Luca Baldesi

Distributed live streaming on mesh networks

PhD Thesis

December 2017

University of Trento
Department of Information Engineering and Computer Science
Via Sommarive 9, I-38123, Povo (TN)

PhD program Cycle 30

PhD Advisors Prof. Renato Lo Cigno
PhD. Leonardo Maccari

Abstract

Internet is evolving in both its structure and usage patterns; this work addresses two trends: i) the increasing popularity and the related generated traffic of media streaming applications and ii) the emerging of network portions following different philosophies from the rest of the internet and being characterized by a mesh topology, such as Community Networks. This thesis presents a modeling for decentralized live streaming for mesh networks based on graph theory, considering the different inter-dependent network abstractions involved. It proposes optimization strategies based on popular centrality metrics, such as betweenness and PageRank. Results on real-world datasets validate the theoretical work and the derived optimizing strategies are implemented in open-source streaming platforms.

Contents

Published articles

This thesis is based on research that lead to the publication of the following papers. For each paper it is reported the chapter it refers to, a short synopsis and my role and contribution in the research.

- L. Baldesi, L. Maccari, and R. Lo Cigno, "Live P2P streaming in CommunityLab: Experience and insights," in *13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, June 2014, pp. 23–30 [Link].

  Synopsis. We analize the feasibility of porting PeerStreamer on a Community Network (CN). My contribution lies on working with a CN testbed, collecting and analysing the data and it is described in Chapter 5.

- L. Baldesi, L. Maccari, and R. Lo Cigno, "Improving P2P streaming in community-lab through local strategies," in *10th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2014, pp. 33–39 [Link].

  Synopsis. We evaluate different streaming parameter settings for a CN and we propose a simple local heuristic. My contribution focuses on testing our streaming platform against several suitable configuration as well as proposing and implementing a novel heuristic technique. A detailed description can be found in Chapter 5.

- L. Baldesi, L. Maccari, and R. Lo Cigno, "Improving P2P streaming in Wireless Community Networks," *Computer Networks*, vol. 93, no. Part 2, pp. 389 – 403, 2015 [Link].

  Synopsis. We extend the aforementioned works in evaluating heuristics suitable for CN and we propose a framework for mapping logical links to physical ones improving the performance. My contribution deals with ideating the cross-layer

framework, developing a topology building strategy and driving the experiments. A comprehensive description on this topics can be found in Chapters 5 and 6.

- L. Maccari, L. Baldesi, R. Lo Cigno, J. Forconi, and A. Caiazza, "Live Video Streaming for Community Networks, Experimenting with PeerStreamer on the Ninux Community," in *ACM Workshop on Do-it-yourself Networking: An Interdisciplinary Approach*, ser. (ACM Co-located with MobySis), 2015, pp. 1–6 [Link].

  Synopsis. We envision the use of a community video streaming service for CNs. My contribution is related to the technical detailing part and it is described in Chapter 8.

- L. Baldesi and L. Maccari, "NePA TesT: network protocol and application testing toolchain for community networks," in *12th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, Jan 2016, pp. 1–8 [Link].

  Synopsis. We present the Network Protocol and Application Testing Toolchain for emulations focused on CNs. My contribution deals with the main design and development of the platform whose description is detailed in Chapter 4.

- L. Baldesi, L. Maccari, and R. Lo Cigno, "Optimized cooperative streaming in wireless mesh networks," in *IFIP Networking Conference (IFIP Networking) and Workshops*, May 2016, pp. 350–358 [Link].

  Synopsis. We formalize our cross-layer optimization approach in terms of graph theory and we highlight its binding to betweeness centrality. We further validate our optimizing approaches through our emulation platform. My contribution relates to the theoretical modeling the cross-layer intersection graph and the implementation code. Such work is reported in Chapter 6.

- L. Maccari, N. Facchi, L. Baldesi, and R. Lo Cigno, "Optimized P2P streaming for wireless distributed networks," *Pervasive and Mobile Computing*, 2017 [Link].

  Synopsis. We extend the aforementioned publication considering a simple yet enhancing pruning heuristics. My contribution relies on the fundamentals of the theory. The extension is described in Chapter 6.

- L. Baldesi, L. Maccari, and R. Lo Cigno, "On the Use of Eigenvector Centrality for Cooperative Streaming," *IEEE Communications Letters*, vol. 21, no. 9, pp. 1953–1956, Sept 2017 [Link].

Synopsis. We propose and validate an optimization strategy for boosting the video distribution on a mesh network. My contribution consists in modeling the problem according to the graph theory, proving the theorem on boosting the distribution and implementing it on a state of the art simulator. This contribution description can be found in Chapter 7.

- L. Baldesi, A. Markopoulou, and C. Butts, "Spectral Graph Forge: Graph Generation Targeting Modularity," in *IEEE International Conference on Computer Communications - INFOCOM*, 2018, Accepted, to appear.

  Synopsis. We envision the possibility to create synthetic graphs with prescribed global properties. My contribution is in developing the theoretical part as well as implementing the graph generating platform. Both those topics are covered in Chapter 9.

Research/study activities

During my PhD program I had been working with national and international collaborators.

Period abroad. I spent nine months (10/3/2016-19/12/2016) visiting the department of Electrical Engineering and Computer Science at the University of California, Irvine, under the supervision of Athina Markopoulou. During such period I focused on synthetic graph global properties and our findings are published [9] and presented in Chapter 9.

Research projects. I collaborated to the following research projects:

- *Open Source P2P Streaming* (sub-project of the European FP7 CONFINE project); I contributed on evaluating and tailoring existing P2P distribution systems in the environment of Wireless Community Networks.
- *Horizon 2020 - netCommons*; I designed and realized the PeerStreamer-ng platform (presented in Chapter 8), implementing the peer-to-peer (P2P) technologies discussed in this thesis.

