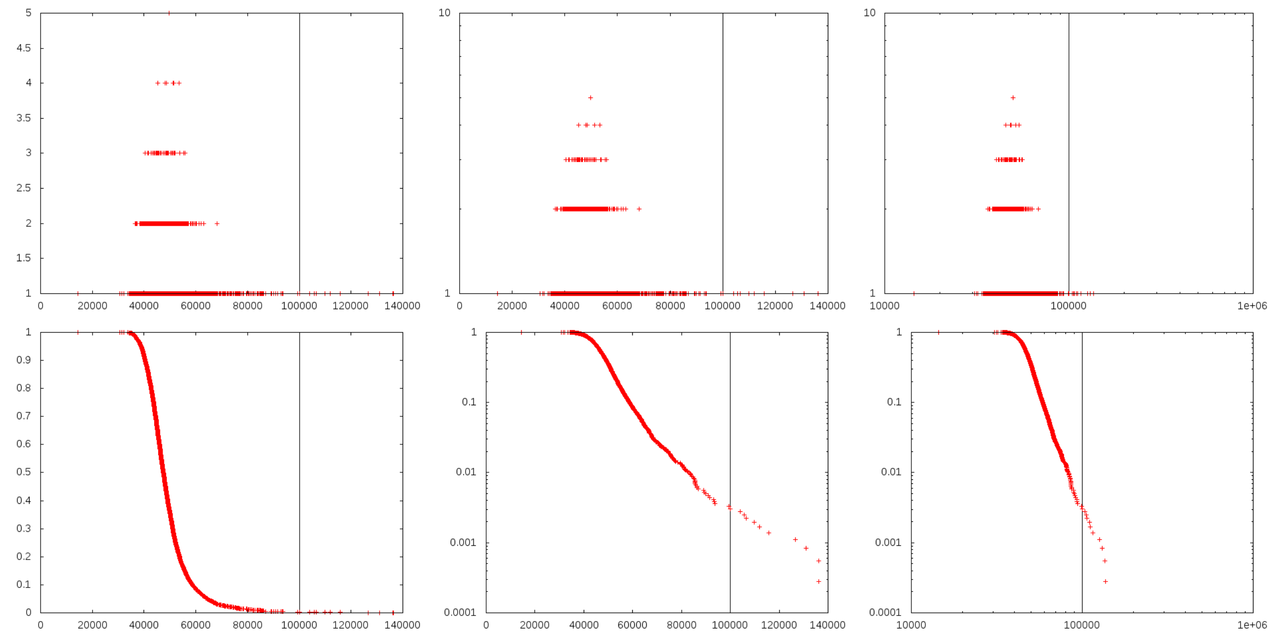


Figure 10.5: Top: Distributions of the values for the number of packets per second. Bottom: Inverse cumulative distributions.

Manual inspection of the distributions shows that we are in the case of a homogeneous distribution with outliers. We consider all points above $100,000$ as outliers, and we end up with 11 outliers; 9 of them are concentrated in the time interval $A = [133, 149]$ and 2 in the time interval $B = [1530, 1531]$: there are two *detected events*, that we denote A and B. We now attempt to identify these events.

In order to identify the event A, we consider the number of packets exchanged for each pair of IP addresses, *i.e.* for each (u, v), $|\{(t, u, v) \in \mathcal{D}, 133 \leq t \leq 149\}|$. Figure 10.6 (left) shows the inverse cumulative distribution of these values for all pairs active over $[133, 149]$. Clearly, one pair, that we denote by (IP_{A_1}, IP_{A_2}), exchanges a lot more packets than all the other pairs over this interval [4]. We identify this pair as responsible for the detected event. Notice that this event alone accounts for 0.5% of the total number of packets in the data.

4. This one pair exchanges $502, 320$ packets, and the second highest ranking pair exchanges $15, 420$ packets.

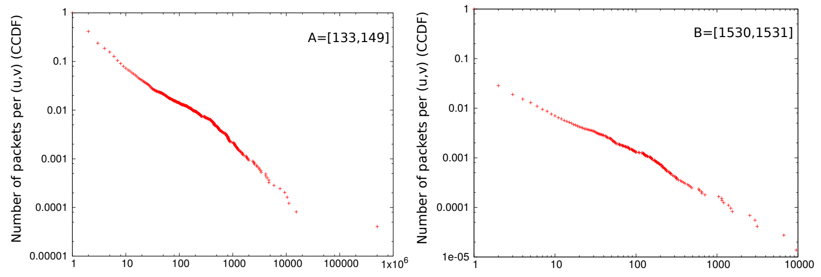


Figure 10.6: Inverse cumulative distributions of the values for the number of packets exchanged per pair of IP addresses for the 2 detected events $A = [133, 149]$ (left) and $B = [1530, 1531]$ (right). In the case of event A, one point stands out from the distribution, which helps us identify event A. However, in the case of event B, no such point exists, and we keep B as detected.

We repeat the process with the remaining outliers, concentrated in the interval $B = [1530, 1531]$. However, as Figure 10.6 (right) shows, no pair of IP addresses stands out, and the increases in the number of packets is due to another cause. We keep this event as detected, but not as identified, and wait for another feature to identify it.

We proceed by computing the number of distinct IP addresses per second, *i.e.* for all $s = 0..3599$, $|\{u : \exists(t, u, v) \in \mathcal{D}, s \leq t < s + 1\}|$. The number of IP addresses per second remains stable around $3, 000$ IP addresses at each second, with some sharp peaks standing out. There is also a pattern of interest between seconds 1500 and 2000, where the mean value changes for a short period of time.

The distributions in all scales for the values of IP addresses per second are displayed in Figure 10.7. Just like for the number of packets per second, we are faced with a homogeneous distribution with outliers. In order to not risk overestimating the number of outliers, we set the threshold to identify outliers to $20,000$, leaving 6 outliers.

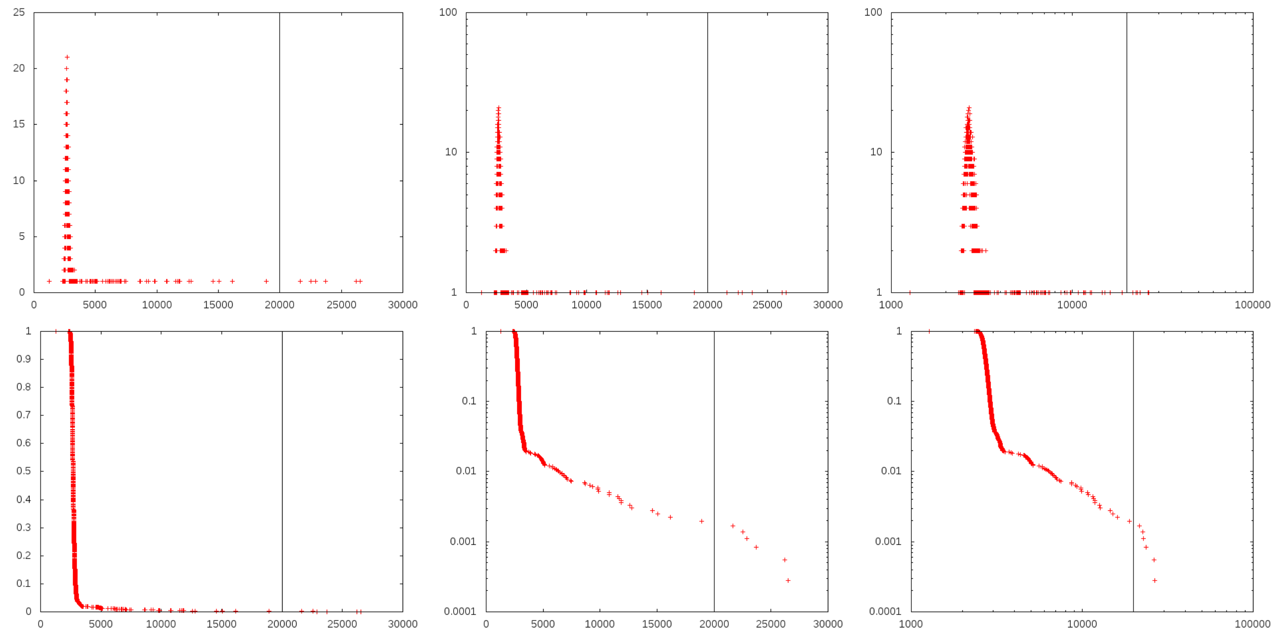


Figure 10.7: Distributions and inverse cumulative distributions for the values of the number of IP addresses per second. The dark vertical line at $x = 20,000$ is the threshold set for determining outliers.

The 6 outliers are concentrated at 5 different periods of time: 2 of them over interval $C = [1529, 1531]$, one at second $D = [1341, 1341]$, one at second $E = [1610, 1610]$, one at second $F = [1908, 1908]$ and one at second $G = [2737, 2737]$. Since the considered feature is the number of IP addresses per second, we turn to the number of packets exchanged for each IP address in order to seek identification for these detected events, *i.e.* for all u, $|\{(t, u, v) :$

$\exists(t, u, v) \in \mathcal{D}\}|$, and show the distributions of these values for each detected event in Figure 10.8.

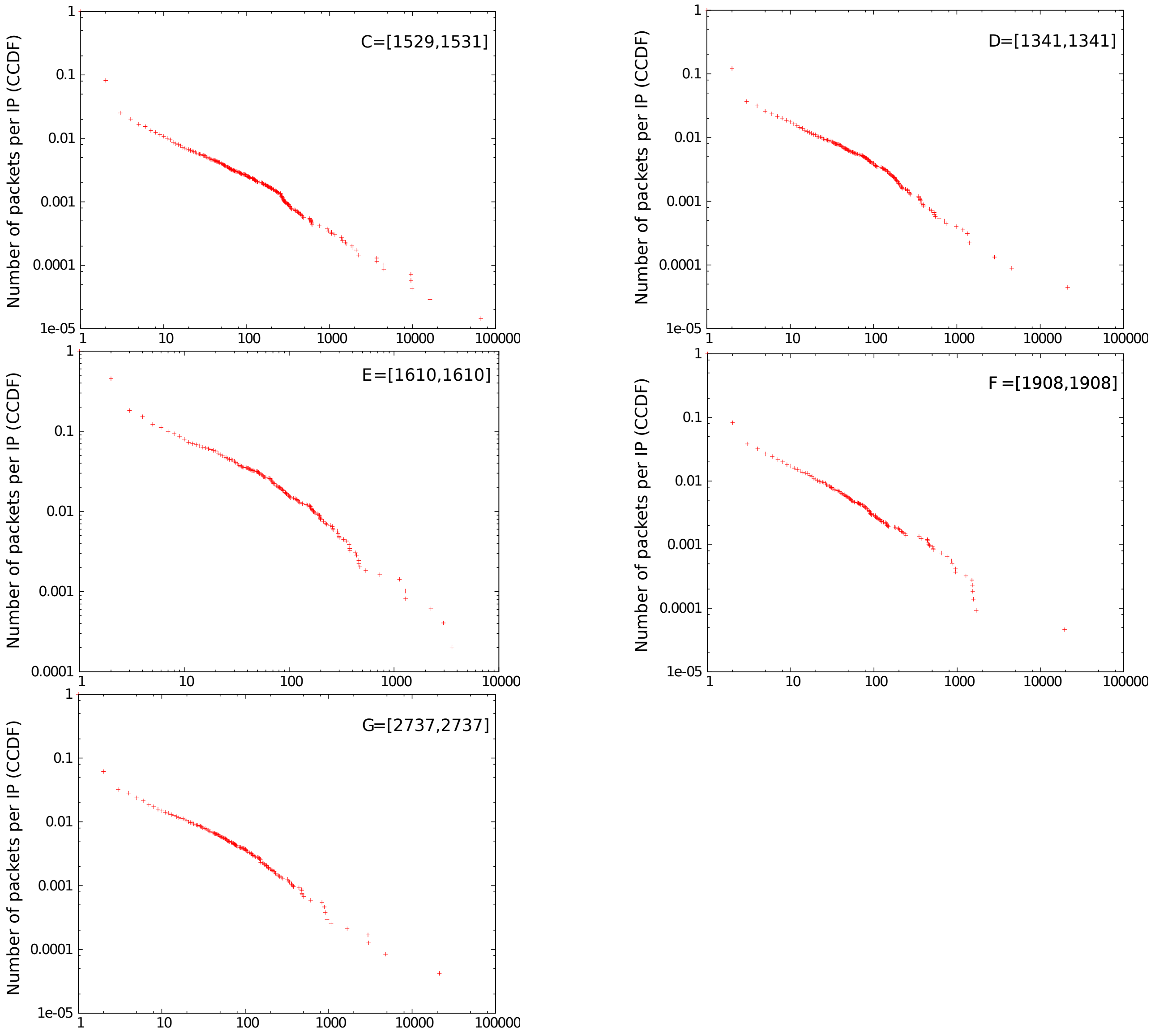


Figure 10.8: Inverse cumulative distributions of the values for the number of packets exchanged per IP address for the 5 detected events C (top left),D (top right),E (middle left),F (middle right) and G (bottom). In the case of events C, D, F and G, one point is clearly standing out from the rest of the distribution, and so we use this point to identify the event. In the case of event E, however, no value stands out, and so we keep E as a detected event.

The distributions for event C in Figure 10.8 is heterogeneous but shows a clear value standing out from the others; one IP address, that we denote IP_C, exchanges many packets with a substantial number of IP addresses over interval $[1529, 1531]$; we label C as an identified event. It is likely that the detected event B, happening at the same period of time and involving the same IP addresses, refers to the same event. Hence, we consider that event B has been identified too.

Similarly, events D, F and G can be identified, since in all cases, there is a single value an order of magnitude superior to all values in the distributions shown in Figure 10.8 for these events.

However, this is not the case for event E. Instead, as Figure 10.8 shows, the values are distributed in a heterogeneous way with no point standing out. We leave E as a detected event.

At the end of this step, we have detected 7 events – A, B, C, D, E, F, G –, and have been able to identify 6 of them – A, B, C, D, F, G. We now explain in details how we remove identified events from the data. For event A, the identified cause was a large volume of packets between a pair of IP addresses, IP_{A_1} and IP_{A_2}. We remove from the data all (t, u, v) such that $t \in [133, 149]$, $u = IP_{A_1}$ and $v = IP_{A_2}$.

We plot the number of packets per second before (top) and after (bottom) removal of the identified event A in Figure 10.9.

We show in Figure 10.10 the number of IP addresses per second over time before (top) and after (bottom) removal of the identified events C, D, F, G. Notice that the peak at second 2737, associated to event G, did not disappear but was instead reduced; however this feature will not tell us more about the cause of this peak.

Notice that we did not remove event E, which was detected but not identified.

After having removed these 6 events, that account for 1% of the traffic, we obtain a new dataset $\mathcal{D}_1$, and proceed with more complex features.

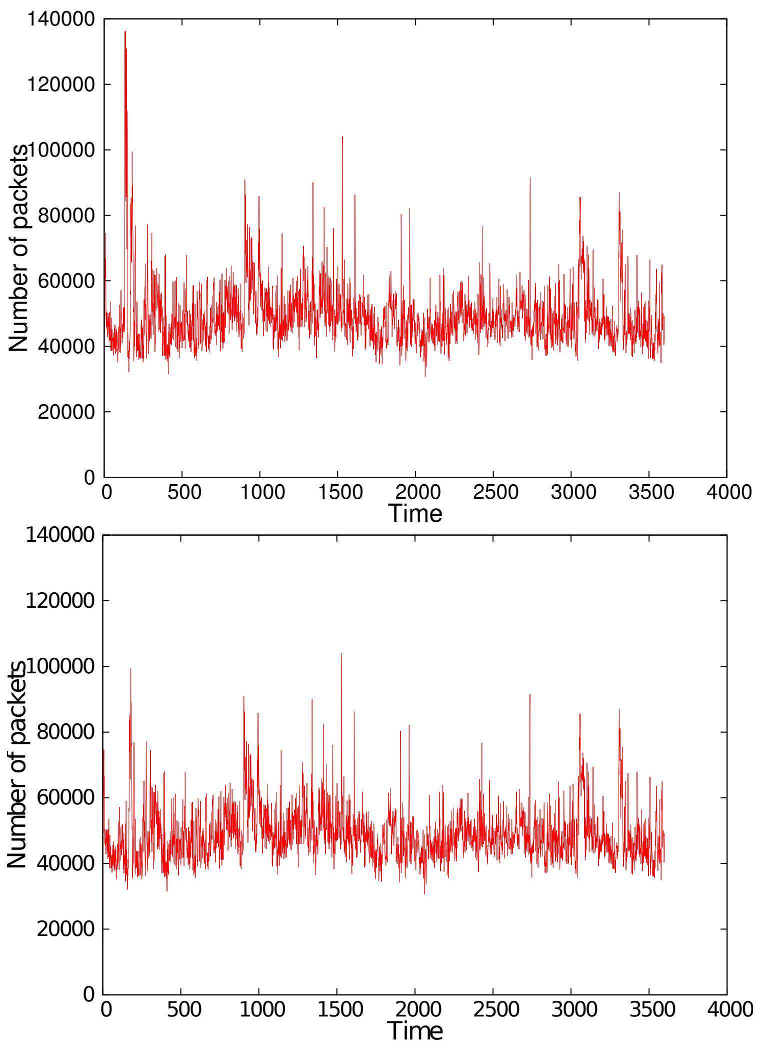
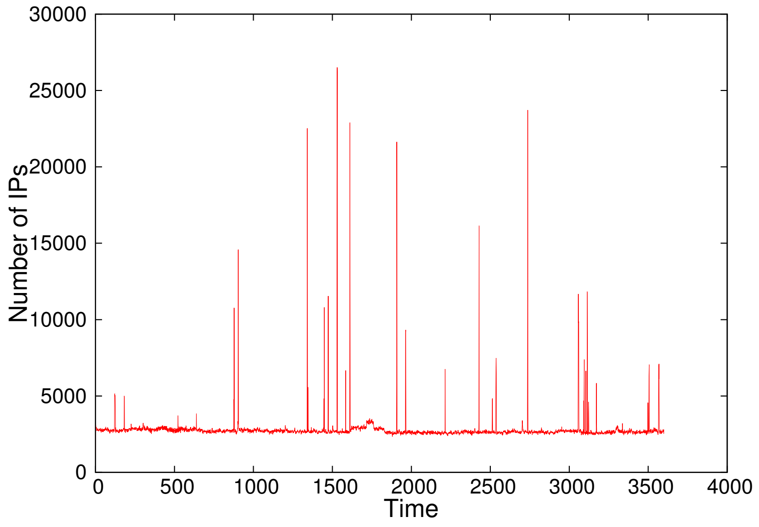


Figure 10.9: **Top**: Number of packets per second as a function of time. **Bottom**: Number of packets per second as a function of time without packets involved in event $A = [133, 149]$. The peak has decreased under $100,000$ packets per second.



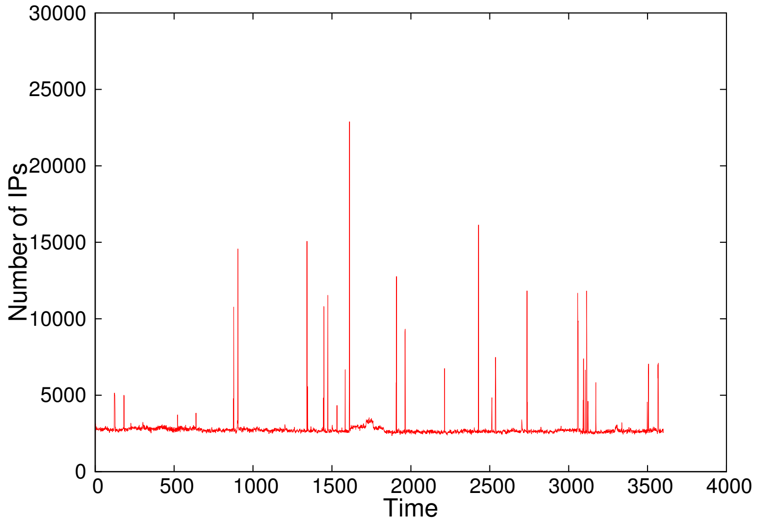


Figure 10.10: **Top:** Number of IP addresses per second as a function of time. **Bottom:** Number of IP addresses per second as a function of time after removal of the 4 explained events C, D, F and G. Event C (over interval $[1529, 1531]$, corresponds to an event that had previously been detected but not identified. Event E (at second 1610), that has not been identified, is not removed.