Teaching. During my PhD period I served as teaching assistant for the following courses at the University of Trento:

- Simulation and Performance Evaluation; course at the Department of Information Engineering and Computer Science.
- Privacy, Trust and Security; part of the SCoDeM master at the Department of Information Engineering and Computer Science.

I also co-advised two bachelor thesis:

- L. Ghiro, "Logic topology adaptation strategy comparison for P2P video distribution on wireless community netwroks," 2014, University of Trento, Italy

- R. Francescato, “Feasibility study of an audio conferencing system based on PeerStreamer,” 2015, University of Trento, Italy

Chapter 1

Introduction

Media streaming is a killer application for every-day users; its traffic consitutes the most important portion of the overall internet traffic. Video streaming has been estimated to generate 73% of the total consumer internet traffic in 2016 and it is expected to reach 82% of bandwidth usage by 2021 [12]. Typically, streaming is provided through centralized, cloud-based services such as Youtube or Vimeo.

Media streaming has a plethora of applications and it can be used in several different contexts. In this work, the scope is specifically narrowed on live streaming on mesh networks.

Live streaming, has already mentioned, is generally orchestrated using centralized approaches but it is characterized by:

- being of interest for a limited (short) amount of time,
- conveying content within a given maximum delay (generally in the order of seconds).

Examples of live streaming include: IP-TV, VoIP applications, internet radios and video conferences.

In considering the dichotomy of the centralized versus decentralized approaches, the reasons behind the success of the centralized approaches are various, most of them non-technical, but derived from business logic. For example, centralized solutions do not scale well with a growing number of consumers but they perfectly suite the purpose of control over data and user accountability. The business model behind this kind of distribution grants enough incomes that providers need for the periodic upgrade of a communication backbone able to cope with the scalability issue. On the other side,

decentralized solutions have faced several difficulties from the business model point of view, generally lacking a clear accountable model, Internet Service Providers (ISPs) have considered P2P systems more a threat than a resource and decentralized systems had been hindered in optimizing their distribution. In fact, ISPs are generally reluctant to disclose network details so efforts like ALTO [13] and P4P [14] cannot in practice provide decentralized applications with enough information for optimization. This scenario has recently slowly started to change both from reconsidering the distributed system economic potentials [15, 16] and from the emerging of new type of internet infrastructures with very different (or absent) business logic, namely mesh networks.

Mesh networks have been extensively studied and a large body of literature reports their characteristics and performance [17–19]. Their growing success is mainly due to two reasons:

- the continuous improvement of the 802.11 standard,
- the continuously decreasing price of powerful wireless devices.

Indeed, it is reasonably easy to establish bi-directional point-to-point and reliable 802.11ac links with a bandwidth up to $300\,\mathrm{Mbit\,s^{-1}}$, spanning over kilometres for the price of less than 150$. The evolution of mesh networks helps the spreading of new network paradigms, like CNs. Community Networks follow fully-open and promising principles [20]; they are internet portions organized as mesh networks generally providing ISP services, but they are almost free from business logics and, as opposed to ISPs, they are not reluctant to disclose their network topology structures (see Section 2.1.1). Moreover, their substantial lack of backbones makes centralized approaches less suitable for deployment while their high bandwidth symmetric links provide the best scenario for P2P approaches. CN bottom-up organization model enables them to potentially grow on exponential basis, hence making them an increasingly important communication asset inter connecting tens of thousands of users worldwide [21]. Industry has noticed their potential and fostered their integration with 5G cellular netwoks [22].

Distributed live streaming on mesh networks is the focus of this thesis, targeting in particular CNs with P2P approaches. This topic presents several challenges to be addressed; as CNs are created with a bottom-up approach, often on volunteer basis and depending on the specific physical impairments of the geographic surrounding area (buildings, mountains, etc.) they can hardly be planned when they scale to hundreds or thousands of nodes. The live streaming goal of a *timely and effective*

content distribution (Section 3.2.2) on CN depends hence on the cooperation of nodes and their ability to orchestrate the unplanned available resources.

The main goals of this work are the following:

i) drawing attention on the need of reshaping killer services for the CN context,

ii) creating a partnership with CN initiatives for feedbacks and early solution deployments,

iii) creating a mathematical and algorithmic framework for dealing with P2P streaming on CNs,

iv) creating a suitable tool for distributed live streaming tailored for CNs.

Goals i-ii) are non-technical and meant to create a favorable environment for the deployable solutions, resulting from the goals iii-iv).

Point i) is tackled by analysing real-world CNs with performance measurement for an off-the-shelf P2P streaming platform, called PeerStreamer (described in Section 2.1.2) and by giving insights on the different structures and needs a CN can have with respect the rest of the internet (results are shown in Section 4.1.1).

To address point ii), this work as been supported since the early beginnings by European projects focused on CNs, namely Open Source P2P Streaming (OSPS)[1] and netCommons[2] which provided important CN insights and feedbacks and lastly allowed the thesis outcome tool, PeerStreamer-ng, to be included among the default services in the CN operating system *cloudy*[3].

The proposed model for P2P systems on mesh networks (detailed in Chapter 3) satisfies point iii), seeking to be the more generic and expressive as possible without neglecting any distribution detail and the derived solutions (described in Chapters 6 and 7) are tested against simulations and emulations of real-world networks as well as a plethora of synthesized realistic dataset granting a sufficient level of generalization for assessing the solutions robustness (experiment frameworks are presented in Sections 4.2 and 4.3).

Point iv) has been addressed by implementing a platform based on the aforementioned strategies (Chapter 8).