10.2.2 *Basic features on the link stream*

Given the dataset $\mathcal{D}_1$ obtained in the last section, we transform it into a link stream using the procedure described in section 9.1 of Chapter 9. As stated before, we chose $\Delta = 2$ seconds.

We denote by $L_1 = (T_1, \top_1, \bot_1, E_1)$ the obtained link stream, with $T_1 = [0, 3600]$. By convention, $\top_1$ is the set of IP addresses in WIDE, and $\bot_1$ is the set of IP addresses not in WIDE.

The number of links per second and nodes per second in the link stream are very similar to the number of packets per second and the number of IP addresses per second, and their analysis leads to identifying the same events. For these reasons, we do not display them here.

In the number of IP addresses per second displayed in Figure 10.10 (bottom), a motif of interest is visible over $H = [1500, 2000]$.

Our intuition is that this short regime change will be noticeable when looking at the degrees in the link stream [5]. In order to compute the degrees, we transform the stream into a sequence ± 1 indicating the appearance or disappearance of a link: in other words, each link (b, e, u, v) is transformed into $(b, 1, u, v), (e, -1, u, v)$, a 1 in the second field indicating that the link started, and a -1 indicating that the link stopped. By sorting the elements of this sequence by u, and by increasing time for each u, it is possible to obtain the degree profile [6] of each node in the stream in a single pass of this file without storing anything else in memory than the degree of a node at time t, $d_t(u)$. Notice that the degree of u, $d(u)$, is nothing but the average value of its degree profile $p(u)$.

5. Remember that the degree of a node u in L is its number of neighbors, each neighbor weighted by the duration of its interactions with u.

6. The degree profile of a node u is the function that associates to any t the degree of u at time t, *i.e.* $p(u) : t \mapsto d_t(u)$

Let us first look at the degree distribution of L, presented in Figure 10.11. It it the probability, for all k, when one chooses a node u at random and a time t at random, that node u has degree k at time t. In complex networks, degree distributions have a strong descriptive importance, as they carry a lot of information about the network. It is the same in link streams. Notice some peaks around powers of 2, that are typical of IP traffic.

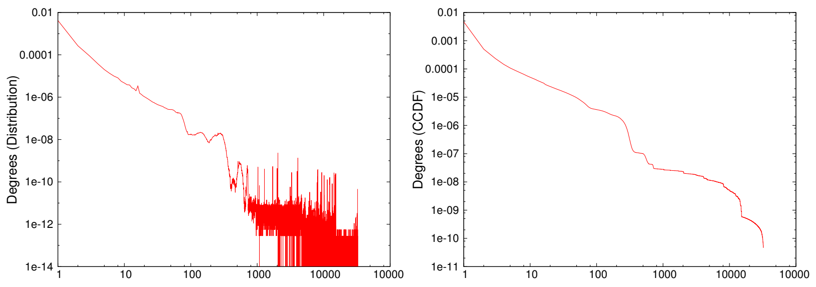


Figure 10.11: Degree distribution of the link stream in log-log scale (left), and inverse cumulative degree distribution in the link stream in log-log scale (right). The degree is the is the number of neighbors of a node u chosen at random at a time t chosen at random. Some small peaks can be observed in the distribution at powers of 2, a typical feature of IP traffic.

However, the degree distribution itself contains too much information to pinpoint the regime change we are looking for. We manually inspect the degree profiles of nodes that reach a high degree, *i.e.* for each node we keep its maximum degree in time [7], $d_{max}(u)$, and rank the nodes by decreasing order of maximal degree in time. We search, in the first nodes of this ranking, for nodes active over $H = [1500, 2000]$ that might explain the motif found through manual inspection.

7. *i.e.* for each node u, $d_{max}(u) = max(d_t(u))$.

We denote by IP_{H_1} and IP_{H_2} the nodes that rank respectively 21^{st} and 23^{rd} and which seem good fits, since they are active in the stream only during the interval $H = [1500, 2000]$, as their degree profiles show in Figure 10.12. Moreover, when they are active, these nodes have approximately 600 and 700 neighbors, respectively, which approximately corresponds to the amplitude of the change in the number of IP addresses per second, see Figure 10.13 (top).

We consider event H as identified, and we remove IP_{H_1} (Figure 10.13, middle), then IP_{H_2} (Figure 10.13, bottom) from the data over interval $[1500, 2000]$. The pattern of interest was solely caused by these two IP addresses: indeed, it has completely disappeared in Figure 10.13 (bottom).

10.2.3 *Comparison with MAWILab*

We have searched the *MAWILab* anomaly database for the events we have shed light onto, to validate our approach.

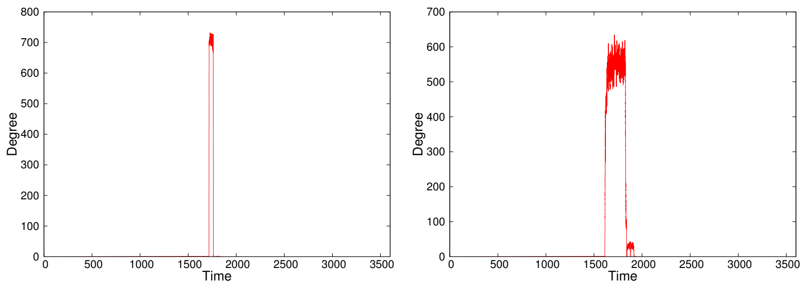


Figure 10.12: Degree profiles of the 21st (left) and of the 23rd (right) nodes of highest maximal degree (right). On both plots, a point y at time t means that the considered node has active links with y neighbors at time t. For example, the 21st node of highest maximal degree has degree 0 (*i.e.* does not interact with any other IP address) at all times except in $[1716, 1834]$, where it has between 700 and 800 active neighbors in the same second.

Surprisingly, the event A, identified through the number of packets per second over $[133, 149]$ is not present in *MAWILab*. One possible explanation is that this event is not a *anomaly*, and is constituted of legitimate traffic.

Concerning the events C, D, F, G that we have pinpointed in the number of IP addresses per second, they are all present in *MAWILab*, and are associated with time intervals and port numbers.

The event H, identified using the degree profiles of nodes of highest maximum degree, was caused by two IP addresses IP_{H_1} and IP_{H_2}, that are both present in *MAWILab*, and associated with times $[900, 1800]$ and $[1576, 1622]$, respectively.

An interesting point is that *MAWILab* reports some of these events as lasting over 900 seconds [8], while we are able to narrow these events to more precise intervals. For instance, the IP IP_{H_2} is flagged as anomalous from time 1576 to time 1622 in *MAWILab*, but the degree profile of this node, in Figure 10.12, left, shows that this IP address has no interactions before second 1612.

8. Which does not mean that the anomaly lasted 15 minutes, but rather that the detectors were not able to identify a more precise interval.

10.3 Conclusion

In this chapter, we have devised a method for studying IP traffic as a link stream, and have applied it to a trace of traffic from the MAWI dataset.

A core idea of our approach is that with enough information, events can be removed from the data and allow to detect smaller events. The method we devised works by identifying statistical outliers, and labelling them as

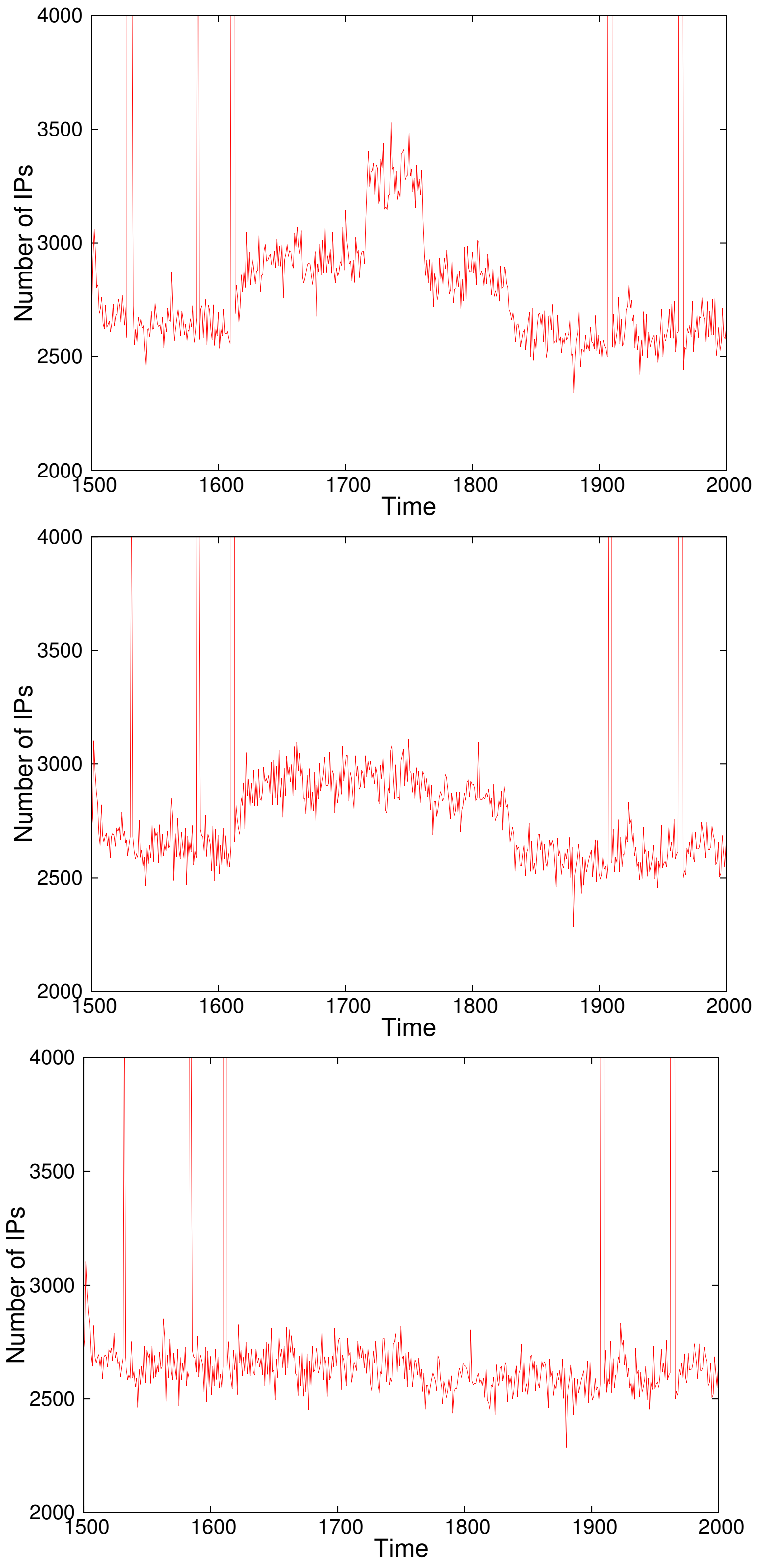


Figure 10.13:
Top: Inset of Figure 10.10 (bottom) between times 1500 and 2000.
Middle: Number of IP addresses per second after removal of the 21st node of highest maximal degree.
Bottom: Number of IP addresses per second after removal of the 21st and 23rd nodes of highest maximal degrees.

detected events; manual inspection allows to identify the cause of certain events, making it possible to remove them from the data in a careful and precise manner.

We have illustrated our approach by shedding light on events in 1 hour of traffic taken from the MAWI dataset. We model the traffic as a link stream; since the traffic is a collection of (instantaneous) packets, we chose a value of $\Delta = 2$ seconds to analyze the traffic. Overall, we detect 7 events and identify 6 of them, that we are able to remove from the traffic. We summarize our results in Figure 10.14.

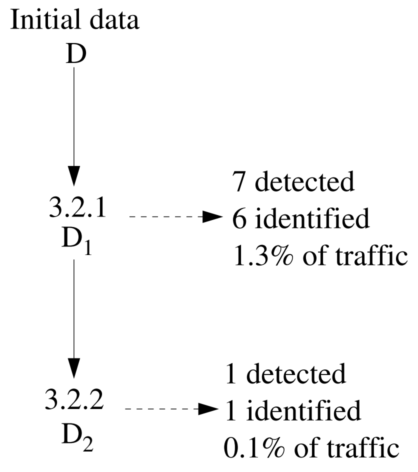


Figure 10.14: Summary of the events found in the different sections. In section 10.2.1, we detected 7 events (2 with the number of packets per second, and 5 with the number of IP addresses per second), and identified 6; in section 10.2.2, we detected 1 event and identified it. Altogether, these events represent 1.4% of the traffic.

This first work is exploratory, yet it shows the relevance of analyzing IP traffic with link streams, which was its primary goal. We have only studied a few of the features defined in Section 10.1.3; our goal was to validate our approach, yet the other features described in Section 10.1.3 are worthy of interest. We discuss now ways to apply these more subtle features.

The degree profiles, as the ones showed in Figure 10.12, carry a lot of information about the behaviour of individual nodes. For example, most profiles show nodes that are active over short periods of time, and have very few neighbors; others have few neighbors consistently through time; overall, the degree profiles are diverse.

Classifying these degree profiles can help understand the role played by nodes in the traffic. We identify two ways of doing so:

1. One could use expert knowledge and design a taxonomy with which profiles can be matched. A straightforward taxonomy is to separate nodes that have many neighbors versus nodes that have few neighbors, or nodes that are active over long periods of time versus nodes that are active over short periods of time [9].

2. One could use statistical models, and fit the distributions of values of the degree profiles to Gamma distributions. Gamma distributions are a two-parameter (α, β) family of distributions, including among others exponential distributions and normal distributions. Parameters can for instance be determined using maximum likelihood estimation. Classifying the estimated parameters may give insight on the laws governing the nodes' behaviour.

9. The thresholds to determine what "long", "short", "few" and "many" mean in the context of each dataset can be found be inspecting the distribution of these values.

In order to explain the behaviour of groups of nodes, bipartite statistics are of particular interest. For instance, traffic from different users to a same service might be distributed to a group of servers of the same company. These servers are expected to have a lot of users, and some them in common: as a consequence, their degree in the stream will be high, and their degree in the bipartite projection will be high too. On the contrary, users typically interact with a few services, that most users interact with: consequently, users will likely have a low degree in the stream, but a high degree in the bipartite projection. Looking at the correlations between the degree in the stream and the degree in the bipartite projection is a promising direction of research.

Finally, finding bipartite cliques of interest in the traffic is an important perspective. Given the volume of traffic, enumerating all maximal cliques is currently out of reach, yet heuristics can help finding cliques of interest. For example, one could pick a random link (b, e, u, v) of the stream and its corresponding starting clique, $c =$

$(\{u, v\}, [b, e])$. One can then, for instance, choose randomly one of the cliques produced by c. This corresponds to choosing a path at random in the configuration space, and has the advantage of storing nothing more than the link stream, the current clique and its children in the configuration space. The rationale behind this idea is that substreams induced by large maximal cliques will contain more links, and that more paths in the configuration space lead to a large maximal clique. However, in the context of IP traffic, large maximal cliques in terms of number of nodes are maybe not the most interesting: indeed, the largest cliques will likely contain hundreds of nodes, but they are distributed as hundreds of nodes interacting with a single node; this is a typical signature of widely frequented servers [10]. Instead, one could choose preferentially paths in the configuration space that lead to balanced cliques, *i.e.* cliques that have a similar number of nodes in $\top$ and in $\bot$. Since all nodes are added in all possible ways in our algorithm, it should be possible to keep cliques balanced at each step in the configuration space.

10. The maximal cliques we obtained on the MAWI dataset found out Google and Twitter as cliques of respectively 600 and 300 nodes, approximately.

Chapter 11
Conclusions and perspectives

In this second part of the thesis, we have applied the link stream framework to the analysis of IP traffic and have assessed the relevance of this approach. In this proof-of-concept work, we have set our focus on showing the relevance of the link stream formalism for modelling IP traffic, and to identify events in the traffic.

We have shown that it is possible to handle link streams of millions of links under reasonable processing times. Indeed, all the computations presented here are performed in less than one hour.

We have adapted the formalism introduced in the first part of this thesis to model bipartite link streams, and extended existing notions that are specifically defined for bipartite graphs to the case of link streams.

With our definition of event being a statistical one, we devised a method that sheds light on events in the traffic, ideally leading to a stream devoid of events. We observe that different features explain different kinds of events, and that even simple link stream features have relevance in terms of interpretability and realism of computation.

We have demonstrated that removing parts of the data can be done carefully. Conscious that removing information from the data is a delicate process, we have given some pointers about how and when to remove elements of the data.

The work presented in this second part remains exploratory, and opens a variety of perspectives, be it in terms of automated event detection or in the use of more subtle link stream features. We now present some of these perspectives that seem the most relevant to us.

Since we are in the context of a proof-of-concept, the easiest way to identify outliers was with the bare eye. Yet, many methods in the literature aim at automatically detecting outliers in distributions or in time series. A substantial amount of work in statistics has been done on outlier detection, and books by [Leroy and Rousseeuw, 1987] and [Bamnett and Lewis, 1994] review this work. For more recent results, one can turn to the work of [Bakar et al., 2006].

For now, we have been working on traffic captures stored on-disk. However, some of our features can readily be computed in a streaming fashion, where one keeps a (possibly evolving) sample of the past, and sees the packets as they arrive on the network card. This is the case for the degrees, and the degree profiles, as well as for density and cliques. Bipartite features, however, such as the projection, can hardly be updated on the fly as new packets arrive. One possible approach is to use on-the-fly features to identify groups of nodes and periods of time of particular interest, store them on-disk and analyze them afterwards.

Studying the influence of parameter Δ is a perspective of our work. Instead of having one Δ as a parameter of the stream, one can imagine a different Δ for each node at each time. For example, for each node u, one could set Δ inversely proportional to the number of interactions involving u. Another possibility is to use port or protocol knowledge to choose Δ, for example using the default timeouts. The formalism we provide is already adapted to these transformations. Identifying precisely the scales at which events happen is particularly appealing, and one can do so by resorting to a set of $\{\Delta_i\}_{i=1..k}$ instead of a single parameter, and, for all i, observe the impact of Δ_i on the stream.

Using our features into machine learning classifiers is also a prospective work. In particular, assessing the correlations between our features and normalizing their outputs are two short term directions of research.

Another perspective is to identify and describe the links between our approach and signature based approaches, *e. g.* to characterize mice flows in the link stream, or heavy hitters. Expressing these particular flows in terms of link streams requires further work.

A more rigorous comparison to *MAWILab* is also a prospective work. Hitherto, we have simply checked that the events we found were also flagged as anomalies in *MAWILab*; we have also seen that while *MAWILab* sometimes flags an anomaly as lasting 900 seconds, while link stream metrics allow us to be more precise. However, a future step is to compute the false positives, false negatives, true positives and true negatives when comparing our results to *MAWILab*. On the long-term, our output being groups of nodes over intervals of time, it is not incompatible with integration in *MAWILab*.

Part III

Conclusions and perspectives

Chapter 12
Summary and Contributions

In this thesis, our aim was to propose a formalism to describe sequences of interactions, just like graph theory is a formalism to describe networks.

In our opinion, a satisfactory formalism for link streams should be simple, and generalize both signal theory and graph theory [1]. In this endeavour, special attention has been paid to retain the relations existing between notions in graph theory.

1. For instance, a link stream L where all links last from α to ω encodes no temporal information, and our notions become equivalent to the ones of graph theory: the density of L is exactly the density of the induced graph $G(L)$, and so on. Similarly, a link stream with only 2 nodes encodes no structure, and can readily be studied as a time series.

Chapter 2 is dedicated to the introduction of basic definitions: link streams, substreams, as well as graphs induced by streams and line streams. Consistency with the existing definitions in the state-of-the art, such as studying aggregated graphs, or sequences of graphs, is demonstrated.

Density is a fundamental notion of graph theory, and we focus our attention in Chapter 3 on the multiple notions derived from density: neighborhood, degree, clusters of nodes, cliques, and clustering coefficient and transitivity ratio. These notions are fundamental in graph theory, are interesting in themselves, and have a strong descriptive power. They make it possible to formally define dense groups over time, notions of similarity between neighborhoods, density between two clusters, and so on.

Then, in Chapter 4, we work on notions derived from paths: centralities, connected components, k-closure. These notions pave the way for assessing the importance of nodes over time, as well as studying reachability in link streams.

In many contexts, the data comes as a sequence of instantaneous contacts: this is the case of IP traffic, face-to-face contacts, or email exchanges. Moreover, one often wants to consider a time scale Δ coarser than the exact times of links in L. In Chapter 5, we discuss Δ-analysis of link streams, and show that notions defined on link streams in chapters 2 to 4 can readily be applied to such datasets.

Studying link streams enhanced our understanding of graphs. For instance, instead of considering that a node u in the stream is in cluster A from time b_A to e_A, then in cluster B from time b_B to e_B, one can consider that in a graph, node u belongs to both communities A and B at different levels of implication: node u might be involved at 66% in community A and 33% in community B, for instance. This work gave us a more thorough understanding of overlapping communities in graphs.

A research paper presenting the work of Chapters 2 to 4 is in preparation.

Conscious that our formalism raises algorithmic questions, we focus on computing cliques in link streams in Chapter 6. We propose an algorithm to enumerate the maximal Δ-cliques of a link stream. Instead of focusing on algorithmic efficiency [2], we concentrated on exploring data and assessing the relevance of Δ-cliques on contacts between individuals. We have found that Δ-cliques exhibit different behaviours from cliques in graphs: for instance, while nodes involved in a maximal clique in the induced graph are typically individuals belonging to a group (*i.e.* a class, or a team), maximal Δ-cliques in link streams indicate short meetings, exams, breaks, and so on. This work has led to publications in an international journal [Viard et al., 2016], as well as an international workshop [Viard et al., 2015b] and a national conference [Viard et al., 2015a].

2. Recent work done by [Himmel et al., 2016] adapts the classical Bron-Kerbosch algorithm to Δ-cliques, with significant improvements in performance.

We conclude on this part of the thesis in Chapter 7 and present some perspectives of our formal work.

It is of critical importance to us that our work brings both *fundamental* and *applied* progress. During our reflection, we have made constant round-trips between theory and application: applications gave us insights on what phenomena were out of our understanding, and we designed the theoretical tools that had meaning with respect to applications. We focused on one application, the analysis of IP traffic, in the second part of this thesis.

After having briefly reviewed the context of IP traffic analysis in Chapter 8, we present the MAWI dataset, a collection of daily captures of traffic at a capture point between the WIDE academic network and the upstream ISP. We chose this dataset mainly because it collects a large number of different behaviours, over large periods of time (up to 72 hours in a row, and 15 years of daily 15-minute captures); finally, a key advantage of this dataset is the *MAWILab* initiative, that runs various anomaly detectors on the traffic, and provides a public breakdown of the anomalies found in each capture. *MAWILab* offers an excellent baseline to compare our work and results.

We discuss ways to model IP traffic as a link stream in Chapter 9. The traffic can be studied either at the packet level or the flow level; both are relevant, but in our work we focus on the traffic at packet level.

The nature of the capture makes it intrinsically bipartite, and we present adaptations of the formal work defined in Part 1 to take this specificity into account. After having formally defined bipartite link streams, we extend the notions of density and clustering coefficient to such link streams. We also extend from the literature on bipartite graphs the notion of projection, an operation to obtain a non-bipartite stream from a bipartite stream, and redundancy.

Chapter 10 is dedicated to the analysis of IP traffic and its events. We design a method to analyze the traffic, and, under the assumption that events happen at different scales, to remove carefully parts of the stream corresponding to large-scale events, in order to detect smaller-scale events. The core idea is that simple events (such as a router crash) can easily be identified with simple statistics, such as the number of packets per second (which will drop to 0 in the case of a crash). Once they have been identified, one can remove them from the data, and more complex statistics will reveal more subtle events. Removing parts of the data is delicate, as we discuss in Chapter 10.

In this exploratory work, we rely on manual inspection of the distributions of values to identify events. We apply our methodology to a trace of 1 hour of IP traffic, and effectively identify some events. We characterize these events, and show that some of them are also present in the *MAWILab* database. These first results demonstrate the relevance of our framework, and open numerous perspectives, like introducing link stream features in machine learning algorithms.

We conclude on the second part of this thesis in Chapter 11 and present some perspectives for the analysis of IP traffic.

In collaboration with researchers of the Fukuda Lab, Tokyo, Japan, a research paper presenting the work of Chapters 9 to 10 is in preparation.

In this thesis, we define a consistent and comprehensive framework to describe streams of interactions; we adapt existing notions from the literature, such as paths, and define novel notions, such as the density. This framework allow us to analyze IP traffic directly as a stream of packets, and shed light on events in it.

Chapter 13
Perspectives

The work done in this thesis opens numerous perspectives. We present some of them in the following chapter.

13.1 Generalizing link streams

In link streams, we consider that all nodes are present from α to ω. Our work on node clusters gave us a glimpse of a more general definition of link streams (see Figure 13.1); a link stream $\mathcal{L} = (\mathcal{V}, \mathcal{T}, V, E)$, with $V \subseteq \mathcal{V} \times \mathcal{T}$ and $E \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{T}$. In other words, nodes are pairs (v, t), and a link (u, v, t) in E implies that (u, t) and (v, t) are in V. In the case where nodes are always present (*i.e.* for all $v \in \mathcal{V}$ and for all $t \in \mathcal{T}$ there exists $(v, t) \in V$), one obtains a link stream identical to what has been previously defined.

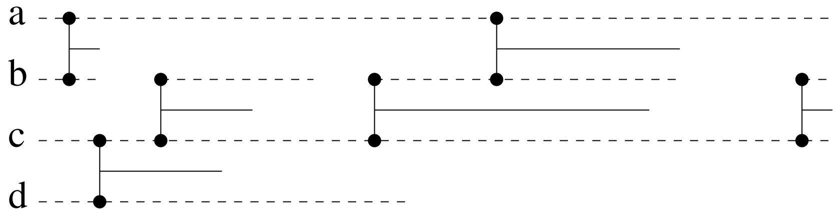


Figure 13.1: Another definition for link streams, in which nodes are not present for the whole stream duration. In this case a link (u, v, t) can only exist if u and v are in the stream at time t.

Though more general, this definition does not simplify the notions we introduced, as we have seen when studying node clusters (Chapter 3, section 3.2). Indeed, one has to count nodes proportionally to their presence, leading to more intricate formulae. Moreover, the relations that presently hold between graphs and link streams do not necessarily hold in such objects. However, developing the formalism for this object, as well as studying its properties and its relevance to real-world datasets is an important perspective.

One obvious perspective of our work is the development of the formal framework we introduced. We paved the way for quantifying the importance of nodes in link streams (in Chapter 4); many of the complex objects defined on graphs (communities, motifs, cores, and many others) have hitherto no defined counterpart in link streams, yet we offer all the theoretical tools for defining such objects. Notions from signal theory are yet to be adapted.

Moreover, researchers working on temporal networks have brought applied progress, and defined precise measures on interaction streams. These measures quantify diffusion [Masuda et al., 2013], burstiness [Lambiotte et al., 2013], and others. Defining these measures within the link stream framework may unify these notions, and bring a better understanding of the results given by these measures. It is also true that analyzing series of graphs can be defined within the link stream framework, and seeing how the great amount of work on dynamic graphs may be translated into the link stream framework is an interesting direction.

13.1.1 Computational considerations

Whilst many graph theoretical notions have a counterpart in link streams, classical algorithms are not trivially adaptable. We made a first step in this direction by providing an algorithm for clique computation in link streams, and exploring how it can be improved is an interesting perspective. Many other notions defined in Part 1, like paths, centralities, communities, etc. raise non-trivial algorithmic issues.

Some work is also required to create distributed algorithms, *i.e.* algorithms where each node is responsible for updating its neighborhood. Clique computation in link streams may be adapted to a distributed algorithm.

Finally, it is common that interaction streams are read in a streaming fashion, meaning that one is only allowed to keep a finite portion of the past, and can only read the data a given number of times [1]. This implies computing statistics on the fly, keeping sketches of the data. Streaming algorithmics is a vivid field of research, and methods from this field can apply to link streams.

1. Often only once.

13.1.2 *Algorithmic considerations*

Going further, a theoretical perspective of our work is the identification of classes of link streams, and the general hierarchy between those classes. Researchers on Time-Varying Graphs have done considerable work in this endeavour for TVGs, and knowing to what extent this knowledge is applicable to link streams is yet to explore.

[Casteigts et al., 2012] defines 13 classes of TVGs, and we reproduce the hierarchy obtained between classes in Figure 13.2 ($A \to B$ means that class B is included in class A). Some of these classes can straightforwardly be rewritten in terms of link streams. For example, the class "Complete graph of interaction" corresponds to all link streams $L = (T, V, E)$ such that, for all $u, v \in V \times V$, there is a link (α, ω, u, v) in $E.(V, [\alpha, \omega])$ is a maximal clique of L.

An interesting direction of work is to define more classes, especially on density-based measures. For instance, one could define the class of streams of density p, the class of link streams having a given degree distribution, or the class of link streams with forbidden substreams, and so on.

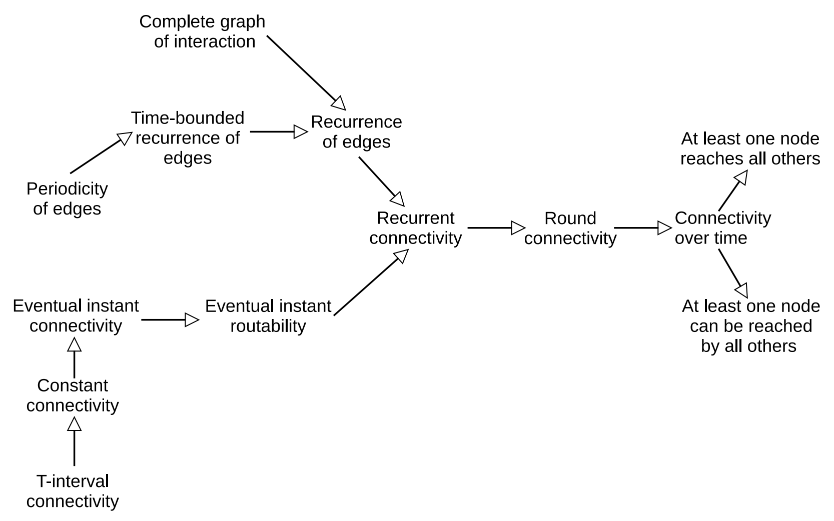

Complete graph of interaction
Time-bounded recurrence of edges
Recurrence of edges
Periodicity of edges
Recurrent connectivity
Round connectivity
Connectivity over time
At least one node reaches all others
At least one node can be reached by all others
Eventual instant connectivity
Eventual instant routability
Constant connectivity
T-interval connectivity

Figure 13.2: Relations of inclusion between classes of Time-Varying Graphs. Reproduced from [Casteigts et al., 2012].

13.2 Modules and dense groups

Many complex objects are of high interest in both graphs and link streams, and developing our theoretical understanding of these objects is interesting. We focus on two examples of such objects: modules and dense groups.

13.2.1 *Modular decomposition*

In graphs, a module is a set of nodes $X \subseteq V$ such that all nodes of X share the same neighborhood outside of X. When $X = V$, or when $|X| = 1$, we call the modules trivial modules. If all modules in a graph are trivial, then the graph is *prime*. Finding the maximal modules, as shown in Figure 13.3 is called *modular decomposition*, and applications can be found in graph compression [Lamarche-Perrin et al., 2014], but also in graph drawing [Papadopoulos and Voglis, 2005].

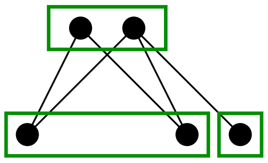


Figure 13.3: Modular decomposition of a small graph. All the nodes in the same box form a module.

With the applicative goal of studying link stream compression, we define modules in a link stream. Intuitively, modules are subsets of $V \times T$. We define a module as a couple $(V', T') \subseteq V \times T$ such that all nodes in V' share the same neighborhood outside V' over T'. In other words, for all u, v in V', $N_{T'}^{V'}(u) = N_{T'}^{V'}(v)$.

This definition is however very stringent, and it seems unlikely that we will find large modules in real-world link streams, given that real-world link streams are typically sparse. We define *relaxed* modules as couples (V', T'), $V' \subseteq V$, $T' \subseteq T$ such that all nodes in V' have similar neighborhoods outside V' over T'.

A common way to assess the similarity of two neighborhoods in a graph is the Jaccard similarity [2], that we extend to link streams. Given the neighborhood of a node $u \in V$ outside of V' on an interval $T' \subseteq T$ [3], we define the Jaccard similarity of two nodes u and v in V' over T' as follows:

2. $\mathcal{J}(u,v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$

3. $N_{T'}^{V'}(u) = \{(v,t) : (b,e,u,v) \in E, v \notin V', t \in [b,e] \cap T'\}$

$$\mathcal{J}_{T'}^{V'}(u,v) = \frac{|N_{T'}^{V'}(u) \cap N_{T'}^{V'}(v)|}{|N_{T'}^{V'}(u) \cup N_{T'}^{V'}(v)|} \qquad (13.1)$$

With this similarity measure, we finally define *ϵ-relaxed* modules. For any ϵ in $[0,1]$, an ϵ-relaxed module is a couple $(V', T') \subseteq V \times T$ such that for all nodes u, v in V', $\mathcal{J}_{T'}^{V'}(u,v) \geq \epsilon$. See Figure 13.4 for an example.

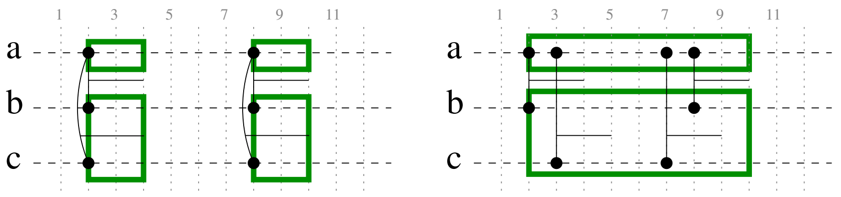


Figure 13.4: ϵ-relaxed modules in a link stream, for $\epsilon = 1$ (left) and $\epsilon < 1$ (right). All the (u,t) in the same box form a module.

Notice that when $\epsilon = 1$, 1-relaxed modules are equivalent to modules.

Once a relaxed modular decomposition $\mathcal{M} = \{(V_i, T_i)\}_{i=1..k}$ of the stream L is obtained, one can define the compressed stream $L_C = (T, V_C, E_C)$, where $V_C = \{V_i\}_{i=1..k}$, and $E_C = \{(\beta, \psi, V_i, V_j) : \exists (b,e,u,v) \in E, [b,e] \cap T_i \cap T_j] \neq \emptyset, u \in V_i, v \in V_j, [\beta, \psi] = T_i \cap T_j\}$. In other words, a link between (V_i, T_i) and (V_j, T_j) in L_C means that nodes in V_i and V_j interact together. Another question of interest is the reconstruction error; it is the error made when trying to build L from L_C.

To enumerate all ϵ-relaxed modules in a link stream, a naive way is to follow a greedy approach: one initializes the algorithm with the minimal modules (*i.e.* $(\{v\}, [t,t]$ for all (b,e,u,v) in E, $t \in [b,e]$). Then, at each iteration, the algorithm picks a module M, and attempts to find modules M' such that $M \subset M'$; if there is none, M is a maximal module. However, this does not form a partition of $V \times T$, a criterion of importance when it comes to compression [Lamarche-Perrin et al., 2014]. Additionally, the complexity of a greedy naive algorithm will be high (at least $\mathcal{O}(2^n m^2)$), whereas one can obtain the modular decomposition of a graph in $\mathcal{O}(n)$ [Tedder et al., 2008]. Another approach is to formulate the problem as an optimization problem. In this case, one may define the area of a module as $\mathcal{A}(M) = \frac{|V'| \cdot |T'|}{|V| \cdot |T|}$, and devising a method that maximizes the area of a module is a promising direction; another promising direction is to minimize the reconstruction error, using information theoretic measures.

Finding relevant values for ϵ is a non-trivial problem; intuitively, on the one hand, the lower ϵ is, the harder it will be to reconstruct the original stream from its compressed version [4]. On the other hand, compressing with a lower ϵ will produce a smaller compressed stream, and as such is more efficient. A first step towards setting ϵ is to formally quantify the reconstruction error in terms of information loss, and then examine the error in function of ϵ. One can expect that the right choice(s) for ϵ will be very dataset-dependent.

4. Notice that even in the case where $\epsilon = 1$, there is a reconstruction error.

13.2.2 Dense groups and communities

A common example of dense groups in graphs are communities. No formal definition of community actually has reached a consensus, but a common intuition is a group of nodes densely connected together, and loosely connected with nodes outside of the community. A partition in communities of a graph $G = (V, E)$ is a partition of the nodes $\{C_i\}_{i=1..k}$ such that the graph induced by C_i is dense, but the graph induced by $C_i \times V \setminus C_i$ is not. If one were to study the interactions in a research laboratory, the communities on the induced graph would typically correspond to the research teams: subsets of people who frequently work together.

In link streams, dense substreams take on a different meaning: while communities in graphs excel at finding groups of people (*i.e.* teams in a laboratory), in link streams, communities are instead meetings, coffee breaks, or discussions: periods of time were individuals are densely connected together.

We have done some work on the study of discussion threads on the Debian user mailing-list [Gaumont et al., 2015]. We analyzed the internal density of threads, as well as their external density. The main result of this work is that threads are typically dense substreams that are loosely connected with the rest of the stream, validating the intuition of community we had beforehand. These results are described more in depth in Appendix A.

However, in most cases, there is no ground truth on existing communities, and automatically detecting communities is a hard problem. Researchers in the *ComplexNetworks* team have extended modularity [5] to link streams, with mitigated results.

5. Modularity is a well-known evaluation function of partitions of graphs, used in the original version of the Louvain algorithm.

Yet, finding quality functions suited to link streams and optimize them, in the spirit of modularity on graphs, is a particularly interesting and promising direction.

Another approach is to extend the work done on cliques to define ϵ-dense groups. A ϵ-dense group is a couple $(X, [b, e])$, $X \subseteq V$, $[b, e] \subseteq T$ such that the density of the link stream induced by X over $[b, e]$, $\delta(L_{b..e}(X))$, is greater than ϵ. Just like with cliques, one is then interested in dense groups that are not included in any other. Notice that finding all maximal ϵ-dense groups of a stream does not form a partition of the links.

The *quotient* graph is another key notion for studying relations between communities in a graph $G = (V, E)$. Given a partition $C = \{C_i\}_{i=1..k}$ of V into communities, in the quotient graph Q_G each node i, $i = 1..k$, represents community C_i and there is a link between nodes i and j, $i \neq j$, if there is a link between a node of C_i and a node of C_j in G. See Figure 13.5 for an illustration. One may add a weight on each link indicating the number of links between communities. Clearly, the quotient graph captures relations between the communities under concern; for example, its density indicates to what extent all communities are linked together.

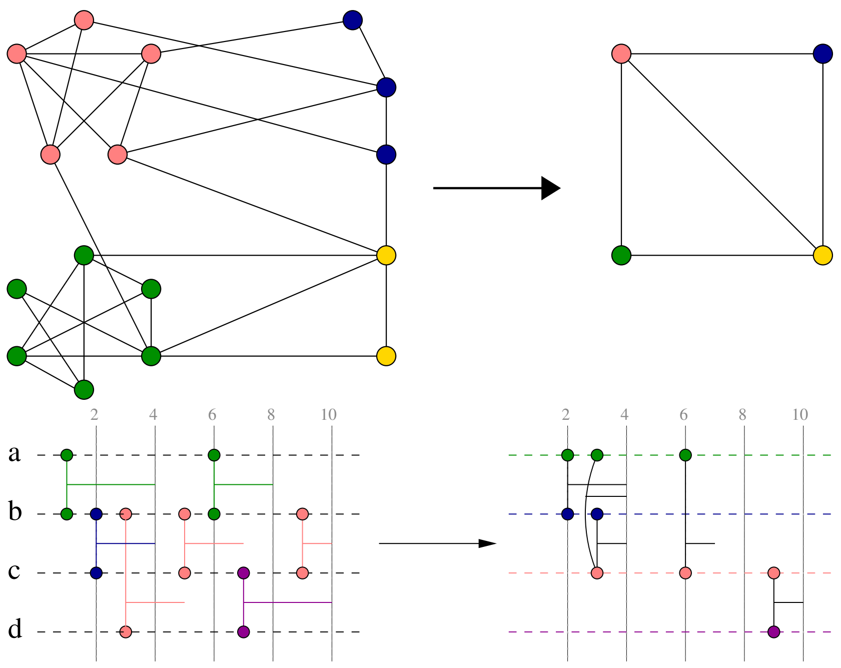


Figure 13.5: **Top:** An example of graph exhibiting communities and its corresponding graph quotient. **Bottom:** An example of link stream with communities and its corresponding quotient stream.