[1] http://osps.disi.unitn.it/
[2] https://netcommons.eu/
[3] http://cloudy.community/

1.1 Guidance for the reader

This thesis was thought to be read from the beginning to the end but readers can readily jump to the topics of interest; in particular

- the reader interested in **theoretical modelling and the algorithmic solutions** derived for P2P streaming on CNs can first have a look at Chapter 3 for the nomenclature and the background recalls and then skip to Chapters 5 to 7,
- the reader interested in evaluation tools for deriving streaming strategies (**testbeds, emulators and simulators**) can readily check Chapter 4,
- the reader interested in **synthetic graph generations** preserving global properties can skip to Chapter 9,
- the reader interested in **PeerStreamer-ng**, the application outcome of this thesis for live streaming on CNs can jump to Chapter 8.

Chapter 2

State of Art

This work tackles challenges specific to P2P over CNs, which have been little covered in literature; however, it has moved from previous works and it has been compared against state of the art solutions.

Section 2.1 presents the technologies and strategies at the basis of this work approach; while developing new solutions, especially focusing on overlay and distribution optimization, the state of art introduced in Section 2.2 has been taken into consideration. Section 2.3 highlights this thesis contribution to the state of the art.

2.1 Background

Mesh networks are a well-established research field and CNs are a notable example from such context. PeerStreamer is the P2P live streaming platform used to implement the proposed solutions.

2.1.1 Community Networks

CNs are a notable example of mesh networks and usually they are mainly made of wireless links, and, for this reason, the terms CN and Wireless Community Network (WCN) are used interchangeably. As CNs are growing larger and larger, they become important communication assets and ISPs serving thousand of users by interconnecting smaller mesh networks [21]. However, they do not match the typical ISP network structure as they typically lack of backbones, powerful data centers and cloud systems. Also they generally lack of a high capacity interconnection with the rest of the internet

making the design of popular services (streaming included) dedicated to CN very appealing for users. The demand for such advanced dedicated services has driven the deployment of many solutions; for example, in Guifi[1], one of the largest WCN, there are fourteen VoIP servers, five video conference systems and nine radio broadcasting stations [4].

WCN links are characterized by:

- high bandwidth ($\sim 10\,\mathrm{Mbit\,s^{-1}}$),
- symmetric bandwidth,
- low delay (few milliseconds).

Hence, generally the packet commuting time at nodes is dominant with respect to transmission delay. Figure 2.1 gives a representation of a mesh network interconnecting laptops and servers.

Networking. WCNs are multi-hop mesh networks so they need routing algorithms to make traffic packets reach their destinations. A good routing algorithm for WCNs should be: decentralized, self-organizable and self-healing [19]. Decentralized as routing

[1] http://guifi.net

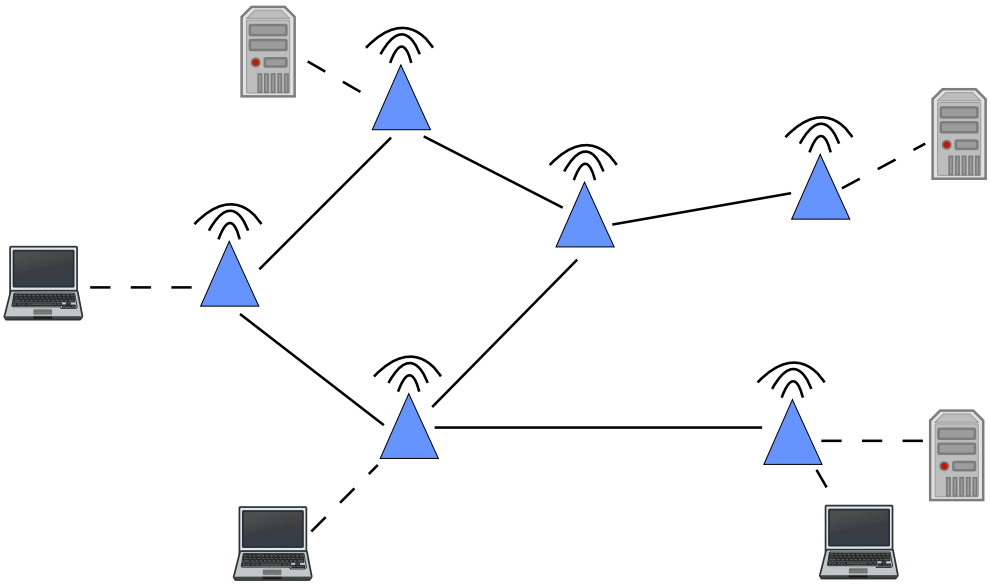


Figure 2.1 – Mesh network representation (blue triangles) interconnecting laptop and servers.

decisions are made by the single nodes according to an agreed routing protocol, self-organizable as such protocol and individual routing decisions determine the overall network routing, irrespective of the network complexity, and self-healing as capable of recovering from node and routing failures.

A routing algorithm performing the way just described is non-trivial as basic routing algorithms in WCNs can lead to sever issues, e.g., large area of packet flooding, node disconnections, large power consumption and unfairness of network resource usage [19]. Therefore, several routing protocols for the WCN context have been proposed. Among the others, two popular protocols are Optimized Link State Routing Protocol (OLSR) [23] and Better Approach To Mobile Adhoc Networking (B.A.T.M.A.N.) [24]. These protocols work by exchanging network routing information among the network nodes, collaborating to forward the network traffic. The routing information exchanged is readily accessible in the network nodes and it can be used to optimize upper-layers communication (see Chapter 6).

This thesis focuses in particular on OLSR as each node of a network running such protocol is able to access the whole network topology.

Philosophy. WCNs philosophy is very peculiar, comprising of a view of networking bound to node/user collaboration and mutual aid which perfectly matches the P2P approach. Such view completely differs from the usual ISP ones and it lacks the reasons to hinder P2P approaches mentioned in Chapter 1. In fact, WCNs are bottom-up mesh networks *built by people for people* [21]. For the CN philosophy, each network node has to be owned by its final user and all the users collaborate to maintain the network by sharing their knowledge and expertise. While the former aspect focuses on the single users as main operators of the network growth and maintenance, the latter requires the constitution of a user community, meaning a relationship network letting users helping each other to overcome their difficulties.

Another pillar of CNs is network neutrality; when entering in the network each user must permit the other traffic through its node without any kind of filtering or shaping. Usually these requirements are stated in documents called *micro-peer agreement* users are expected to accept before joining the CNs.

Mutual-aid principle, resource sharing and network neutrality concepts make CNs a perfect match for P2P approaches.

2.1.2 PeerStreamer

PeerStreamer is the outcome and follow-up of the NAPAWINE project and can be considered today the most advanced open and documented platform to build P2P video streaming services. It is based on a non-structured, highly dynamic, mesh overlay topology that supports the video distribution process through swarming of elementary information units called *chunks*. The participating peers complete the distribution selecting chunks and peers for the information exchange following one of many strategies selectable in PeerStreamer when building the actual *incarnation* of the streamer that one wants to experiment with.

PeerStreamer is composed of several logical blocks: an I/O module to capture video and reproduce it; a topology management subsystem to build and manage the overlay; a chunk and peer selection (within the overlay) to choose what chunk is to be exchanged with what peer in the overlay. A high level description of the logical architecture of the system has been published by Birke et al. [25], while Abeni et al. [26] report the software architecture of the core library, which guarantees high portability and efficiency of the system.

The chunk exchange protocol is based on a Push/Pull approach with Offer/Select and Confirmation [27, 28], which ensure that no duplicated chunks are received by any peer and that network resources are well exploited without building long transmission queues that would increase the chunk delivery and consequently also the playout delay.

A complete overview of P2P live streaming and the PeerStreamer related peculiarities are given in Section 3.2.

2.2 Related Works

The optimization strategies derived in this work focus mainly on applying centrality metrics concepts on two distinct P2P challenges; i) optimal overlay rewiring, and ii) optimal content distribution on a given overlay. Both such optimization goals are typical of P2P streaming systems but centrality metrics have been little applied in this context.

2.2.1 Centrality metrics

Centrality metrics play a central role in both overlay and distribution optimization but, until recently, this has been largely ignored in technical works. Centrality metrics have been used since the 70s in sociology as a way to identify the most influential

nodes or groups in social networks. Surprisingly, they were not applied to multi-hop networks till recent times [29]. Among the other topics, those metrics have been successfully applied in network monitoring and routing [30], intrusion detection and firewalling [31, 32] and topology control [33].

Centrality metrics have also been applied in Pocket Switched Networks (PSNs) [34, 35], where they are central for the optimization of information spreading by leveraging the importance of some particular nodes. The centrality concepts have also been used to improve caching performance [36].

Betweenness and PageRank [37] centralities, which are employed in this work, are introduced and defined in Section 3.1.3.

2.2.2 Overlay optimization

CN networking requires a knowledge of the underlying infrastructure, routes and path weights. Such information is hence collected and utilized by CN nodes and it comes already available for applications willing to optimize logical rewiring. The solutions proposed in this work take advantage from this information and from resulting centrality metrics to optimize the P2P overlay with respect the network resource usage and traffic bottlenecks realizing what is commonly referred to as *cross-layer optimization*.

Several works propose cross-layer optimization in multi-hop wireless mesh networks [38–40], but they require a tweaking of the wireless medium characteristics for a better rate allocation and they do not take advantage from network layer information. In the same direction, Guan et al [41] presented a method for jointly optimize the wireless transmission power and some content streaming rate.

The Software Defined Network (SDN) approach has been also applied to P2P streaming applications [42, 43], orchestrating a network including *super peers* in close collaboration with ISPs, hence, drifting from a pure P2P model.

The use of special peers, called super peers or *helper nodes*, built and running for the purpose of helping the streaming distribution, has been used in the cloud-assisted streaming context [44].

Kuo et al. [45] report a large list of works on optimization of decentralized systems over multi-hop mesh networks; however they do not focus on the *liveliness* of a time-constrained streaming, with the notable exception of the work by Setton et al. [46] although it requires a deep integration with the lower levels of the network stack. The solution proposed by Kuo et al. [45] is even more intrusive, requiring the *extended router header* of IPv6 and the modification of all the network layer implementations.

Among the techniques for overlay optimization, the approach of *network coordinates* is based on node distance evaluation and several algorithms have been proposed [47–49] to work in the heterogeneous environment of the internet. Being tailored for the internet, those methods do not take access of the underlay information mesh networks and, particular, WCNs offer.

A second branch of overlay optimization has focused on tailoring logical links with respect to bandwidth [50] or delay [27] (generally Round Trip Time (RTT)) measures but also on a mix of those [51]. Again, those solutions were tailed for the internet but WCNs are quite different so that delays are quite lower and the links take advantage from bandwidth symmetry, hence, each node can usefully contribute to some content dissemination.

Marco Conti et al. propose a cross-layer overlay rewiring relaying on an extension of the mesh network proactive routing control [52]. Their approach, however, is highly tied to this kind of protocols and it requires the modification of their implementation.

2.2.3 Distribution optimization

The cooperation among the nodes of a network is essential for P2P live streaming. Live streaming tolerates some packet loss but the timely delivery requirement puts a strong focus on the packet delay distribution among the nodes [53–55].

Distribution in a P2P network happens mainly through *scheduling*. Distribution scheduling is already well-investigated topic whose main references are reported in Section 3.2.1.3. Among the others, the work by Zhang et al. [54] extends the most used P2P distribution scheduling algorithms.