We define the quotient stream induced by a partition $P = \{P_i = (T_i, V_i, E_i)\}_{i=1..k}$ of a link stream $L = (T, V, E)$ in substreams [6] as the stream $Q = (T_Q, V_Q, E_Q)$ such that $(max(b_i, b_j), min(e_i, e_j), P_i, P_j) \in E_Q$ if and only if there exists (b_i, e_i, u, v) in E_i, (b_j, e_j, u, v') in E_j and $[b_i, e_i] \cap [b_j, e_j] \neq \emptyset$. In other words, there is a node $u \in V$ that is involved in the two streams during the same time period.

6. *i.e.* $\cup_i V_i = V$ and $\cup_i T_i = T$.

Studying the quotient stream and its implications on the characterization of dense groups in link streams is a promising direction.

13.2.3 *Generation of synthetic link streams*

In the previous section we talked about modularity, of which the basic principle is to compare the studied graph to a randomized version of the same graph. Comparison to *null*-models is important and helpful in many contexts, especially when one wants to know to what extent a result on a given object is significant. On graphs, a prevalent model is certainly the Erdös-Rényi graph [Erdos and Rényi, 1961]. In an Erdös-Rényi graph $\mathcal{G}_{n,m}$, one provides a number of nodes n, and a number of edges m. Then, m edges are picked uniformly at random between the n nodes. This ensures that the generated graph has a given density. A common variant $\mathcal{G}_{n,p}$ instead sets the probability of existence of any edge to $p \in [0,1]$, with equivalent results.

Yet, one particularly unrealistic aspect of an Erdös-Renyi graph is its degree distribution, which follows a Poisson law when the graph is sparse. In contrast, most real-world graphs display heterogeneous degree distributions. In the study of real-world graphs, one is interested in generating graphs with a prescribed degree distribution $\{k_i\}$ [7], where k_i is the degree of node i.

7. If $\{k_i\} \rightarrow Poisson(\frac{c}{n-1})$, then the graph generated is very similar to an Erdös-Renyi graph.

To generate such graphs given a degree distribution, one typically considers n nodes having k "half-edges" (or stubs), corresponding to the input degree distribution. One then pairs the stubs at random until there is no stub left, and obtains a random graph with this degree distribution. This method is called the *configuration model* However, nothing prevents one from pairing two stubs that had already been paired before, which might create loops or multiple links. See Figure 13.6 for an example. To circumvent this problem, it is common instead to take the original graph $G = (V, E)$ and swap two edges taken at random from E (*i.e.* if (u, v) and (x, y) are taken, create (u, x) and (v, y), and remove (u, v) and (x, y)). This approach has no guarantee on the number of swaps needed to converge to the random case, but studies suggest that empirically, G can be considered randomized after $10 \cdot |E|$ swaps) [Viger and Latapy, 2005].

Some models have already been devised by [Holme and Saramäki, 2012] to randomize link streams.

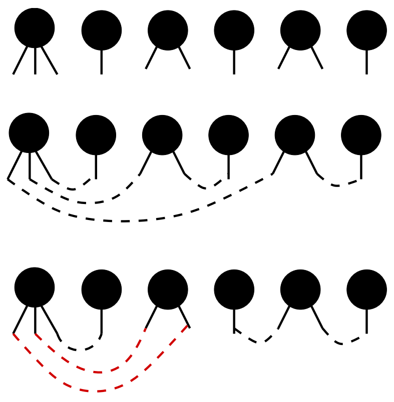


Figure 13.6: **Top:** n nodes with stubs corresponding to their respective degree.
Middle: randomly pairing stubs to create edges.
Bottom: one case of rewiring where multiple links are created.

However, as the authors state themselves, it is next-to-impossible to measure the impact of the randomization on the properties of the stream, and as such these models are unsatisfactory.

One solution is to generate a graph $\mathcal{G}_{n,p}$, then use a distribution (for example Poisson) to generate times of activation for each (u, v). Given that we define a notion of density of link streams, one could imagine generating $\mathcal{L}_{n,p}$, a link stream with fixed density.

Using the same argument, one could think that since there is a notion of degree on link streams, devising an equivalent to the *configuration model* is an easy task. A naive way to do so is the following: consider the degree distribution of a stream L. One then extracts elements (k, d) for all nodes, one element meaning that one node generates k stubs of duration d. One can then pair the stubs randomly, like in graphs. One may think that this simple methodology is a good equivalent of the configuration model.

However, this approach is not as simple as for graphs, and one is at risk of creating overlapping links (*i.e.* two links (b, e, u, v) and (b', e', u, v) such that $[b, e] \cap [b', e'] \neq \emptyset$), as well as pairing edges that have already been paired before. Efficiently avoiding these issues seems hard and computationally costly. For these reasons, it seems preferable to search for swapping algorithms to randomize link streams with prescribed degree distributions.

Ultimately, an important aim is to obtain a hierarchy of generative models, depending on the properties one wants to model. This hierarchy can start with basic features, *i.e.* the number of links, or the distribution of durations, toward more subtle features, *i.e.* degree, density, clustering, and even more subtle, *i.e.* burstiness, centrality, and so on.

13.2.4 Visualization

In recent years, graph visualization has allowed breakthroughs in complex network analysis, both by helping detecting known features (the expected), and by discovering new insights (the unexpected). The core idea is that the human eye is excellent at identifying structures of interest, even in sizeable graphs [8].

8. Usually thousands of nodes or edges.

Visualization is a great helper in the process of *serendipity*, the process of accidental discovery. Exploration through visualization is a adequate way of looking at noisy and heterogeneous data, since it does not require full understanding of the underlying mathematical structure [Keim, 2002].

In the same way that visualizing a graph is insightful of its general structure, being able to visualize link streams is of interest to anyone looking for dense groups, anomalies, or peculiar mesoscopic structures. Though the visualization introduced by [Holme and Saramäki, 2012] is not as flexible as for graphs, the figures show that it is still possible to detect subsets of links worthy of attention.

In the case of real-world large link streams, studies have shown that both a heterogeneous and noisy behaviour are to be expected [Vestergaard et al., 2014]. Visualization then appears as a promising direction for understanding and learning from link streams.

We have developed a simple software tool, *LinkStreamViz*, whose aim is to draw link streams automatically. The code is available on GitHub [9]. Its input is either a text file containing a list of links in the form `t u v`, or a JSON file containing the previous information, as well as other details, such as the color for example. Link durations are not supported at the moment.

9. `http://github.com/TiphaineV/LinkStreamViz`

Figure 13.7 features a full example, including a color cover of the stream in communities. Notice that the tool is only responsible for the drawing, and that the communities have been found externally.

By default, nodes (i.e. horizontal lines) are drawn conditionally to their order of appearance. This gives bloaty drawings (see Figure 13.8, left), and it is possible to give a better order to the nodes either through a text file, or by performing random permutations on nodes' positions and keeping the permutations that minimizes the distance (in our case, the sum of all lengths) between two linked nodes. An example of a better ordering obtained is shown in Figure 13.8, right.

This first heuristic opens a larger perspective, that of assessing the quality of a representation or of an ordering of the link stream.

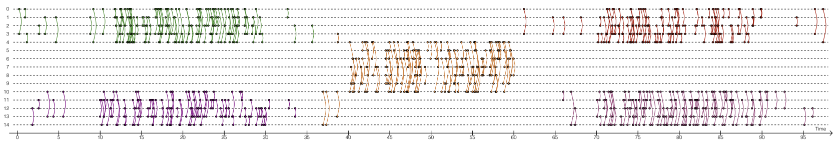


Figure 13.7: A link stream, exhibiting a community behaviour. Communities (in color) have been found externally.

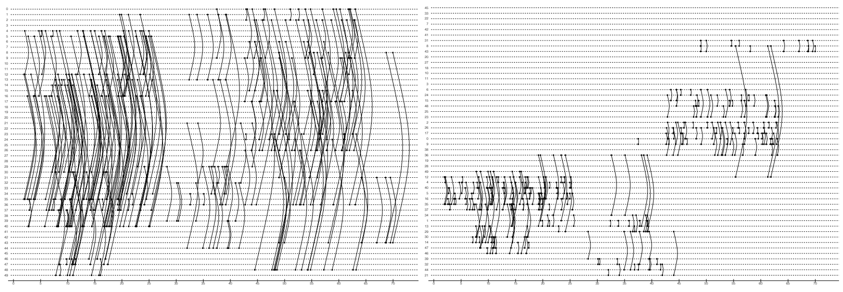


Figure 13.8: Left: a link stream, with random ordering of the nodes. Right: An ordered of the nodes minimizing the sum of link lengths between two connected nodes.

Figures 13.7 and 13.8 show that even a simple visualization can help identifying meaningful groups of links, or give insight on the global structure, and this is the aim of the first version of *LinkStreamViz*.

A potential help for visualization is to reorder nodes according to their relations over time, instead of fixing an order for the whole duration of stream. An example is visible in Figure 13.9. The main drawback is that the nodes become difficult to identify, especially if many nodes change order at the same time.

Finding an appropriate way of visualizing even small link streams, especially with duration, is a question that remains open. Though graphs are high-dimensional objects, these dimensions are only loosely constrained, allowing for many useful and different visualizations. In the case of link streams, the additional temporal dimension carries a lot of meaning, about the causality of events, and special

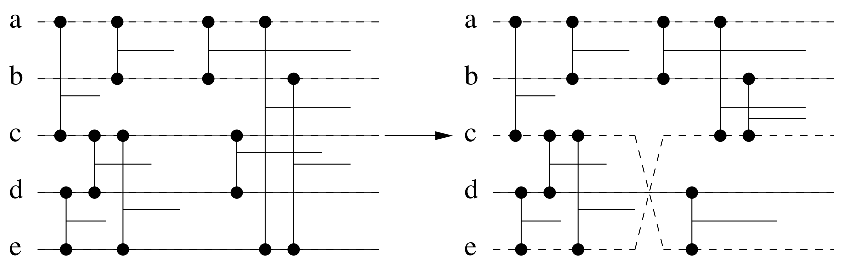


Figure 13.9: A visualization were one is allowed to change the order of nodes over time. In this example, nodes *c* and *e* are swapped midway so that *e* remains spatially close to its neighbors.

attention must be paid to not changing the order of events.

An interesting way to visualize link streams is through movies: one can make a movie of the induced graph, with links appearing and disappearing over time, and unfold the corresponding link stream.

Many improvements are yet to implemented, such as better functions for the node ordering, or notions of quality of representation. A long-term perspective is the emergence of a complete tool for visualizing and manipulating link streams, similar to Tulip or Gephi for graphs.

13.2.5 Event detection in link streams

The work we have done on IP traffic in Part 2 helped us identify perspectives in terms of event detection on link streams.

While large-scale events are of high interest, there is also a considerable number of events that involve few nodes on a diversity of time scales. Detecting such events, and then explaining them, is a challenge that exists in most large datasets.

For example, if your bike gets stolen, it is probably an important event for you, even though dozens of bikes are stolen everyday, and do not constitute an event at the scale of the stream. Another example is a housewarming party: it is likely that at the scale of a city, there are hundreds of such parties everyday, yet for you and your group of friends, it constitutes an event that happens only once every few years.

One can expect such events to be dense sub-streams in the link stream of people-to-people interactions [10]. The it-

10. For example, the ego link stream of a node might be all interactions involving this node.

erative approach we follow allows to remove easily noticeable events, and gives rise to new events that were unseen beforehand. Moreover, the variety of statistics capturing both structure and time gives us a very fine-grained view of interactions, which in turn allows for a more precise identification of events.

13.2.6 Applications

We have devised a theoretical framework to model link streams, and have shown its relevance to IP traffic in Part 2. However, there is a plethora of applicative cases where link streams may bring new results, and we describe some now. This is of course not exhaustive, and one can imagine that any kind of interaction can be modelled with link streams.

Face-to-face contact traces are typically obtained by giving sensors to human beings [11] that activate if two individuals are facing each other. Typically, those streams contain few nodes (less than a few hundreds) and are recorded for varying durations (from a few hours to a few days), which leads to a few thousands of links. These datasets usually exhibit strong structural properties (large cliques, dense groups), and sometimes have enough metadata so that interpretation is eased. In the case of high-school students contacts, we have shown that cliques are typical markers of meetings, or exams, whilst dense groups in the graph typically exhibit the classroom affiliation of each student [Viard et al., 2015b].

11. Or sometimes cats [soc, 2008]!

Another interesting application is software analysis [Latapy and Viard, 2014]. One can model interactions between functions in running code (*i.e.* function u passed a reference to function v at time t) as a link stream. One can then expect a small number of nodes, and many connections between nodes (*i.e.* a very dense induced graph). In this context, active or dense areas in the stream might be indicators of what to put in or out of cache, paths and motifs might give insight on memory leaks, etc. Looking for anomalies in these interactions can point to identifying malwares.

Email exchanges typically contain a large number of nodes, making it closer to classical large, scale-free networks. A link is obtained when u sends an email to v at time t, and studying the structure of these exchanges over time can give insight on the nature of relations between people, or larger groups [Gaumont et al., 2015].

Interactions networks in biology are also important. For example, brain networks contain the electrical interactions between millions of neurones. Animal contact networks are closer to face-to-face human contact traces, have been already widely studied [Sueur et al., 2011].

13.3 Prediction

Link prediction focuses on predicting the links that will appear in the stream, given a known history. Prediction can be based on statistical indicators, motifs indicating causality (*e.g.* when a interacts with b, this triggers an interaction between b and c), and can lead to normal behaviour characterization and anomaly detection – links that have not been predicted deviate from the expected, and are events.

Based on work in link prediction, recommender systems is an interesting perspective. In graphs, recommender systems aim at predicting the preference a user would give to an item [12]. It is a recent and extremely active field of research. In link streams, recommandation can use the combination of structure and time to make more relevant decisions. For example, recommending the peers one should interact with at time t to get a file on a peer-to-peer network, or whom one should email to maximize her chance of getting a reply.

12. For example, suggesting books one might like given her previous readings.

In this last chapter, we have made a short overview of some perspectives opened by the work done in this thesis. Yet, the directions presented in this chapter are far from exhaustive.

Appendices

Appendix A

Analysis of the temporal and structural features of threads in a mailing-list

In this appendix, we present our published work on the study of emails exchanges in the Debian mailing list, and show that threads of emails, like communities in graphs, are dense subsets loosely connected from a link stream perspective.

A.1 Introduction

Exchanges in a mailing list are often studied as complex networks: there is a link between two individuals if they exchange emails. In particular, communities in such complex networks capture groups of friends or close colleagues (individuals that exchange many more messages within the group than outside the group, typically) [Newman, 2004]. However, removing all time information has important consequences if one wants to study the dynamics of email exchanges.

In order to study those dynamics, one may label each link with the frequency of exchanges or the times at which they occur [Sun et al., 2007], but capturing both the structure and the dynamics of exchanges remains challenging. In particular, studying threads calls for methods that capture the temporal nature of interactions more accurately, without loosing the power of network analysis.

We propose here to model email exchanges directly as link streams, *i.e.* series of triplets (t, a, b) meaning that individuals a and b exchanged an email at time t. We then introduce notions that capture both the temporal and structural nature of these exchanges. We use a typical dataset obtained from a public mailing-list archive to illustrate our approach. We analyze this dataset using our model, with a special focus on the properties of threads within the whole archive. Our goal is to understand how the now classical concept of communities in complex networks may translate to threads in link streams representing email exchanges. Indeed, we expect the exchanges of a given thread to involve a specific set of individuals for a specific period of time, thus being dense from both structural and temporal point of views. This is illustrated in Figure A.1.

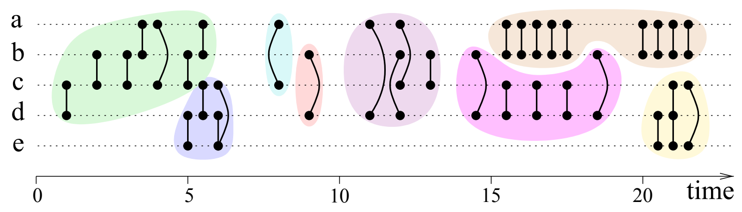


Figure A.1: An example of link stream representing email exchanges between individuals a, b, c, d and e, with threads represented by colored areas. For instance, at time 5, b and c exchange an email, as well as d and e. Threads are *a priori* dense series of exchanges involving a limited group of nodes during a limited period of time.

A.2 Dataset

Archives of exchanges in various mailing-lists are readily available on the web, and studying them provides very rich insights on various issues. They have the advantage of being publicly available in many cases, and some involve large amounts of users over long time periods.

A typical example is provided by Debian mailing list [SPI, 2015]: it contains emails sent from over 51753 email addresses, over 20 years. In addition, exchanges in this mailing-list have been studied in the past [Wang, 2014, Sowe et al., 2006, Dorat et al., 2007]. Finally, this dataset provides the thread information for each message, that we can use as a ground truth. For all these reasons, we use in this paper the Debian mailing list to illustrate and validate our approach.

More precisely, we crawled the Debian mailing list web archive [SPI, 2015]. In this archive, each message is stored as a separate HTML page; for each month, a page links to every message that was sent during this month, and so on for years. From the initial link of the mailing-list, we follow all links on the page corresponding to months, and then to messages. Then, for each message, we extract its author u, the date t at which it was posted (converted into UTC time), and the message it is replying to (through the IN-REPLY-TO entry), which has a corresponding author v. This corresponds to an interaction between u and v at time t in the link stream.

We denote by α the time of the first interaction, and by ω the time of the last recorded interaction, i.e. we selected only messages appearing after (resp. before) this time. Here, α is January 1st, 1996 and ω is December 31st, 2014 We call $\omega - \alpha = 20$ years the *duration* of the dataset.

We obtain a dataset $\mathcal{D}$ of $n = 722716$ emails sent between January 1st, 1996 and December 31st, 2014 from 51753 distinct email addresses.

Each email m in $\mathcal{D}$ has an author (which we identify by its email address in first approximation), denoted by $a(m)$, a time at which it was posted, denoted by $t(m)$, and it may be an answer to a previous message, denoted by $p(m)$. Some messages are not answers to any other message (they are directly sent to the mailing-list), and in this case we state that $p(m) = m$. Such messages are called *root* messages.

Each root message m naturally induces a thread: it is the set $\mathcal{T}(m)$ of messages such that m belongs to $\mathcal{T}(m)$ and if a message m' is in $\mathcal{T}(m)$ then all messages m'' such that $p(m'') = m'$ also belong to $\mathcal{T}(m)$. In other words, $\mathcal{T}(m)$ contains exactly m, the answers to m, the answers to these answers, and so on. The focus of this paper is the study of structural and temporal features of these threads.

Our data contains incomplete threads: the ones that have an email in our dataset but began before and/or continued after the data collection period. Some threads also exhibit incoherences, for instance a reply has a smaller timestamp than the message it replies to. We manually remove those threads, as well as all threads that last for than 2 years, or that start 2 years before the end of our data collection.

After this bias correction procedure, we are left with $n = 316569$ emails, involving 34648 distinct authors over a duration 598532269 seconds (18 years, 11 months and 19 days).

A.3 Framework and notations

Our goal is to study the structural and temporal properties of threads within a mailing-list archive. In order to do so, we propose a model of the data that captures both its temporal and structural nature, and allows for easy manipulation of threads.

We model our mailing-list archive as the link stream $D = (T_D, V_D, E_D)$ with $T_D = [\alpha, \omega]$, $V_D = \{\forall m_i : a(m_i), a(p(m_i))\}$ and $E_D = \{\forall m_i : (t(m_i), a(m_i), a(p(m_i)))\}$ where $(m_i)_{i=1..k}$ is the sequence of messages in our dataset. In other words, a triplet (t_i, u_i, v_i) in E_D indicates that individual u_i answered to an email of individual v_i at time t_i, for all i.