As this thesis focus on distribution optimization instead, in the following the main related works on distribution scheduling optimization are reported. Few works are dedicated to optimize the distribution parameters for a given P2P overlay while a large body of work is dedicated to embedding multicast trees on a given mesh topology (either physical or logical, P2P) and it is summarized by Mokhtarian and Jacobsen [56] who also model and optimize the live distribution over mesh networks by using concurrent multicast tree distribution. Optimal rate allocation for content distribution on mesh networks has been already investigated [57, 58]; the most recent work, by Wu and Li [59], proposes a *fluid flow*-based model for node multicast and it optimizes with respect to each node distribution sending rate. Their strategy can be implemented in a decentralized way but it introduces a lot of node overhead in terms of both signaling traffic and computational resources.

2.3 This work contribution

Summing up, this work presents two main contributions to the state of art:

- A new overlay optimization scheme based on betweenness centrality;
- A new distribution optimization scheme based on PageRank centrality.

The overlay optimization strategy, presented in Chapter 6, is designed to be: not intrusive, meaning it does not require any existing layer implementation modification; completely decentralized, without helper or super nodes, ISP independent; and exploiting the advanced mesh network routing resources.

The distribution optimization strategy, presented in Chapter 7, focuses on tweaking the distribution parameters, hence being applicable to a broad range of scenarios, and being decentralized and lightweight. The Wu and Li strategy is considered as a baseline for this context and it is more deeply presented in Section 7.4.1.

Chapter 3

Theoretical modelling

The distribution optimization goal of this thesis requires an adequate modeling of P2P live streaming on CNs. This chapter presents a formalization of such context in terms of graph theory. To this end, recalls on the notions from the state of art are briefly reported (Section 3.1) and, next, the proposed formalization of a P2P distribution system is given (Section 3.2). For a complete overview of theoretical graph foundations see, for example, the book by Harary [60]; for network specific aspects see the book by Newman [61]. The following description adopts the convention of upper case letters for set and matrice symbols and lower case letters for set elements and vector symbols. P2P peers are referred with $P_1, \ldots, P_i$. Given a generic vector $v \in \mathbb{R}^n$, the diagonal matrix having the elements of v on the diagonal is indicated with $I_v \subset \mathbb{R}^{n \times n}$.

3.1 Graph theory background

A network can be represented through a simple graph (undirected and without self loops) $G(V, E)$, comprising of a node set V and a set of bidirectional links E. The maximum number of links in G are indicated with $m_V = \frac{|V|^2 - |V|}{2}$. There is a natural isomorphism between graphs and matrices. Given an undirected graph $G(V, E)$ it is straightforward to represent it with a symmetric adjacency matrix $A \in \{0, 1\}^{|V| \times |V|}$ where $A_{ij} = 1 \iff (i, j) \in E$. The set of neighbours of node i is indicated with $N_i = \{j \in V : (i, j) \in E\}$. Given a graph $G(V, E)$, $D : V \times V \to 2^E$ is the function mapping two vertexes i, j to the minimal ordered set of links connecting node i to node j computed using the Dijkstra algorithm on G[1].

[1] 2^E is the power set of E, meaning the set of all subsets of E

3.1.1 Graph properties

Being graphs a well investigated mathematical abstraction, several local and global properties have been formalized and studied. After the definition of each crucial graph property, the need to synthetically produce random graphs realizing the given property rises naturally. These synthetic graph generators generally take a given target graph family in input and produce a single random realization of it. In the following the global properties used in the rest of this work are introduced.

Degree sequence. The vertex degree sequence can be computed as $d = A\vec{1},\ d \in \mathbb{N}^{|V|}$ where $\vec{1} \in \{1\}^{|V|}$ is the unary vector. The degree sequence had been considered a crucial graph property for a long time, however, it is easy to see that very different graphs can have the same degree sequence. Indeed, early degree sequence based generators (configuration model) failed to produce graphs preserving important characteristics other than the degree sequence itself.

Average clustering. Average clustering is the measure of the tendency of graph nodes to form clusters. It is of particular interest in the context of online social networks [62], it measures how likely the cliques form and it is typically defined as:

$$\bar{c} = \frac{1}{|V|} \sum_{i \in V} \frac{2T_i}{d_i(d_i - 1)}$$

where T_i is the number of distinct triangles including node i (a triangle here indicates an induced sub-graph of three nodes connected one-another). Average clustering is one of the interesting graph properties modern graph generators target [63].

Modularity. Modularity is a well-known, widely used metric to measure and identify community structures in graphs [61]. It leverages the Newman definition of node groups as particular node subsets whose intra-group links are denser then inter-group links. The modularity value over a set of non-overlapping groups (a.k.a. partitions or communities) expresses the extent to which these partitions tend to be loosely connected one another but very well connected inside. It is of particular interest in sociology, since it is related to the notion of *cohesive subgroup* [64, 65]. In order to introduce more formally the concept of graph modularity, it is convenient to define first the *modularity matrix* $B \in \mathbb{R}^{|V| \times |V|}$ as:

$$B_{ij} = A_{ij} - \frac{d_i d_j}{\sum_i d_i}$$

Considering a given a set of node partition labels $\Pi_1, \ldots, \Pi_m$ with a corresponding partition node assignment $\pi_1, \ldots, \pi_{|V|} \in \{\Pi_1, \ldots, \Pi_m\}$ (such that node i belongs to partition π_i), modularity is defined as [61]:

$$Q = \frac{1}{\sum_i d_i} \sum_{i,j} B_{ij} \delta(\pi_i, \pi_j)$$

where $\delta(\pi_i, \pi_j) = 1 \iff \pi_i = \pi_j$ and 0 otherwise.