Such a link stream naturally contains sub-streams: $L' = (T', V', E')$ is a substream of $L = (T, V, E)$ if and only if $T' \subseteq T$, $V' \subseteq V$ and $E' \subseteq E$. In other words, all the interactions of L' also appear in L. Given a set of nodes S, we define the sub-stream $L(S)$ of L induced by S as the largest sub-stream of L such that all the links in $L(S)$ are between nodes in S.

Each thread $\mathcal{T}(m)$ in our mailing-list archive modeled as link stream D is naturally modeled as sub-stream $T(m)$ of D. See Figure A.1.

Any link stream $L = (T, V, E)$ also induces a graph $G = (V_G, E_G)$ where $V_G = \{u : \exists t \in T, v \in V \text{ s.t. } (t, u, v) \in E\}$ and $E_G = \{(u, v) : \exists t \in T \text{ s.t. } (t, u, v) \in E\}$. In our case, the whole mailing-list archive induces the graph $G(D)$ among authors of emails, and each thread T induces a sub-graph $G(T)$ of $G(D)$.

In a graph $G = (V, E)$, a community structure is defined by a partition $C = \{C_i\}_{i=1..k}$ of V into k communities. In other words, $\bigcup_i C_i = V$ and $C_i \cap C_j = \emptyset$ whenever $i \neq j$. In a similar way, one may consider a link stream $L = (T, V, E)$ and a partition of its links into k sub-streams $P = \{P_i = (T_i, V_i, E_i)\}_{i=1..k}$. In other words, for any $(t, u, v) \in E$, there exists a unique j between 1 and k such that (t, u, v) is a link of E_j.

The threads in our email dataset are exactly a partition of the whole stream, which we denote by $T = \{P_i\}_{i=1..k}$ where k is the number of threads and each P_i is a sub-stream representing a thread (with our notations above, there exists a message m such that $P_i = T(m)$).

Notice that, although the threads are a partition of the whole stream, their induced graphs may overlap: some nodes and links of $G(D)$ belong to several sub-graphs $G(T_i)$. As a consequence, threads do not induce a partition of $G(D)$ into communities. Instead, one may see the partition of D into threads as a community structure, and this is the focus of our work.

Notice finally that we consider that links are undirected (i.e. $(t, u, v) = (t, v, u)$ and happen at an instant in time (regardless, for instance, of when the message is read).

A.4 Basic statistics

In this section, we present the basic statistics describing the threads in our dataset and the whole archive.

The most basic description of our data certainly is the number of links (*i.e.* emails) they contain, the number of distinct nodes (*i.e.* authors) involved, the number of distinct links they contain (distinct pairs of authors in direct interaction), and their duration (time from the first email to the last one).

The whole archive contains 722716 emails involving 51754 authors (and 260392 distinct pairs of authors), over

654094800 seconds (20 years). After our filtering procedure, we are left with $n = 316569$ emails, involving 34648 distinct authors (and 226879 distinct pairs of authors) over a duration 598532269 seconds (18 years). Figure A.2 display the distribution of these values for each thread.

Though the largest thread lasts long (more than a year), most threads and contained within a few days (100000 seconds is a bit more than 24 hours). Similarly, the largest thread involves 100 messages, though all intermediate sizes are represented in the dataset. Most threads are very short and involve less than 3 messages.

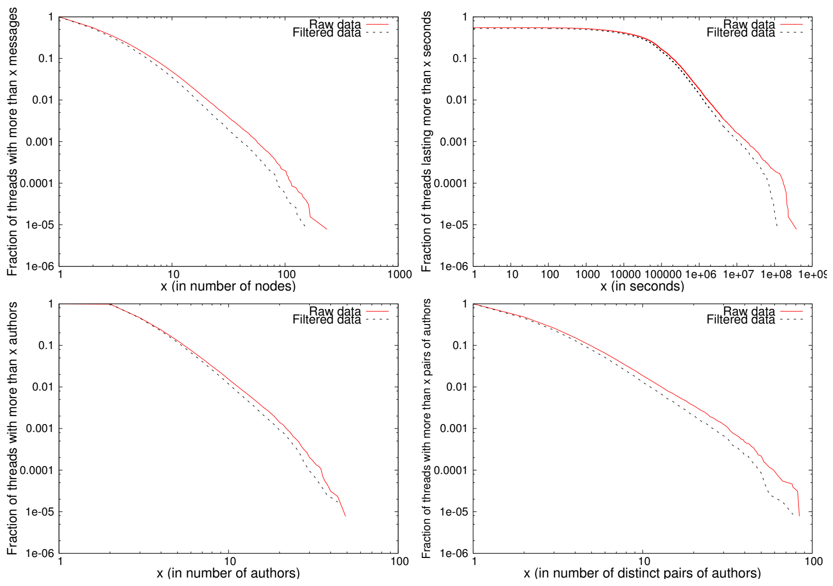


Figure A.2: Complementary cumulative distributions for basic statistics of our raw (solid line) and filtered (dotted line) dataset. Top left: thread sizes (number of messages per thread); top right: thread durations (time elapsed between the first and the last message of the thread); bottom left: number of distinct authors; bottom right: number of distinct pairs of authors.

In order to gain more insight, we show correlations between some of these basic statistics in Figure A.3. Figure A.3 (left) shows that thread duration and size are correlated (the larger a thread is, the longer it is likely to be); notice however that for small-sized threads, all types of durations are represented. Looking at the correlations between the size of threads and the number of distinct authors involved shows that threads nearly always involve more messages than authors. This is a typical feature of mailing-lists [Dorat et al., 2007] and as such is dataset-dependent.

In a link stream $L = (T, V, E)$ with $E = \{(t_i, u_i, v_i)_{i=1..n}\}$

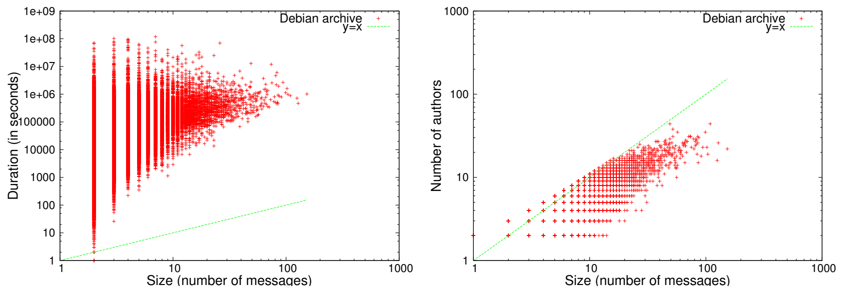


Figure A.3: Left: Correlations between size and duration of threads. Right: Correlations between size of threads and the number of authors involved.

for all i starting at time α and ending at ω, the inter-contact time series is the series $\tau = (\tau_i)_{i=0..n}$ in which $\tau_0 = t_1 - \alpha$, $\tau_{n+1} = \omega - t_n$ and for all i from 1 to $n-1$, $\tau_i = t_{i+1} - t_i$ []. The inter-contact time series of a pair of nodes u and v is the inter-contact time series of the sub-stream $L(\{u,v\})$ they induce. In other words it is the series of times elapsed between two consecutive occurrences of a link between them.

Figure A.4 shows the inter-contact times distribution in the Debian mailing list.

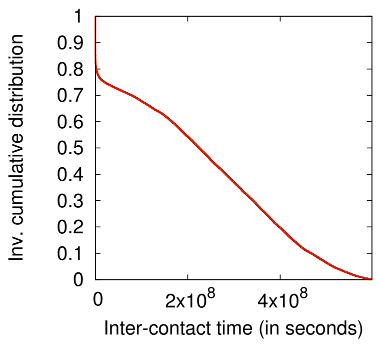


Figure A.4: Inter-contact times distribution in the Debian mailing list dataset.

A.5 Interactions within threads

The key feature of communities is the fact that they form dense subgroups. This section is therefore devoted to the study of density of interactions within threads, from both structural and temporal point of views.

A.5.1 *Density of threads*

In a graph, the density is the probability that two randomly chosen nodes are linked together. In other words, it captures the extent at which *all* nodes are directly connected to each other. The density of the graph $G(D)$ induced by our dataset is 3.139×10^{-4}.

In [Viard and Latapy, 2014b], we introduced the notion of Δ-density to capture a similar intuition in link streams, involving both structure and time. Indeed, given a duration Δ, the Δ-density of link stream L is the probability that a link appears between two randomly chosen nodes during a randomly chosen time interval of duration Δ. It captures the extent at which all nodes are directly connected to each other at least every Δ time units. Formally, it is defined as:

$$\delta_\Delta(L) = 1 - \frac{2 \cdot \sum_{u,v \in V, u \neq v} \sum_{t \in \tau(u,v)} \max(0, t - \Delta)}{|V| \cdot (|V| - 1) \cdot \max(0, \omega - \alpha - \Delta)}$$

where $\tau(u,v)$ denotes the inter-contact times between u and v, and α and ω are the start and end time of the link stream.

In order to study the Δ-density in our data, we first have to choose an appropriate Δ. We use here several values which capture email dynamics at different scales: $\Delta = 1$ minute, 1 hour, 1 day, 1 week, 1 month, 1 year and 20 years (the whole duration of the dataset). Figure A.5 displays the evolution of the Δ-density of the stream for all theses values of Δ. It shows that the Δ-density is small for small Δs, and converges to the density of the graph induced by the email exchanges (in our case, $3.139 \cdot 10^{-4}$).

In Figure A.5, the inflexion points give information on the values of Δ where the dynamics change. Still, looking at the density of the whole stream is very coarse and yields little information. A finer approach consists in looking at the Δ-density of relevant sub-streams. In our case, the threads between authors are a natural object to study.

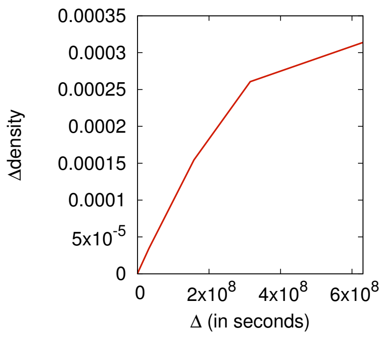


Figure A.5: Evolution of the Δ-density of the link stream for Δ from 1 second to 20 years.

A.5.2 *Intra-thread density*

More globally, given a graph $G = (V, E)$ and a partition $C = \{C_i\}_{i_1..k}$ of V into k communities, the density within communities of C is captured by the *intra-community density*:

$$\frac{2 \cdot \sum_i |\{(u,v) \in E, u \in C_i \text{ and } v \in C_i\}|}{\sum_i |C_i| \cdot (|C_i| - 1)}$$

In other words, intra-community density is the probability that two nodes chosen at random in the same community are linked together.

In our case, this notion does not directly make sense: as already noticed, we do not have communities defined on $G(D)$ since the graphs induced by threads overlap. However, we extend the notion of intra-community density to link streams as follows. The intra-thread Δ-density is the probability that two randomly chosen authors contributing to the same thread are linked together within a randomly time interval of duration Δ, for a given Δ:

$$1 - \frac{2 \cdot \sum_i \sum_{u,v \in V_i, u \neq v} \sum_{t \in \tau_i(u,v)} \max(0, t - \Delta)}{\sum_i |V_i| \cdot (|V_i| - 1) \cdot \max(0, \omega_i - \alpha_i - \Delta)}$$

where V_i is the set of authors involved in thread P_i, α_i is the time of the first message in the thread (i.e the minimal t such that there exists a $(t, u, v) \in E_i$) and ω_i is the time of the last message in the thread (i.e the maximal t such that there exists a $(t, u, v) \in E_i$).

In our data, the inverse cumulative distribution of intra-thread Δ-densities are in Figure A.6 (left) for several values of Δ ranging from 1 minutes to 1 year. For each point on the x-axis, the plot gives the proportion of threads in the mailing-list that have an intra-thread Δ-density higher than x. As expected, the higher the Δ used, the higher the density is. However, there is no significant change between a Δ of 7 days and a Δ of 1 year.

Moreover, these distributions confirm that the interactions within threads are much denser (both structurally and temporally) than in the global mailing-list. Indeed, the median intra-thread Δ-density ranges from 2.69×10^{-4} to 0.28 while the link stream Δ-density ranges from 1.05×10^{-10} to 3.42×10^{-5}. The intra-thread Δ-density typically is 10^5 times larger than the global Δ-density.

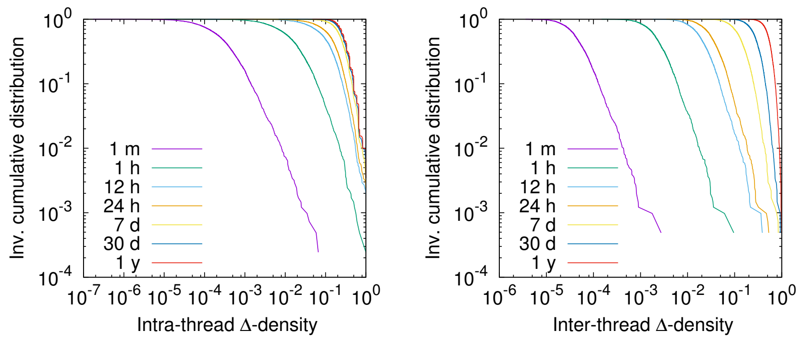


Figure A.6: Left: Inverse cumulative distributions of values of intra-thread Δ-density for different Δs. Right: Inverse cumulative distributions of values of inter-thread Δ-density for different Δs.

This shows that threads are indeed dense substreams in our link streams.

A.6 Relations between threads

In the previous section, we focused on structural and temporal properties *inside* threads, compared to the whole link stream. We now turn to the study of relations *between* threads.

A.6.1 Inter-thread density

Let us first study the density of relations between threads in a way similar to above. Given a graph $G = (V, E)$ and a partition $C = \{C_i\}_{i_1..k}$ of V into k communities, the inter-community density is the probability that two nodes chosen at random in two different communities are linked together:

$$\delta^{inter}(C_i) = \frac{1}{|C|} \sum_{j,i \neq j} \frac{|\{(u,v) \in E \text{ s.t. } u \in C_i \text{ and } v \in C_j\}|}{|C_i| \cdot |C_j|}$$

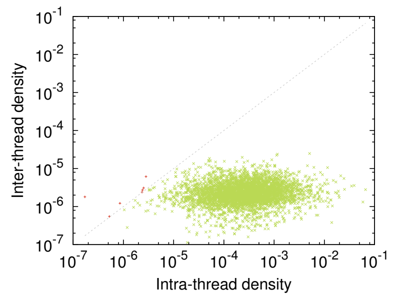
Again, this notion does not directly make sense in link streams, as threads do not induce a partition of nodes. As a consequence, we introduce the inter-thread Δ-density as the probability that two randomly chosen nodes in different communities are linked together during a time interval of duration Δ chosen at random during the time duration of both threads.

Let us define the inter-thread substream between a thread P_i and a thread P_j: $L_{ij} = (T_{ij}, V_{ij}, E_{ij})$, with $T_{ij} = [\min(\alpha_i, \alpha_j), \max(\omega_i, \omega_j)]$, $V_{ij} = V_i \cup V_j$ and $E_{ij} = \{(t,u,v) : t \in T_{ij}, u, v \in V_{ij}, (t,u,v) \in E \setminus E_i \cup E_j\}$. In other words, this is the substream containing the links between nodes of P_i or P_j that are not involved in threads P_i and P_j. The inter-thread density between P_i and P_j is the Δ-density of L_{ij}. In order to obtain the inter-thread Δ-density of P_i to all other threads, we simply average the inter-threads Δ-densities of P_i and all other threads. More precisely:

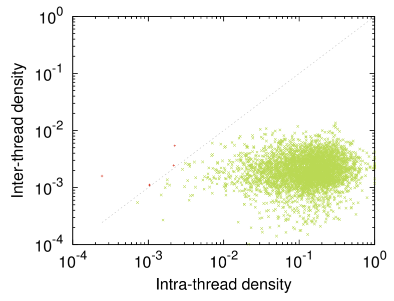
$$\delta_\Delta^{inter}(C_i) = \frac{1}{|C|} \sum_{j,i \neq j} \delta_\Delta(L_{ij})$$

In our data, the inverse cumulative distribution of inter-thread Δ-densities are displayed in Figure A.6 (right) for different values of Δ. For each point on the x-axis, the plot gives the proportion of threads in the mailing-list that have an intra-thread Δ-density higher than x. Again, larger Δ correlates with larger Δ-densities. However, the inter-thread Δ-density does not plateau, even for large values of Δ. This is natural, since the number of links considered in the computation of the inter-thread Δ-density naturally grows with Δ.

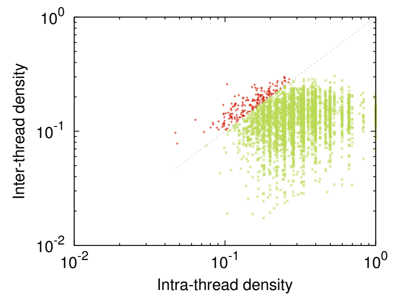
In Figure A.7, the correlations between the inter- and intra-thread Δ-density are plotted for some values of Δ. As expected, intra-threads are denser than inter-threads. This relation holds as Δ is bigger, even though the difference between inter and intra thread Δ shrinks. Figure A.7(d) that for $\Delta = 20$ years, the difference is non-existent. This is due to the fact that the bigger the Δ, the less the temporal characteristics of threads are important.



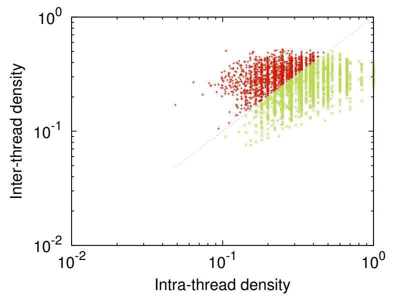
(a) Δ =1 minute



(b) Δ =1 day



(c) Δ =1 year



(d) Δ =20 year

Figure A.7: Correlations between inter- and intra-thread densities for different values of Δ.

A.6.2 Graphs between threads

The *quotient* graph is another key notion for studying the relations between communities in a graph $G = (V, E)$. Given a partition $C = \{C_i\}_{i=1..k}$ of V into communities, in the quotient graph $\overline{G}$ each node i, $i = 1..k$, represents community C_i and there is a link between two nodes i and j, $i \neq j$, if there is a link between a node in C_i and a node in C_j in G. See Figure A.8 for an illustration. One may add on each link a weight indicating the number of links between communities. Clearly, the quotient graph captures relations between the communities under concern; for instance, its density indicates up to what point all communities have links between them.

Relations between sub-streams L_i, $i = 1..k$, may have dif-

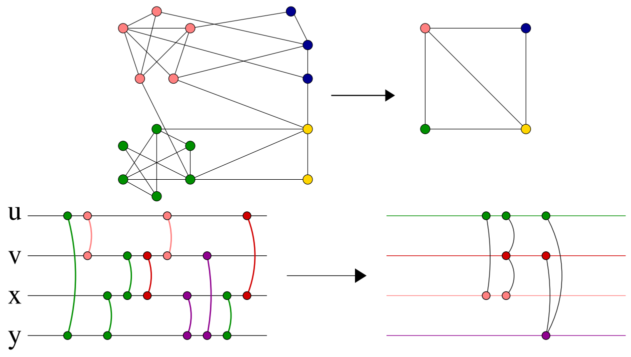


Figure A.8: Top: An example of graph exhibiting communities and its corresponding graph quotient. Bottom: An example of link stream with communities and its corresponding stream quotient.

ferent forms, and in particular they have a temporal and a structural nature. In order to capture the temporal relations between sub-streams, one may define the temporal overlap graph as follows: $X = (V, E)$ with $V = \{i, i = 1..k\}$ and there is a link (i, j) in E whenever P_i and P_j have a temporal intersection (*i.e.* $[\alpha_i, \omega_i] \cap [\alpha_j, \omega_j] \neq \emptyset$). Likewise, one may define the node overlap graph as follows: $Y = (V, E)$ with $V = \{i, i = 1..k\}$ again and there is a link (i, j) in E whenever there is a node v involved in both P_i and P_j (*i.e.* there exists a t, a t'n a u and a u' such that there is a link (t, u, v) in P_i and a link (t', u', v) in P_j.