The notion of modularity presented can be used in three ways:

1. Given a partition node assignment $\pi_1, \ldots, \pi_{|V|} \in \{\Pi_1, \ldots, \Pi_m\}$, Q expresses how well the partition fits the topology of $G(V, E)$

2. The computation of Q can drive the search for the partition $\pi_1^*, \ldots, \pi_{|V|}^* \in \{\Pi_1^*, \ldots, \Pi_m^*\}$ that maximize its value, referred to as Q^*

3. The maximum modularity value Q^* for a graph $G(V, E)$ indicates to which extent its topology reflects a natural partition in communities (sometimes generically called the *modularity of the graph*)

3.1.2 Intersection graph

While generally graphs are defined starting from a node set V and an edge set E, they can also be defined using sets of elements. Intersection graphs are graphs defined in this way. Let $S \neq \emptyset$ be a set and $F = \{S_1, \ldots, S_n\}$ a family of sets $S_1, \ldots, S_n : S_i \subseteq S \ \forall i$. The intersection graph $\Omega(F)$ of F is made of a set of nodes $V = \{1, \ldots, n\}$ and a set of links $E = \{(i, j) \iff S_i \cap S_j \neq \emptyset\}$.

There is hence an isomorphism between graphs defined through $G(V, E)$ and graphs defined as intersection graphs $\Omega(F)$. The second interpretation can be convenient in certain cases like the one presented in Chapter 6. Figure 3.1 presents an example of intersection graph.

3.1.3 Centrality

In the context of networks, centrality has a broad range of meanings. Initially introduced in the field of sociology, generally speaking centralities rank the nodes or the links according to a measure indicating their *importance* in the network. The measure used depends on the definition of what a network node or link importance is.

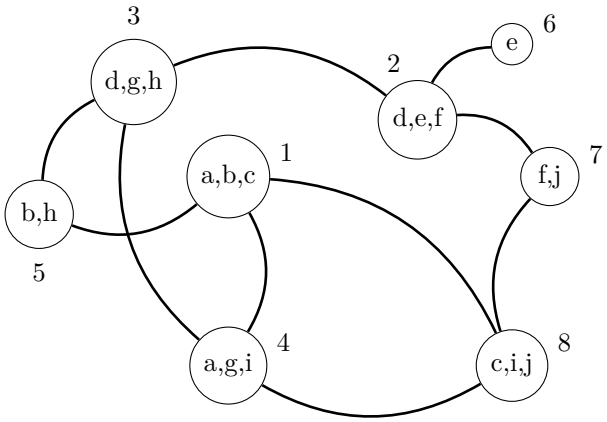


Figure 3.1 – Intersection graph with $S_1 = \{a, b, c\}$, $S_2 = \{d, e, f\}$, $S_3 = \{d, h, g\}$, $S_4 = \{a, g, i\}$, $S_5 = \{b, h\}$, $S_6 = \{e\}$, $S_7 = \{f, j\}$, $S_8 = \{c, i, j\}$.

Betweenness. Given the multiset of edges used in the shortest paths, $\mathcal{D} = \bigcup_{i=1,j>i}^{n} D(i,j)$, the link betweenness centrality of (i,j) is given by:

$$b_{ij} = \frac{m_{\mathcal{D}}(i,j)}{m_V}$$

where $m_{\mathcal{D}} : E \to \mathbb{N}$ is the multiset multiplicity function and it indicates how many shortest paths insist on a specific edge; i.e., the edge betweenness is a property of the graph edges defined, for each edge (i,j), as the fraction of the total number of shortest paths passing on that edge. Edge betweenness relates to the identification of more *important* edges in multi-hop communication; the more shortest paths insist on an edge, the more this edge is probable to be interested during communication routing.

PageRank. Here it follows a brief description of the PageRank centrality, the interested reader can check the work by Bryan and Leise [37] for a more comprehensive presentation. This centrality metric was originally created for the context of graphs where the nodes are web pages and the links are hyperlinks inter connecting those pages, but this concept can naturally be applied to any graph $G(V,E)$. The goal of PageRank is to measure the importance of the nodes (web pages). The importance of a node is defined with respect to the importance of its neighbours; if a node is linked by important nodes it means it is also important. In the rest, the PageRank value of node $i \in V$ is indicated with $x_i \in (0,1)$. Conventionally, if $x_i > x_j$ than node i is

more important (central) than node j $\forall i, j \in V$. By definition:

$$x_i = \sum_{j \in N_i} \frac{x_j}{d_j},\ \forall i \in V$$

Vector x is normalized so that $\sum_{i=1}^{|V|} x_i = 1$. Given an adjacency matrix A of $G(V, E)$ the PageRank vector x can be redefined as

$$x = \bar{A}x,\ \sum_{i=1}^{|V|} x_i = 1$$

where $\bar{A} = AI_d^{-1}$ is the matrix whose columns are normalized with respect to the corresponding node degree. If A is the adjacency matrix of a strongly connected graph $G(V, E)$ then $\bar{A}$ is irreducible and for the Perron-Frobenius theorem it has unique eigenvector with eigenvalue 1 which is also the maximum eigenvalue [66].

3.1.4 Spectral considerations

Every real and symmetric matrix $M \in \mathbb{R}^{|V| \times |V|}$, such as A, with eigenvectors $v_1, \ldots, v_n$ and eigenvalues $\lambda_1, \ldots, \lambda_{|V|}$ (all of them necessarily real) can be decomposed as a summation of matrices;

$$M = \sum_{i=1}^{n} \lambda_i v_i v_i^T$$

Without loss of generality, it is considered $\parallel v_i \parallel = 1\ \forall i$.