These graphs encode much information about relations between threads. For instance, the degree of node i in T is the number of threads active at the same time as P_i.

We display in Figure A.9 (left) the correlations between the degree in X and the thread size. Comment...

Figure A.9 (right) shows the correlations between the degree in Y and the thread duration. Comment...

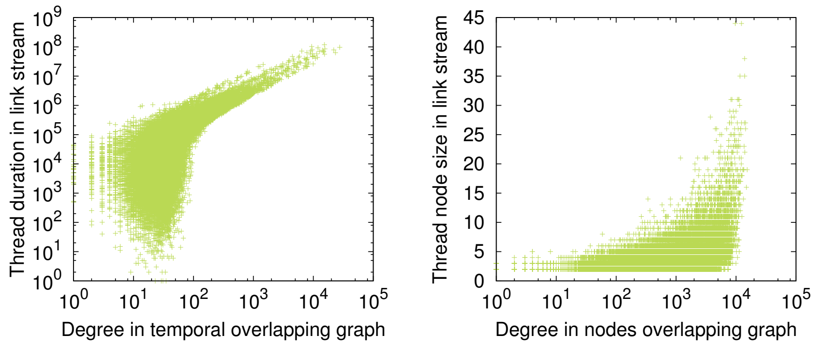


Figure A.9: Left: Correlation between the degree in the time overlap graph X and the thread size. Right: Correlation between the degree in the node overlap graph Y and the thread duration.

A.6.3 Quotient stream

To deepen our understanding of our data, we capture here the both temporal and structural nature of relations between sub-streams as follows. We define the quotient stream induced by a partition $P = \{P_i = (T_i, V_i, E_i)\}_{i=1..k}$ of link stream L as the stream $Q = (T_Q, V_Q, E_Q)$ such that $(P_i, P_j, t) \in E_Q$ if and only if there exists (u, v, t_1) in E_i, (u, v', t_2) in E_i and (u, v'', t) in E_j with $t_1 \leq t \leq t_2$. In other words, there is a node u that has a link within P_j occurring between two of its links in P_i. This means that u is involved in the two streams during the same time period.

The stream quotient induced by the threads in our dataset has 12281269 links and involves 68524 distinct nodes (i.e. threads). Since our dataset contains 116999 threads, this implies that 48475 threads are not in relation with any others.

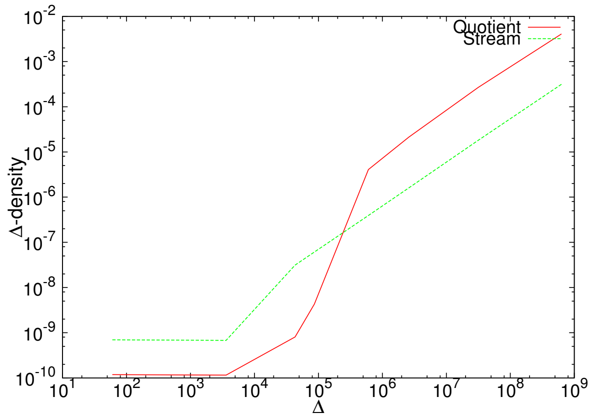


Figure A.10: Δ-density of the link stream and the stream quotient as a function of Δ, for $\Delta = 1mn, 1h, 12h, 1d, 37d, 30d, 1y$ and $20y$.

Figure A.10 shows the Δ-density of the quotient stream and the Δ-density of the original stream for different values of Δ. The quotient is not very Δ-dense, i.e. threads are not densely connected together, though it is slightly denser than the stream for large values of Δ. This is comparable to graphs.

A.7 Conclusion

Through the prism of link streams, we have studied the email exchanges in the Debian mailing list over 20 years.

From Δ-density, we define notions of *inter* thread density, *intra* thread density and quotient stream, that are generalizations of the equivalent notions in graphs. We show the relevance of these notions on a real-world dataset of email exchanges; however, further understanding of these notions is necessary.

We have shown that threads in the mailing list are Δ-dense substreams from a link stream perspective, just like communities are dense subgraphs from a graph perspective.

Though threads are readily identified in the Debian mailing list archive, this is usually not the case. Detecting dense substreams loosely interconnected without *a priori* knowledge remains a challenge.

Our scheme is dependant of a parameter Δ, that has to be chosen externally. Considering links with durations (i.e. (b, e, u, v), meaning that u and v are interacting continuously from b to e) instead of punctual links is a promising direction of work.

Appendix B

Identifying Roles in an IP Network with Temporal and Structural Density

In this appendix we present the published work proposing a notion of density that captures both temporal and structural features of interactions, that generalizes the classical notion of clustering coefficient.

We use it to point out important differences between distinct parts of the traffic, and to identify interesting nodes and groups of nodes in terms of roles in the network.

B.1 Introduction

Measurement, analysis and modeling network traffic at IP level has now become a classical field in computer networking research [Fraleigh et al., 2003, Qadeer et al., 2010, Klemm et al., 2003]. It relies on captures of traffic traces on actual networks, leading to huge series of packets sent by machines (identified by their IP adress) to others. It is therefore natural to see such data as graphs where nodes are IP adresses and links indicate that a packet exchange was observed between the two corresponding machines. One obtains this way large graphs which encode much information on the structure of observed exchanges, and network science is the natural framework for studying them [Iliofotou et al., 2009b, Eberle and Holder, 2007].

One key feature of network traffic is its intense dynamics. It plays a crucial role for network optimization, fault/attack detection and fighting, and many other applications. As a consequence, much work is devoted to the analysis of this dynamics [Abry et al., 2002b, Fontugne et al., 2010c, Guralnik and Srivastava, 1999b, Crovella and Kolaczyk, 2003]. In network science, studying such dynamics means that one studies the dynamics of the associated graphs [Broido and Claffy, 2001]. The most common graph approach relies on series of snapshots: for a given Δ, one considers the graph G_t induced by exchanges that occured in a time window from t to $t + \Delta$, then the graphs $G_{t+\Delta}$, $G_{t+2\Delta}$, and so on [Lee and Maggioni,]. Many variants exist, but the baseline remains that one splits time into (possibly overlapping) slices of given (but possibly evolving) length Δ [Basu et al., 2010].

Obviously, a key problem with this approach is that one must choose appropriate values of Δ: too small ones lead to trivial snapshots, while too large ones lead to important losses of information on the dynamics. In addition, appropriate values of Δ may vary over time, for instance because of day-night changes in activity. As a consequence, much work has been done to design methods for choosing and assessing choices in the value of Δ [Benamara and Magnien, 2010b, Caceres and Berger-Wolf, 2013, Aynaud and Guillaume, 2011]. In [Caceres and Berger-Wolf, 2013, Aynaud and Guillaume, 2011, Hulten et al., 2001b], the authors even propose methods to choose values of Δ that vary over time, or to consider non-contiguous time windows. In all situations, however, authors assume that merging all the events occurring at a same time is appropriate.

On the countrary, we argue that there are interactions in IP traffic that occur concurently but at different time scales, and that they should not be merged. For instance, users interacting with a system will have a faster dynamics than a backup service that automatically saves data every

24 hours, and a slower dynamics than a P2P system or a large file transfer between two machines. Likewise, attacks may have dynamics that distinguish them from legitimate traffic [Zhou et al., 2009]. This means that different parts of the traffic may have different appropriate values of Δ, even though they occur at the same time (or in the same time window). These interactions are different in nature; they reflect different roles for involved nodes (like an end-user machine, or a backup server) that should be studied separately to accurately reflect the actual activity occurring in the network.

We propose in this paper an approach for doing so. It relies on a notion of Δ-density that captures up to what point links appear *all the time* and/or all possible links between considered nodes occur *all the time* (Section B.2). To this regard, it may be seen as a generalization of classical graph density and its local version, clustering coefficient. We show how this notion may be used to identify one or several appropriate time scales for various parts of the traffic, and how mixing time and structure makes it possible to identify (groups of) machines playing specific roles in a network (Section B.3). All along this paper, we illustrate and validate our approach using two real-world captures of traffic on a firewall between a local network and the internet. It consists of packets that were observed on the firewall in a time period of one month.

B.2 Notion of Δ-density

We first present the framework and notations we use in the whole paper. Then we define the Δ-density of one link and finally we extend it to sets of links and nodes.

B.2.1 Framework

We model a trace of IP traffic as a link stream $L = (l_i)_{i=1..n}$ where $l_i = (t_i, u_i, v_i)$ means that we observed at time t_i a packet from u_i to v_i. Such a stream comes from a capture started at time α and stopped at time ω, and so $\alpha \leq t_i < \omega$

for all i. We assume in addition that the stream is temporally ordered: for all i and j, $i < j$ implies $t_i \leq t_j$. We call n the *size* of L and denote it by $|L|$. We call $\overline{L} = \omega - \alpha$ its *duration*.

A link stream S is a substream of L if there exists a function σ such that for all $i = 1..|S|$, $s_i = l_{\sigma(i)}$, and for all $i = 1..|S| - 1$, $\sigma(i) < \sigma(i+1)$. In other words, all the links in S also appear in L and they are in the same order. We denote by $S \subseteq L$ the fact that S is a substream of L.

Given a pair of nodes u and v, we denote by $L(u,v)$ the substream of L induced by (u,v), namely the largest substream (t_i, u_i, v_i) such that for all i, $u_i = u$ and $v_i = v$. By extension, given any set of pairs of nodes we define the substream $L(S)$ induced by S as $L(S) = \cup_{(u,v)\in S} L(u,v)$. For any given set of nodes S we define $L(S)$ the substream induced by S as $L(S) = L(S \times S)$.

The graph $G(L)$ induced by stream L is defined by $G(L) = (V(L), E(L))$, where $V(L) = \{u_i, \exists v_i, t_i, (u_i, v_i, t_i) \in L\}$ and $E(L) = \{(u_i, v_i), \exists t_i, (u_i, v_i, t_i) \in L\}$. In our case, $V(L)$ is the set of observed IP adresses, and there is a link (u,v) in $E(L)$ if and only if we observed a packet from u to v. As discussed in the introduction, IP traffic and other link streams are often studied through this induced graph.

B.2.2 Δ-density of links

Suppose a Δ between 0 and $\overline{L}$ is given. We first define the Δ-density of a pair of nodes u and v, that we denote $\delta_\Delta(u,v)$. If there is no link involving them in L, *i.e.* $|L(u,v)| = 0$, then we state that their Δ-density is zero: $\delta_\Delta(u,v) = 0$. Now let us assume that at least one link involving u and v occurs.

There is no significant structure in just one link, and so the Δ-density of (u,v) is only defined with respect to time. It captures up to what point (u,v) appears in every time interval of size Δ in L. To do so, we compute the fraction of non-overlapping time intervals of size Δ that contains no occurrence of the link. More formally:

$$\delta_\Delta(u,v) = 1 - \frac{\left\lfloor \frac{t_1-\alpha}{\Delta} \right\rfloor + \left\lceil \frac{\omega-t_n}{\Delta} \right\rceil - 1 + \sum_i \left\lceil \frac{t_{i+1}-t_i}{\Delta} \right\rceil - 1}{\left\lceil \frac{\omega-\alpha}{\Delta} \right\rceil - 1} \tag{B.1}$$

where t_i denotes the time at which (u,v) occured for the i-th time. The numerator counts the number of non-overlapping intervals of size Δ that contain no occurrence of (u,v): the number of such intervals between the beginning of the stream and the first occurrence (at t_1), plus the number between the last occurrence (at t_n) and the end of the stream, plus the number between any pair of consecutive occurrences. This is illustrated in Figure B.1. The denominator counts the total number of non-overlapping intervals of size Δ, thus ensuring that the Δ-density is always between 0 and 1. It reaches 1 if and only if a link between u and v appears at least every Δ time, and it is closer and closer to 0 as more and more intervals of size Δ contain no such link. As stated above, it is exactly 0 when no link involving u and v occurs.

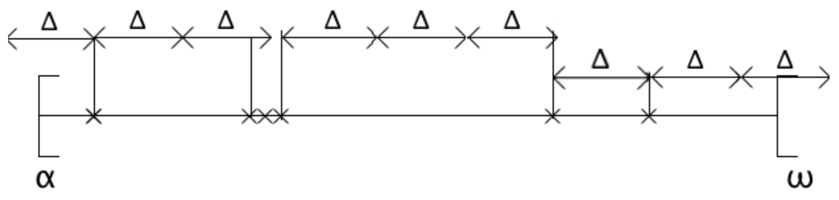


Figure B.1: Counting of the number of non-overlapping time intervals of a given size Δ that contain no occurrence of a pair of nodes. Each cross represents an occurrence of the pairs of nodes on the time line.

In order to extend the notion of Δ-density to any set S of pairs of nodes, we define it as the average of the Δ-density of the elements of S:

$$\delta_\Delta(S) = \frac{\sum_{(u,v)\in S} \delta_\Delta(u,v)}{|S|} \tag{B.2}$$

This notion still captures no notion of structure and only focuses on temporal aspects: it measures up to what point interactions between pairs of nodes in S occur (at least) every Δ time.

B.2.3 *Δ-density of streams and sets of nodes*

In a classical (undirected, simple) graph $G = (V, E)$, the density captures the extent at which every node is connected to all others: $\delta(G) = \frac{2 \cdot m}{n \cdot (n-1)}$ where $n = |V|$ is the number of nodes and $m = |E|$ is the number of links. In other words, it measures the extent to which all possible links exist.

In a link stream S, we mix this structural point of view with the temporal aspects captured above as follows:

$$\delta_\Delta(S) = \frac{2 \cdot \sum_{(u,v) \in V \times V} \delta_\Delta(u,v)}{|V| \cdot (|V| - 1)} \tag{B.3}$$

where V is the set of nodes involved in S. In other words, the Δ-density of a link stream captures the extent at which all possible links occur (at least) every Δ time in the stream. It is the average of the Δ-density of all possible pairs of nodes, including the ones which do not interact in the stream.

Finally, just like one often studies the density of subgraphs induced by a given set of nodes, we define the Δ-density of any set S of nodes as $\delta_\Delta(L(S))$, which capture the both structural and temporal intensity of interactions among nodes in this set. It is equal to 1 only if all nodes interact with each another, and do so at least every Δ time. It decreases whenever two nodes in the set do not interact or a time interval between two occurrences of a link is greater than Δ.

We call this a Δ-*clique*: just like cliques are graphs with maximal density in classical graph theory, Δ-cliques are streams with maximal Δ-density. Notice that the Δ-cliques of a stream necessarily induce cliques in the graph induced by the stream.

B.3 Identifying roles

We show in this section how our notion of Δ-density may be used to identify distinct roles in a capture of IP traffic. We typically aim at identifying backup servers, user machines, or distributed applications. We first present the datasets we use for our experimentations, then explain how to compute a characteristic time for links and groups of links, and explore a notion of clustering coefficient that combines time and structure. We finally discuss how obtained results may be used for identifying roles in the network.

B.3.1 Our datasets

We rely for our experimentations on two datasets collected in 2012. Both datasets consist of a one-month capture of the headers of all IP packets managed by a firewall between a large local network and the internet. They are however quite different in their key features, which makes it interesting to consider them jointly.

The first dataset, which we model by the link stream $A = (a_i)$, contains 6 million timestamped links. They involve 183 distinct pairs of nodes, between 129 distinct nodes. The second dataset, which we model by the link stream $B = (b_i)$ contains 140 299 timestamped links. They involve 60 330 distinct pairs of nodes, between 38 571 distinct nodes. It therefore appears clearly that, although more exchanges occur in A than in B, these exchanges are between a much smaller number of nodes than the ones in B.

B.3.2 Identifying relevant Δ

Our approach relies on the identification of relevant values of Δ that may reveal the dynamics of links, nodes, and larger parts of the stream. To identify such values, we compute the Δ-density for various values of Δ and observe the variations of the Δ-density as a function of Δ. More precisely, we consider $\Delta = 1.01^i$ for all i such that Δ is between 1 second and the duration of the whole capture (namely $\omega - \alpha = 2808927s$).

The exponential growth in the considered values of Δ deserve explanations. Indeed, we want to be able to identify interesting values which are orders of magnitudes of differences, like one second and one day. In addition, there is a significant difference between $\Delta = 1s$ and $\Delta = 30s$, while we make no significant distinction between $\Delta = 24h = 86400s$ and $\Delta = 24h + 30s = 86430s$. This is exactly what an exponential growth of Δ captures. We chose 1.01 to have a large enough number of points in our plots to allow accurate observation, while remaining reasonable (we obtain here 1118 points).

Notice that the Δ-density of a given pair of nodes (u, v) necessarily grows to 1 when Δ grows, as long as it occurs at least once in the stream (otherwise it is equal to 0 independently of Δ). Indeed, for small Δ it is close to 0, as almost no time interval of size Δ contains an occurrence of the link. When Δ grows, the number of intervals with no such link decreases, and so the Δ-density grows. When Δ reaches its maximal value, *i.e.* the duration of the whole stream, then there is clearly no interval at all that contains no occurrence of the link, and so the Δ-density reaches 1.

When we consider the Δ-density of a set of links, the same remarks hold. When we consider the case of a link stream or the case of a set of nodes, though, the situation is different. Indeed, in these cases the pairs of nodes that never occur are taken into account and lower the value of the Δ-density. Then, the Δ-density still grows when Δ grows, but its maximal value is the (classical) density of the induced graph and it is reached when Δ equals the whole duration of the stream. Then, the Δ-density of each individual pair of nodes is either 0 (if it never occurs) or 1 (if it occurs at least once), and the formulae defining the Δ-density are reduced to the formula for the density of the graphs, see Section B.2.

Figure B.2 presents the evolution of the Δ-density of link streams A and B presented above, as Δ grows.

The plots show clearly that the Δ-density of A increases sharply at $\Delta \sim 10^3$ and $\Delta \sim 10^5$, indicating that these durations play an important role in this dataset. The plot for B instead, grows smoothly towards its maximum. It

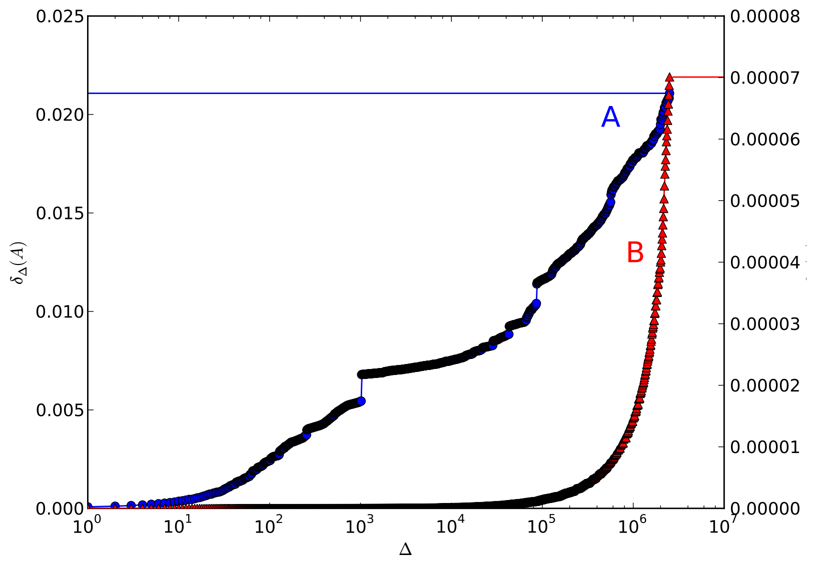


Figure B.2: Δ-density of streams A (blue circles) and B (red triangles) (vertical axis) as a function of Δ (horizontal axis, log scale). The horizontal lines indicate the maximal reachable Δ-density, *i.e.* the density of the induced graphs $G(A)$ and $G(B)$.

increases much faster by the end of the plot, indicating that one must take all the time-span of the stream to see most of its links.

In order to gain more insight on these behaviors, we now study the Δ-density of each single link. We plot the same quantities, namely the value of the Δ-density as a function of Δ, for each link (u, v). Figure B.3 displays two typical examples, one from A and the other from B.

Both plots display a sigmoid shape, indicating that the Δ-density remains very small until a specific value of Δ, and then it rapidly reaches its maximal value 1. Increasing Δ further has no significant impact. This indicates that this specific value plays a key role for this pair of nodes: it is rare to have a longer time interval without an occurence of a link involving them, while it is very frequent for shorter time intervals.