Given M has rank $|V|$, a low rank approximation $\tilde{M} \in \mathbb{R}^{|V| \times |V|}$ of M is a matrix of the same size but with lower rank $m < |V|$. Among the low rank m approximation matrices of M, the best one is the one closest to M with respect to the euclidean distance [67]. In the case of real symmetric matrices, such as M, and considering an ordering on the eigenvectors $v_i, \ldots, v_{|V|}$ such that $|\lambda_i| > |\lambda_j|\ \forall i < j$, such best approximation is given by the truncated sum of its spectral decomposition [68]:

$$\tilde{M} = \sum_{i=1}^{m} \lambda_i v_i v_i^T,\ m < |V|$$

3.2 P2P live streaming

P2P live streaming is about a network of hosts interested in receiving a multimedia content within a certain time deadline from another special host called the source. To this end, the source and the hosts orchestrate a decentralized real-time content distribution. The real-time constraint can be hard, if it requires the content to be received within few hundreds of milliseconds (as in the case of phone calls or video conferencing) or soft, if it just requires the content to be consumed withing few seconds (as in the case of the television and radio broadcasting). The content is typically divided into small *chunks* of data by the source in order to ease the information spreading in the network.

The idea behind the P2P approach is that information is not directly transmitted from the source to the nodes but instead the hosts cooperate and retransmit the content one to the other. There is a broad literature production detailing several possible schemes under which this distribution can happen. P2P systems are categorized according to their communication structure, which can be:

- tree based;
- multi-tree;
- unstructured.

This work focuses on unstructured systems as they prove to be more robust, reliable and more prone to fully exploit the available resources [69]. Section 3.2.1 presents the thesis contribution from the theory point of view with a mathematical framework tackling P2P distribution from a layered perspective.

3.2.1 3-layer P2P model

From the very beginning of this work, P2P distribution system has been logically divided into three different, almost independent, layers. Conceptually, there are three different logical topologies in a P2P distribution; the network of ISO/OSI layer three, namely hosts and links, called the **underlay**; the logical network that a P2P system builds on it during the process of neighbour selection and forming what is generally referred to as *topology* or *embedded network*, called **overlay** in the rest of this document to avoid confusion; and the **distribution** tree for a given data chunk, rooted at the source and consisting of all the nodes that received it and all the edges interested in the forwarding.

3.2.1.1 The underlay

Together with the media content to be distributed, the underlay can be considered the input of the distribution system. The underlay is made of layer three entities connected to each other, such as laptops, routers, smartphone, servers, etc. It is modeled with a simple graph $G_u(V_u, E_u)$ where V_u is the set of all networking devices and E_u the set of all the links interconnecting them. It is worth noticing that generally $G_u(V_u, E_u)$ is not known a priori but WCNs constitute an exception, being able to export their topology (see Section 2.1.1). Figure 3.2 shows an example of underlay in the form of mesh network.

3.2.1.2 The overlay

Given an underlay $G_u(V_u, E_u)$ and a content to be distributed, some underlay node users are interested in receiving this content. Each P2P user instantiates an instance of the P2P distribution system, generally called a *peer*. For the sake of simplicity the set of peers is considered as a subset of the set of underlay devices $V_o \subseteq V_u$. The approximation is reasonable to the extent one is not interested in modeling more than one running peer per underlay device, which, from a distribution point of view makes completely sense as once information reaches a node it should be forwarded directly to all its running peer instances.

Once a peer $P_i \in V_o$ is launched and bootstrapped, it receives messages listing random subsets of the other peers through the gossiping methods. Although gossiping

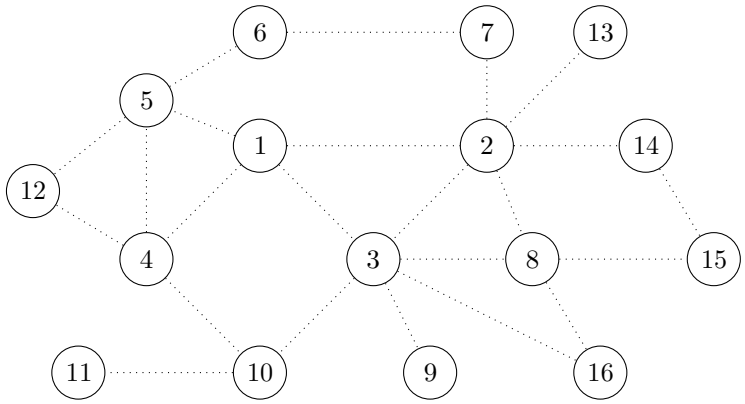


Figure 3.2 – Example of mesh underlay network represented as a graph $G_u(V_u, E_u)$. Hosts are numbered with an arbitrary ordering. Copyright © 2016 IFIP.

methods are not the focus of this work they are of main importance for a P2P system; they are used to periodically provide a *fresh* random sample of peers participating to the distribution [70, 71]. That is particularly useful in the context of live streaming which is generally subject to churn.

When each peer P_i receives the peer random sample, it can pick some peers as its new neighbours. Generally, peers target a specific neighbour set size so that they also drop some of their neighbour selections. The neighbour set construction and continuous update is the core of the P2P overlay building and it is called *rewiring*. Each node P_i creates its neighbour set $N_i \subseteq V_o \setminus \{P_i\}$ according to some *topology policy*. Given its robustness against churn, the most successful topology policies combine the random selection of neighbours with a fixed minimal neighbour set size. For neighbour set sizes much larger than $\log_2|V|$, those policies are granted (with high probability) to create connected Erdős-Rényi graphs.

Each neighbour selection involves the establishing of a logical communication link (P_i, P_j); if $P_j \in N_i$ then P_i and P_j exchange messages directly. Thus, the overlay links can be defined as $E_o = \{(P_i, P_j) : P_j \in N_i\}$. In a P2P system logical links are generally bidirectional, $(P_i, P_j) \in E_o \iff (P_j, P_i) \in E_o$ which implies $G_o(V_o, E_o)$ is also a simple graph. This, together with a suitable choice of the target neighbour set size, grants that resulting graph is strongly connected with high probability, a property very important in an unstructured P2P network [51].