For the example from dataset A, the sharp increase occurs between $\Delta = 10^4 s$ and $\Delta = 10^5 s$. For the example from dataset B, the sharp increase is by the end of the plot only. This indicates that one needs very large values of Δ to be unable to find many intervals of size Δ with no occur-

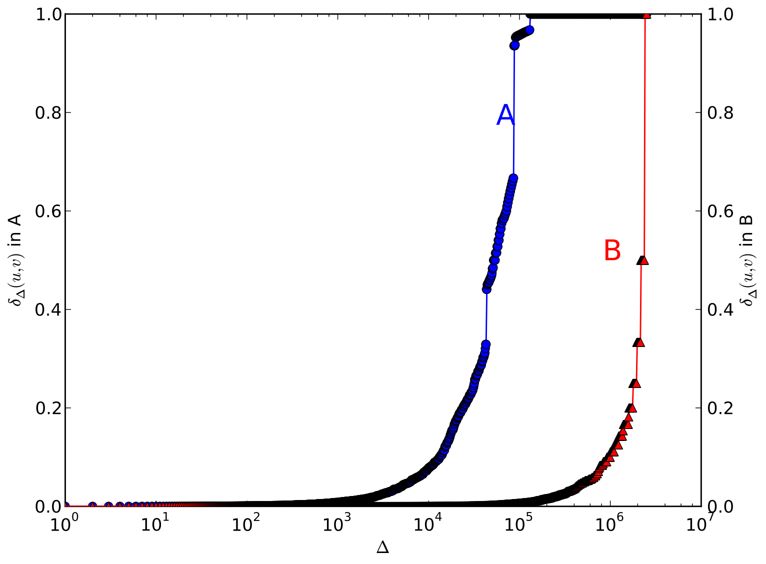


Figure B.3: Δ-density (vertical axis) as a function of Δ (horizontal axis, log scale), for two typical links (one of A and one of B).

rence of the link. In other words, all the occurrences of the link fit in a small time interval, and studying the Δ-density of this pair of nodes has little meaning, if any.

In order to build a more global view of a dataset, we apply the following protocol. For each pair of nodes (u, v), we seek the largest variation in the value of $\delta_\Delta(u, v)$ as a function of Δ (which corresponds to the sharpest increase in the plots of Figure B.3). To ensure that this variation is significant enough, we discard the pairs for which it is lower than 15%. We call the value of Δ at which this largest variation occurs the *characteristic time* of (u, v), and we denote it by $\tau(u, v)$.

We plot in Figure B.4 the distribution of characteristic times we obtain for each dataset.

It appears clearly that a large fraction of the links in A have specific but distinct characteristic times: many have a characteristic time close to $10^3 s$, many around $10^5 s$ and most others between $10^5 s$ and $10^6 s$. This indicates three classes of links (*i.e.* computer communications), which we will discuss in Section B.3.4. Notice however that large characteristic times mean that all occurences of the corre-

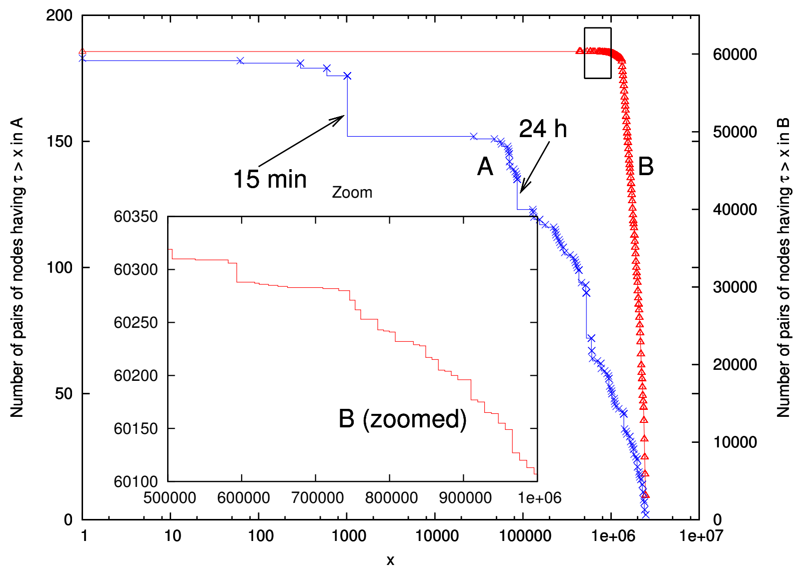


Figure B.4: Inverse cumulative distribution of the characteristic time of all pairs of nodes in our two datasets: for each value x on the horizontal axis, we plot the number y of pairs having characteristic time larger than x.

sponding links appear in a very short period of time. This typically reveals pairs of nodes that exchange packets during a connection that lasts only a few seconds or minutes, but that do not exchange data on a regular basis.

The situation for dataset B is quite different: a huge majority of all characteristic time are close to the maximal possible value, indicating that the occurrences of most links appear in a very short period of time, and do not appear outside this time interval. However, as displayed in the inset of Figure B.4, there is a non negligible number of links with a drastically different behaviour, evidenced by much smaller characteristic times. This shows that some links in the stream have a specific role that distinguishes them from the vast majority of links.

B.3.3 Neighborhoods and clustering coefficient

We focused above on links only. In order to gain insight on more subtle structures, we study here the Δ-density of nodes and their neighbors, and introduce a generalization of the classical notion of clustering coefficient.

Let us first denote by $N(v)$ the neighborhood of any node v, *i.e.* the set nodes to which it is linked. Then the substream $L(\{v\} \times N(v))$ is the stream of all the links between v and its neighbors, while the substream $L(N(v))$ is the stream of links involving two neighbors of v. The Δ-density of these two substreams contains important information about v: $\delta_\Delta(L(\{v\} \times N(v))$ indicates up to what extent the interactions between v and its neighbors occurs at least every Δ time; $\delta_\Delta(L(N(v))$ indicates up to what extent all possible pairs of neighbors of v interact at least every Δ time.

Notice that $\delta_\Delta(L(\{v\} \times N(v))$ captures the Δ-density of v's interactions. We therefore call it the Δ-density of v, and we denote it by $\delta_\Delta(v)$. Likewise, $\delta_\Delta(L(N(v))$ is the Δ-density of the stream induced by the neighbors of v, just like the classical clustering coefficient of a node in a graph is the density of the subgraph induced by its neighbors [Watts and Strogatz, 1998]. For this reason, we call it the Δ-clustering coefficient of v, we denote it by Δ-cc(v).

We now define for each node v its characteristic time $\tau(v)$ in a way similar to previous section: we compute the variations of $\delta_\Delta(v)$ as a function of Δ and select the value of Δ at which this variation is maximal. Figure B.5 presents the distribution of the characteristic times of all nodes.

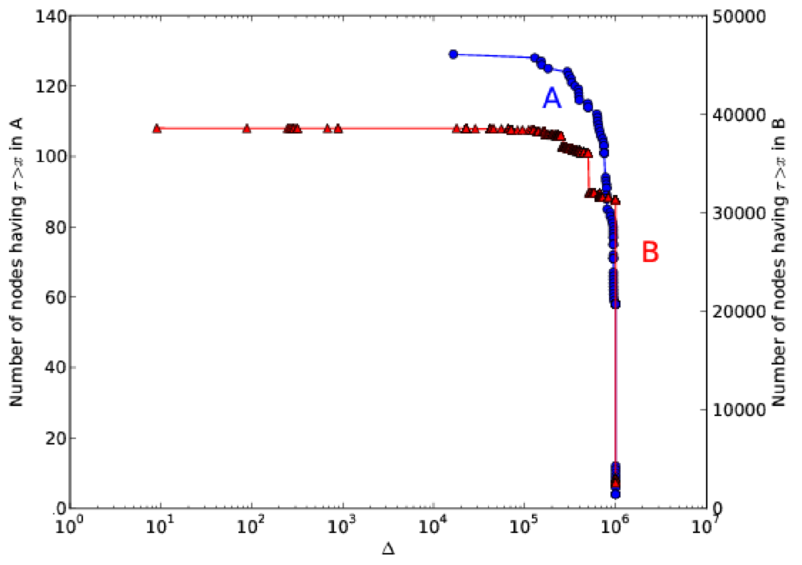


Figure B.5: Inverse cumulative distribution of the characteristic time $\tau(v)$ of each node v of both our datasets: for each value x we plot the number of nodes v such that $\tau(v)$ is larger than x.

For both datasets, we observe a significant number of nodes with non-trivial (*i.e.* much smaller than the whole duration of the trace) Δ-density. This means that these nodes have specific roles in the network, as we will discuss in next section. We also observe that some values of characteristic times are overrepresented, which is revealed by sharp decreases in the plots. This indicates classes of nodes with similar behaviors (at least regarding Δ-density).

When we turn to the computation of Δ-clustering coefficient, we face a problem related to the way our data is collected. Indeed, it consists in traffic managed by firewalls, and so they mostly consist in packets exchanged between an internal network and the rest of the internet. As a consequence, the graph they induce between IP addresses is close to a bipartite graph: nodes are separated into two distinct sets V_1 and V_2 and links exist mostly between nodes in both sets. This implies that there is only very rarely a link between two neighbors of a same node. In our case, this happens for only 33 nodes in dataset A, and this never happens in dataset B.

As the Δ-clustering coefficient of a node is 0 whenever there is no link between its neighbors (like the classical clustering coefficient in graphs), we focus here on the 33 nodes of A for which the clustering coefficient is not 0. We compute for these nodes their τ-clustering coefficient, *i.e.* for each node its Δ-clustering coefficient when the value of Δ is the characteristic time of the node. These values are strongly influenced by the degree of the nodes, and so we plot in Figure B.6 for each node a point indicating its degree and its τ-clustering coefficient.

This plot shows that most considered nodes have a significant τ-clustering coefficient, much larger than 0 even for nodes with large degree. This means that these nodes belong to very structured substreams: many links exist among their neighbors, and that these links are often observed at least once in a time-interval of size τ. An exception is visible on the plot: a node has degree over 100 but a τ-clustering coefficient close to 0, meaning that this node

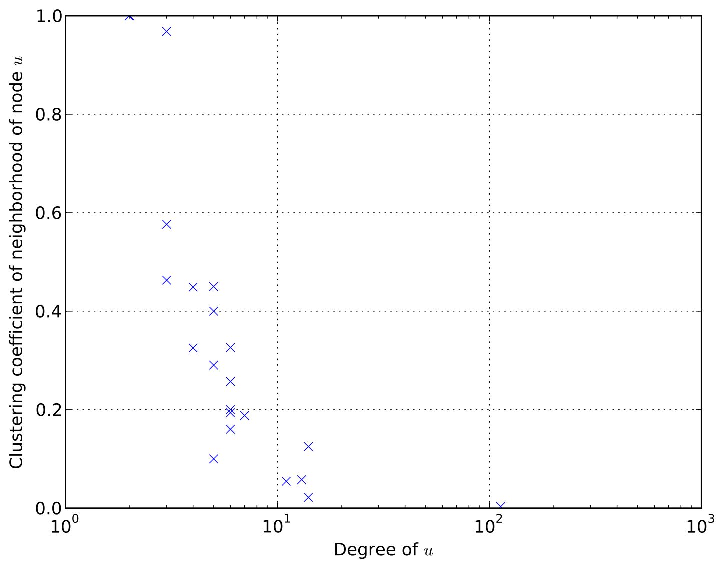


Figure B.6: For each node with non-trivial clustering coefficient, we plot its τ-clustering coefficient (vertical axis) as a function of its degree (horizontal axis).

belongs to a star-like structure (almost none of its neighbors are linked together).

B.3.4 Interpretation

In the previous sections, we have computed and observed several statistics describing the temporal and structural behaviors of nodes and links in our datasets. We now turn to an interpretation of these results in terms of the application area, and in particular regarding the identification of links, nodes, or groups of elements playing specific roles in the network.

We first identified in Section B.3.2 three characteristic times playing a key role in dataset A: around 1000 seconds (approximately 16 minutes), around 90000 seconds (approximately 24 hours), and around 500000 seconds (approximately 5 days). Manual inspection of the data and discussion with network operators revealed the presence of a backup server in the local network, used by external machines, responsible for the $24h$ characteristic times. We also found, without being able to identify their cause, regular communications every 15 minutes from a subset of nodes. Finally, the largest characteristic time is probably due to links appearing only a few times, and is too large compared to the duration of the whole measurement to be significant.

In dataset *B*, many pairs of nodes have a high characteristic value which, as already said, has little significance. However, a few pairs of nodes have a more interesting behaviour, as seen on the inset of Figure B.4. By inspecting the dataset, we could identify from this a few servers with a regular pattern of action: local backup servers and mail servers mostly.

The study of clustering coefficients revealed that some nodes forms groups which are densely connected: most of all possible links among them appear, and do so on a regular basis. This holds for a dozen groups of more than 5 nodes, and even for a few groups of more than 10 nodes. This probably reveals nodes involved in a common task distributed among them, like a complex web service, a distributed computation, or a distributed database.

We also noticed a node with high degree, above 100, but very low clustering coefficient. This means that this machine has many connections, but its neighbors are almost not linked at all: we therefore have a star structure for this machine. This information, added to the fact that this substructure has a characteristic time close to 24 hours, makes it identifiable as a backup server, periodically contacted by the same set of nodes to save their data.

B.4 Conclusion

In this paper, we have introduced the notion of Δ-density, which captures up to what point links appear *all the time* and/or all possible links between considered nodes occur *all the time*. We illustrated the use of this notion on two real-world captures of network traffic, and we have shown that it allows to determine the characteristic times of parts of the traffic in a simple manner. We have shown that many different characteristic times coexist in such traffic, and we used them to distinguish between nodes or set of nodes playing specific roles in the network. This includes for instance backup servers or distributed applications. Such information is useful in two means: to an attacker, who could identify relevant targets, and to network operators, who could optimize services, improve security, etc. It is also a contribution to our understanding of real-world traffic, with applications to improved modeling and simulation.

Our work may be extended in several ways. In particular, we proposed one approach for quantifying the intuition behind Δ-density but variants may also be relevant. For instance, one may slice the stream into pieces of duration Δ and count the fraction of slices containing the considered link. One may also compute the probability that a randomly chosen interval of size Δ contains an occurrence of the link. Although all these definitions are very similar, they have small differences that should be studied.

Our initial goal was to be able to identify distinct characteristic times in a link stream, whereas most studies aggregate information over a given time interval. There is still room for significant progress in this direction. In particular, one may identify several characteristic times for a same substream, by detecting several sharp increases in the Δ-density as a function of Δ instead of only one. This may reflect for instance the fact that users typically have daily, weekly and yearly activity patterns. Going further, a node may have a characteristic time that varies during time, like the characteristic times between two connections during week days and during week-ends, or characteristic times before and after an intrusion. We think that Δ-density may easily be extended to study such phenomena, and this is one of the main directions of our future work.

In the context of IP traffic analysis and in other areas, an important direction also is to extend our definitions to the case of bipartite graphs, in particular the ones regarding clustering coefficient. This may help in capturing more complex phenomena and behaviors, and the notions defined in [Latapy et al., 2008b] could certainly be useful for doing so.

Last but not least, the notions of Δ-density and τ-clustering coefficient defined in this paper are very general, and may be used to study any link stream like email exchanges, financial transactions, and others. In all these cases, questions similar to the ones addressed here arise (in particular the co-existence of different characteristic times that one should distinguish).

Bibliography

[wid, 1984] (1984). Wide project website: http://www.wide.ad.jp/.

[kdd, 1999] (1999). Kdd cup'99 dataset: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[CAI, 2001] (2001). Caida dataset repository: http://www.caida.org/data/overview/.

[ANT, 2002] (2002). Usc ant lab dataset repository: https://ant.isi.edu/datasets/all.html.

[soc, 2008] (2008). Sociopatterns data repository : http://sociopatterns.org.

[Abry et al., 2002a] Abry, P., Baraniuk, R., Flandrin, P., Riedi, R., and Veitch, D. (2002a). Multiscale nature of network traffic. *IEEE Signal Processing Magazine*, 19(3):28–46.

[Abry et al., 2002b] Abry, P., Baraniuk, R., Flandrin, P., Riedi, R., and Veitch, D. (2002b). Multiscale nature of network traffic. *Signal Processing Magazine, IEEE*, 19(3):28–46.

[Abry and Veitch, 1998] Abry, P. and Veitch, D. (1998). Wavelet analysis of long-range-dependent traffic. *IEEE transactions on information theory*, 44(1):2–15.

[Abry et al., 1998] Abry, P., Veitch, D., and Flandrin, P. (1998). Long-range dependence: Revisiting aggregation with wavelets. *Journal of Time Series Analysis*, 19(3):253–266.

[Ankerst et al., 1999] Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). Optics: ordering points to identify the clustering structure. In *ACM Sigmod Record*, volume 28, pages 49–60. ACM.

[Aynaud and Guillaume, 2011] Aynaud, T. and Guillaume, J.-L. (2011). Multi-Step Community Detection and Hierarchical Time Segmentation in Evolving Networks. In *Fifth SNA-KDD Workshop Social Network Mining and Analysis, in conjunction with the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2011)*.

[Bakar et al., 2006] Bakar, Z. A., Mohemad, R., Ahmad, A., and Deris, M. M. (2006). A comparative study for outlier detection techniques in data mining. In *2006 IEEE conference on cybernetics and intelligent systems*, pages 1–6. IEEE.

[Bamnett and Lewis, 1994] Bamnett, V. and Lewis, T. (1994). Outliers in statistical data.

[Barford et al., 2002] Barford, P., Kline, J., Plonka, D., and Ron, A. (2002). A signal analysis of network traffic anomalies. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 71–82. ACM.

[Basu et al., 2010] Basu, P., Bar-Noy, A., Ramanathan, R., and Johnson, M. P. (2010). Modeling and analysis of time-varying graphs. *CoRR*, abs/1012.0260.

[Batagelj and Praprotnik, 2016] Batagelj, V. and Praprotnik, S. (2016). An algebraic approach to temporal network analysis based on temporal quantities. *Social Network Analysis and Mining*, 6(1):1–22.

[Benamara and Magnien, 2010a] Benamara, L. and Magnien, C. (2010a). Estimating properties in dynamic systems: The case of churn in p2p networks. In *INFOCOM IEEE Conference on Computer Communications Workshops, 2010*, pages 1–6. IEEE.

[Benamara and Magnien, 2010b] Benamara, L. and Magnien, C. (2010b). Estimating properties in dynamic systems: The case of churn in p2p networks. In *INFOCOM IEEE Conference on Computer Communications Workshops , 2010*, pages 1–6.

[Berman, 1996] Berman, K. A. (1996). Vulnerability of scheduled networks and a generalization of menger's theorem. *Networks*, 28(3):125–134.

[Bhuyan et al., 2014] Bhuyan, M. H., Bhattacharyya, D. K., and Kalita, J. K. (2014). Towards an unsupervised method for network anomaly detection in large datasets. *Computing and Informatics*, 33(1):1–34.

[Brauckhoff et al., 2012] Brauckhoff, D., Dimitropoulos, X., Wagner, A., and Salamatian, K. (2012). Anomaly extraction in backbone networks using association rules. *IEEE/ACM Transactions on Networking (TON)*, 20(6):1788–1799.

[Braud-Santoni et al., 2016] Braud-Santoni, N., Dubois, S., Kaaouachi, M.-H., and Petit, F. (2016). The next 700 impossibility results in time-varying graphs. *International Journal of Networking and Computing*, 6(1):27–41.

[Broido and Claffy, 2001] Broido, A. and Claffy, K. (2001). Internet topology: connectivity of ip graphs.

[Bron and Kerbosch, 1973] Bron, C. and Kerbosch, J. (1973). Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9).

[Bui-Xuan et al., 2003] Bui-Xuan, B., Ferreira, A., and Jarry, A. (2003). Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285.

[Caceres and Berger-Wolf, 2013] Caceres, R. S. and Berger-Wolf, T. (2013). *Temporal Networks*, chapter Temporal Scale of Dynamic Networks. Springer Link.

[Casteigts et al., 2015] Casteigts, A., Flocchini, P., Godard, E., Santoro, N., and Yamashita, M. (2015). On the expressivity of time-varying graphs. *Theoretical Computer Science*, 590:27–37.

[Casteigts et al., 2012] Casteigts, A., Flocchini, P., Quattrociocchi, W., and Santoro, N. (2012). Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408.

[Cazals and Karande, 2008] Cazals, F. and Karande, C. (2008). A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407(1–3):564 – 568.

[Crovella and Kolaczyk, 2003] Crovella, M. and Kolaczyk, E. (2003). Graph wavelets for spatial traffic analysis. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1848–1857 vol.3.

[Dainotti et al., 2011] Dainotti, A., Pescapé, A., and Kim, H.-c. (2011). Traffic classification through joint distributions of packet-level statistics. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6. IEEE.

[Dewaele et al., 2007] Dewaele, G., Fukuda, K., Borgnat, P., Abry, P., and Cho, K. (2007). Extracting hidden anomalies using sketch and non gaussian multiresolution statistical detection procedures. In *Proceedings of the 2007 workshop on Large scale attack defense*, pages 145–152. ACM.

[Dorat et al., 2007] Dorat, R., Latapy, M., Conein, B., and Auray, N. (2007). Multi-level analysis of an interaction network between individuals in a mailing-list. In *Annales des télécommunications*, volume 62, pages 325–349. Springer.

[Eberle and Holder, 2007] Eberle, W. and Holder, L. (2007). Anomaly detection in data represented as graphs. *Intell. Data Anal.*, 11(6):663–689.

[Eppstein et al., 2013] Eppstein, D., Löffler, M., and Strash, D. (2013). Listing all maximal cliques in large sparse real-world graphs. *Journal of Experimental Algorithmics*, 18:3.1:3.1–3.1:3.21.

[Erdos and Rényi, 1961] Erdos, P. and Rényi, A. (1961). On the evolution of random graphs. *Bull. Inst. Internat. Statist*, 38(4):343–347.

[Fernandes and Owezarski, 2009] Fernandes, G. and Owezarski, P. (2009). Automated classification of network traffic anomalies. In *International Conference on Security and Privacy in Communication Systems*, pages 91–100. Springer.

[Fontugne et al., 2010a] Fontugne, R., Borgnat, P., Abry, P., and Fukuda, K. (2010a). Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proceedings of the 6th International COnference*, page 8. ACM.

[Fontugne et al., 2010b] Fontugne, R., Borgnat, P., Abry, P., and Fukuda, K. (2010b). Uncovering relations between traffic classifiers and anomaly detectors via graph theory. In *International Workshop on Traffic Monitoring and Analysis*, pages 101–114. Springer.

[Fontugne et al., 2010c] Fontugne, R., Borgnat, P., Abry, P., and Fukuda, K. (2010c). Uncovering relations between traffic classifiers and anomaly detectors via graph theory. In Ricciato, F., Mellia, M., and Biersack, E., editors, *Traffic Monitoring and Analysis*, volume 6003 of *Lecture Notes in Computer Science*, pages 101–114. Springer Berlin Heidelberg.

[Fontugne and Fukuda, 2011] Fontugne, R. and Fukuda, K. (2011). A hough-transform-based anomaly detector with an adaptive time interval. *ACM SIGAPP Applied Computing Review*, 11(3):41–51.

[Fournet and Barrat, 2014] Fournet, J. and Barrat, A. (2014). Contact patterns among high school students. *PLoS ONE*, 9:e107878.

[Fraleigh et al., 2003] Fraleigh, C., Moon, S., Lyles, B., Cotton, C., Khan, M., Moll, D., Rockell, R., Seely, T., and Diot, S. (2003). Packet-level traffic measurements from the sprint ip backbone. *Network, IEEE*, 17(6):6–16.

[Gama et al., 2014] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):44.

[Gaumont et al., 2015] Gaumont, N., Viard, T., Fournier-S'niehotta, R., Wang, Q., and Latapy, M. (2015). Analysis of the temporal and structural features of threads in a mailing-list. *arXiv preprint arXiv:1512.05002*.

[Gauvin et al., 2014] Gauvin, L., Panisson, A., and Cattuto, C. (2014). Detecting the community structure and activity patterns of temporal networks: a non-negative tensor factorization approach. *PloS one*, 9(1):e86028.

[Guralnik and Srivastava, 1999a] Guralnik, V. and Srivastava, J. (1999a). Event detection from time series data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 33–42. ACM.

[Guralnik and Srivastava, 1999b] Guralnik, V. and Srivastava, J. (1999b). Event detection from time series data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 33–42, New York, NY, USA. ACM.

[Heymann et al., 2012] Heymann, S., Latapy, M., and Magnien, C. (2012). Outskewer: Using skewness to spot outliers in samples and time series. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pages 527–534. IEEE Computer Society.

[Himmel et al., 2016] Himmel, A.-S., Molter, H., Niedermeier, R., and Sorge, M. (2016). Enumerating maximal cliques in temporal graphs. *arXiv preprint arXiv:1605.03871*.

[Holme, 2015] Holme, P. (2015). Modern temporal network theory: a colloquium. *The European Physical Journal B*, 88(9):1–30.

[Holme, 2016] Holme, P. (2016). Temporal network structures controlling disease spreading. *arXiv preprint arXiv:1605.00915*.

[Holme and Saramäki, 2012] Holme, P. and Saramäki, J. (2012). Temporal networks. *Physics reports*, 519(3):97–125.

[Hulten et al., 2001a] Hulten, G., Spencer, L., and Domingos, P. (2001a). Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106. ACM.

[Hulten et al., 2001b] Hulten, G., Spencer, L., and Domingos, P. (2001b). Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 97–106, New York, NY, USA. ACM.

[Iliofotou et al., 2009a] Iliofotou, M., Faloutsos, M., and Mitzenmacher, M. (2009a). Exploiting dynamicity in graph-based traffic analysis: techniques and applications. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 241–252. ACM.

[Iliofotou et al., 2009b] Iliofotou, M., Faloutsos, M., and Mitzenmacher, M. (2009b). Exploiting dynamicity in graph-based traffic analysis: Techniques and applications. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, pages 241–252, New York, NY, USA. ACM.

[Iliofotou et al., 2007] Iliofotou, M., Pappu, P., Faloutsos, M., Mitzenmacher, M., Singh, S., and Varghese, G. (2007). Network monitoring using traffic dispersion graphs (tdgs). In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 315–320. ACM.

[Johnson et al., 1988] Johnson, D. S., Yannakakis, M., and Papadimitriou, C. H. (1988). On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123.

[Kanda et al., 2010] Kanda, Y., Fukuda, K., and Sugawara, T. (2010). Evaluation of anomaly detection based on sketch and pca. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5. IEEE.

[Keim, 2002] Keim, D. A. (2002). Information visualization and visual data mining. *Visualization and Computer Graphics, IEEE Transactions on*, 8(1):1–8.

[Kempe et al., 2000] Kempe, D., Kleinberg, J., and Kumar, A. (2000). Connectivity and inference problems for temporal networks. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 504–513. ACM.

[Kim and Reddy, 2008] Kim, S. S. and Reddy, A. (2008). Statistical techniques for detecting traffic anomalies through packet header data. *IEEE/ACM Transactions on Networking (TON)*, 16(3):562–575.

[Klemm et al., 2003] Klemm, A., Lindemann, C., and Lohmann, M. (2003). Modeling ip traffic using the batch markovian arrival process. *Performance Evaluation*, 54(2):149 – 173. Modelling Techniques and Tools for Computer Performance Evaluation.

[Koch, 2001] Koch, I. (2001). Enumerating all connected maximal common subgraphs in two graphs. *Theoretical Computer Science*, 250:1–30.

[Kostakos, 2009] Kostakos, V. (2009). Temporal graphs. *Physica A: Statistical Mechanics and its Applications*, 388(6):1007–1023.

[Lakhina et al., 2004] Lakhina, A., Crovella, M., and Diot, C. (2004). Diagnosing network-wide traffic anomalies. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 219–230. ACM.

[Lakhina et al., 2005] Lakhina, A., Crovella, M., and Diot, C. (2005). Mining anomalies using traffic feature distributions. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 217–228. ACM.

[Lamarche-Perrin et al., 2014] Lamarche-Perrin, R., Demazeau, Y., and Vincent, J.-M. (2014). Building optimal macroscopic representations of complex multi-agent systems. In *Transactions on Computational Collective Intelligence XV*, pages 1–27. Springer.

[Lambiotte et al., 2013] Lambiotte, R., Tabourier, L., and Delvenne, J.-C. (2013). Burstiness and spreading on temporal networks. *The European Physical Journal B*, 86(7):1–4.

[Latapy et al., 2013] Latapy, M., Hamzaoui, A., and Magnien, C. (2013). Detecting events in the dynamics of ego-centred measurements of the internet topology. *Journal of Complex Networks*, page cnt014.

[Latapy et al., 2008a] Latapy, M., Magnien, C., and Del Vecchio, N. (2008a). Basic notions for the analysis of large two-mode networks. *Social networks*, 30(1):31–48.

[Latapy et al., 2008b] Latapy, M., Magnien, C., and Vecchio, N. D. (2008b). Basic notions for the analysis of large two-mode networks. *Social Networks*, 30(1):31 – 48.

[Latapy and Viard, 2014] Latapy, M. and Viard, J. (2014). Complex networks and link streams for the empirical analysis of large software. In *International Conference on Applications and Theory of Petri Nets and Concurrency*, pages 40–50. Springer.

[Lee and Maggioni,] Lee, J. D. and Maggioni, M. Multiscale analysis of time series of graphs.

[Lee and Maggioni, 2011] Lee, J. D. and Maggioni, M. (2011). Multiscale analysis of time series of graphs. In *International Conference on Sampling Theory and Applications (SampTA)*.

[Léo et al., 2015] Léo, Y., Crespelle, C., and Fleury, E. (2015). Non-altering time scales for aggregation of dynamic networks into series of graphs. In *11th International Conference on emerging Networking EXperiments and Technologies–CoNEXT 2015*.

[Leroy and Rousseeuw, 1987] Leroy, A. M. and Rousseeuw, P. J. (1987). Robust regression and outlier detection. *Wiley Series in Probability and Mathematical Statistics, New York: Wiley, 1987*, 1.

[Leskovec et al., 2005] Leskovec, J., Kleinberg, J., and Faloutsos, C. (2005). Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187. ACM.

[Li et al., 2006] Li, X., Bian, F., Crovella, M., Diot, C., Govindan, R., Iannaccone, G., and Lakhina, A. (2006). Detection and identification of network anomalies using sketch subspaces. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 147–152. ACM.

[Marnerides et al., 2014] Marnerides, A. K., Schaeffer-Filho, A., and Mauthe, A. (2014). Traffic anomaly diagnosis in internet backbone networks: a survey. *Computer Networks*, 73:224–243.

[Masuda et al., 2013] Masuda, N., Klemm, K., and Eguíluz, V. M. (2013). Temporal networks: slowing down diffusion by long lasting interactions. *Physical review letters*, 111(18):188701.

[Mazel et al., 2015] Mazel, J., Casas, P., Fontugne, R., Fukuda, K., and Owezarski, P. (2015). Hunting attacks in the dark: clustering and correlation analysis for unsupervised anomaly detection. *International Journal of Network Management*, 25(5):283–305.

[Mazel et al., 2014] Mazel, J., Fontugne, R., and Fukuda, K. (2014). A taxonomy of anomalies in backbone network traffic. In *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 30–36. IEEE.

[Newman, 2004] Newman, M. E. (2004). Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133.

[Nguyen et al., 2012] Nguyen, T. T., Armitage, G., Branch, P., and Zander, S. (2012). Timely and continuous machine-learning-based classification for interactive ip traffic. *IEEE/ACM Transactions on Networking (TON)*, 20(6):1880–1894.

[Nicosia et al., 2013] Nicosia, V., Tang, J., Mascolo, C., Musolesi, M., Russo, G., and Latora, V. (2013). Graph metrics for temporal networks. In *Temporal Networks*, pages 15–40. Springer.

[Nychis et al., 2008] Nychis, G., Sekar, V., Andersen, D. G., Kim, H., and Zhang, H. (2008). An empirical evaluation of entropy-based traffic anomaly detection. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 151–156. ACM.

[Pan and Saramäki, 2011] Pan, R. K. and Saramäki, J. (2011). Path lengths, correlations, and centrality in temporal networks. *Physical Review E*, 84(1):016105.

[Papadopoulos and Voglis, 2005] Papadopoulos, C. and Voglis, C. (2005). Drawing graphs using modular decomposition. In *Graph Drawing*, pages 343–354. Springer.

[Qadeer et al., 2010] Qadeer, M., Zahid, M., Iqbal, A., and Siddiqui, M. (2010). Network traffic analysis and intrusion detection using packet sniffer. In *Communication Software and Networks, 2010. ICCSN '10. Second International Conference on*, pages 313–317.

[Queyroi, 2014] Queyroi, F. (2014). Delta-duplication of time-varying graphs. In *Conférence MARAMI 2014 (Modèles et Analyses Réseau: Approches Mathématiques et Informatiques)*.

[Redmond and Cunningham, 2016] Redmond, U. and Cunningham, P. (2016). Subgraph isomorphism in temporal networks. *arXiv preprint arXiv:1605.02174*.

[Rowe et al., 2007] Rowe, R., Creamer, G., Hershkop, S., and Stolfo, S. J. (2007). Automated social hierarchy detection through email network analysis. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis*, WebKDD/SNA-KDD '07, pages 109–117, New York, NY, USA. ACM.

[Rubinstein et al., 2009] Rubinstein, B. I., Nelson, B., Huang, L., Joseph, A. D., Lau, S.-h., Rao, S., Taft, N., and Tygar, J. (2009). Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 1–14. ACM.

[Samudrala and Moult, 1998] Samudrala, R. and Moult, J. (1998). A graph-theoretic algorithm for comparative modeling of protein structure. *Journal of Molecular Biology*, 279(1):287 – 302.

[Santoro et al., 2011] Santoro, N., Quattrociocchi, W., Flocchini, P., Casteigts, A., and Amblard, F. (2011). Time-varying graphs and social network analysis: Temporal indicators and metrics. *arXiv preprint arXiv:1102.0629*.

[Saramäki and Holme, 2015] Saramäki, J. and Holme, P. (2015). Exploring temporal networks with greedy walks. *The European Physical Journal B*, 88(12):1–8.

[Silveira et al., 2010] Silveira, F., Diot, C., Taft, N., and Govindan, R. (2010). Astute: Detecting a different class of traffic anomalies. *ACM SIGCOMM Computer Communication Review*, 40(4):267–278.

[Sowe et al., 2006] Sowe, S., Stamelos, I., and Angelis, L. (2006). Identifying knowledge brokers that yield software engineering knowledge in {OSS} projects. *Information and Software Technology*, 48(11):1025 – 1033.

[SPI, 2015] SPI (2015). Debian mailing-list archive: https://lists.debian.org/debian-user/.

[Starnini et al., 2012] Starnini, M., Baronchelli, A., Barrat, A., and Pastor-Satorras, R. (2012). Random walks on temporal networks. *Physical Review E*, 85(5):056115.

[Stolfo et al., 2000] Stolfo, S. J., Fan, W., Lee, W., Prodromidis, A., and Chan, P. K. (2000). Cost-based modeling for fraud and intrusion detection: Results from the jam project. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, volume 2, pages 130–144. IEEE.

[Strogatz, 2001] Strogatz, S. H. (2001). Exploring complex networks. *Nature*, 410(6825):268–276.

[Sueur et al., 2011] Sueur, C., Jacobs, A., Amblard, F., Petit, O., and King, A. J. (2011). How can social network analysis improve the study of primate behavior? *American Journal of Primatology*, 73(8):703–719.

[Sulo et al., 2010] Sulo, R., Berger-Wolf, T., and Grossman, R. (2010). Meaningful selection of temporal resolution for dynamic networks. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, pages 127–136. ACM.

[Sun et al., 2007] Sun, J., Faloutsos, C., Papadimitriou, S., and Yu, P. S. (2007). Graphscope: Parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 687–696, New York, NY, USA. ACM.

[Takaguchi et al., 2012] Takaguchi, T., Sato, N., Yano, K., and Masuda, N. (2012). Importance of individual events in temporal networks. *New Journal of Physics*, 14(9):093003.

[Tang et al., 2010] Tang, J., Scellato, S., Musolesi, M., Mascolo, C., and Latora, V. (2010). Small-world behavior in time-varying graphs. *Physical Review E*, 81(5):055101.

[Tedder et al., 2008] Tedder, M., Corneil, D., Habib, M., and Paul, C. (2008). Simpler linear-time modular decomposition via recursive factorizing permutations. In *Automata, Languages and Programming*, pages 634–645. Springer.

[Tomita et al., 2006] Tomita, E., Tanaka, A., and Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363:28–42.

[Valdano et al., 2015] Valdano, E., Ferreri, L., Poletto, C., and Colizza, V. (2015). Analytical computation of the epidemic threshold on temporal networks. *Physical Review X*, 5(2):021005.

[Vestergaard et al., 2014] Vestergaard, C. L., Génois, M., and Barrat, A. (2014). How memory generates heterogeneous dynamics in temporal networks. *Physical Review E*, 90(4):042805.

[Vestergaard et al., 2016] Vestergaard, C. L., Valdano, E., Génois, M., Poletto, C., Colizza, V., and Barrat, A. (2016). Impact of spatially constrained sampling of temporal contact networks on the evaluation of the epidemic risk. *European Journal of Applied Mathematics*, pages 1–17.

[Viard and Latapy, 2014a] Viard, T. and Latapy, M. (2014a). Identifying roles in an ip network with temporal and structural density. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 801–806. IEEE.

[Viard and Latapy, 2014b] Viard, T. and Latapy, M. (2014b). Identifying roles in an IP network with temporal and structural density. In *Computer Communications Workshops (INFOCOM WKSHPS)*, pages 801–806.

[Viard and Latapy, 2014c] Viard, T. and Latapy, M. (2014c). Source code in python for computing cliques in link streams: `https://github.com/TiphaineV/delta-cliques`.

[Viard et al., 2015a] Viard, T., Latapy, M., and Magnien, C. (2015a). Calcul de cliques maximales dans les flots de liens. In *ALGOTEL 2015—17èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*.

[Viard et al., 2015b] Viard, T., Latapy, M., and Magnien, C. (2015b). Revealing contact patterns among high-school students using maximal cliques in link streams. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 1517–1522. ACM.

[Viard et al., 2016] Viard, T., Latapy, M., and Magnien, C. (2016). Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252.

[Viger and Latapy, 2005] Viger, F. and Latapy, M. (2005). Efficient and simple generation of random simple connected graphs with prescribed degree sequence. *Computing and Combinatorics*, pages 440–449.

[Wang, 2014] Wang, Q. (2014). Link prediction and threads in email networks. In *Data Science and Advanced Analytics (DSAA), 2014 International Conference on*, pages 470–476. IEEE.

[Watts and Strogatz, 1998] Watts, D. and Strogatz, S. (1998). Collective dynamics of 'small-world' networks. *Nature*, (393):440–442.

[Wehmuth et al., 2015a] Wehmuth, K., Fleury, É., and Ziviani, A. (2015a). Multiaspect graphs: Algebraic representation and algorithms. *arXiv preprint arXiv:1504.07893*.

[Wehmuth et al., 2015b] Wehmuth, K., Ziviani, A., and Fleury, E. (2015b). A unifying model for representing time-varying graphs. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–10. IEEE.

[Xu et al., 2014] Xu, K., Wang, F., and Gu, L. (2014). Behavior analysis of internet traffic via bipartite graphs and one-mode projections. *IEEE/ACM Transactions on Networking*, 22(3):931–942.

[Xu et al., 2005] Xu, K., Zhang, Z.-L., and Bhattacharyya, S. (2005). Profiling internet backbone traffic: behavior models and applications. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 169–180. ACM.

[Zander et al., 2005] Zander, S., Nguyen, T., and Armitage, G. (2005). Self-learning ip traffic classification based on statistical flow characteristics. In *International Workshop on Passive and Active Network Measurement*, pages 325–328. Springer.

[Zhou et al., 2009] Zhou, Y., Hu, G., and He, W. (2009). Using graph to detect network traffic anomaly. In *Communications, Circuits and Systems, 2009. ICCCAS 2009. International Conference on*, pages 341–345.