Following the definition of E_o, the bidirectionality of links and the need to avoid graph disconnections, P2P systems do not consider refusing a logic link establishment. That means a peer P_i selecting a subset of peers as its neighbours can end up having a larger neighbour set N_i as it must accept incoming logical link establishments. In the following, this topology policy is referred to as *neighbourhood rule*. An example of overlay $G_o(V_o, E_o)$ is given in Figure 3.3.

3.2.1.3 The distribution

The media content is typically divided in a (possibly infinite) set of chunks $c_1, \ldots, c_k, \ldots$ to ease the distribution. The complete distribution of the content happens if each of the chunks c_k is propagated from the source to all the peers in V_o. Peers store the chunks in a chunk buffer to be able to forward them. Chunk buffers typically have a limited size determined by the level of liveliness of the distribution.

The propagation of each chunk happens through the overly $G_o(V_o, E_o)$ according to:

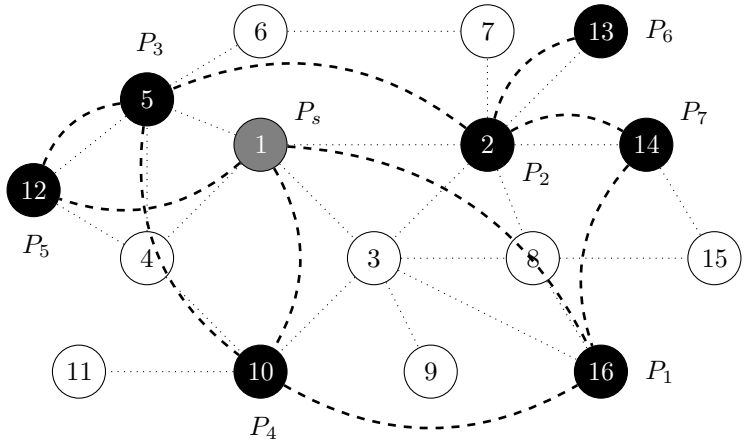


Figure 3.3 – Possible overlay graph $G_o(V_o, E_o)$ (black nodes and dashed edges) over the underlay graph of Figure 3.2; source P_s is represented in grey. Copyright © 2016 IFIP.

- a propagation scheme,
- chunk/peer scheduling.

The propagation is performed by each peer individually; they are expected to agree on a propagation scheme and compute the chunk/peer scheduling. From this decentralized point of view, the distribution can be described through the interaction of only two peers: the one that has some chunks in its chunk buffer (the sender) and the one seeking missing or latest chunks (the receiver). Note the two roles are not mutually exclusive as all peers are expected to eventually have some content in their chunk buffers and they all need newest or missing chunks, so they usually play both roles taking turns.

Propagation schemes require decisions on destination/source peers and on chunk identifiers which are performed through peer and chunk scheduling. Chunk and peer scheduling can be briefly described as the process of determining a suitable couple (c_k, P_j) for a given propagation scheme.

There are two main schemes well-known in literature for propagation: push and pull.

Push propagation scheme. A sending peer P_i schedules (c_k, P_j) and send c_k to P_j.
Pull propagation scheme. A receiver peer P_j schedules (c_k, P_i) and it sends a request for the chunk c_k to the peer P_i which takes care of committing the transfer.

An important element during scheduling is the knowledge about neighbour chunk buffer status; this information improves significantly the effectiveness of chunk schemes as it avoids scheduling non appropriate chunks for a given peer. For this reason the chunk buffer bitmaps, containing the information of the chunks in each peer chunk buffer, are periodically exchanged among neighbours. This information is generally piggybacked on top of gossiping messages but care must be taken in handling that data as its timeliness depends on gossiping internal timers and updates.

Moreover, even if peers have complete information about their neighbour chunk buffers, the lack of synchronization may result in unsuccessful chunk sending (e.g., when two peers push the same chunk to the same receiver). To further improve the propagation effectiveness, signalizing schemes are introduced in the picture. Signaling messages can typically be *offer* or *select* messages.

Push/pull mixed propagation scheme. A sending peer P_i schedules (c_k, P_j) and it sends an offer message to P_j for c_k. When P_j receives the offer it sends a select message back to P_i indicating whether or not it requires c_k. In the affirmative case, P_i sends c_k to P_j. This can be easily extended to an arbitrary number of offer/selected chunks so that, P_i schedules a tuple $(P_j, c_{k_1}, \ldots, c_{k_\rho})$ consisting of ρ different chunks and P_j can select a subset of size $\sigma \leq \rho$ of them.

The scheduling of $(P_j, c_{k_1}, \ldots, c_{k_\rho})$ is not trivial and cannot be easily done jointly peer/chunk-wise. Hence, scheduling is generally performed in two steps, combining two distinct functions for peer and chunk scheduling.

Peer scheduling. Peer P_i wants to transmit c_k, it determines the destination peer $P_j \in N_i$ through the peer scheduling. Examples of peer scheduling include: random peer, peer that needs c_k.

Chunk scheduling. Peer P_i wants to send a chunk to peer P_j, it determines the chunk c_k to be transmitted through the chunk scheduling. Examples of chunk scheduling include: random chunk, chunk useful for P_j.

In this work, the push/pull mixed scheme is used among the peers while the source always pushes its chunks to its neighbours (it is straightforward no peer can have the newly created chunks). The source P_s generates chunks with a certain periodicity; every τ_s milliseconds P_s generates and sends a new chunk c_k to a number m of its neighbours N_s. The chunk is propagated in the overlay $G_o(V_o, E_o)$ as each peer P_i sends an offer every τ_i milliseconds to one of its neighbours $P_j \in N_i$. The propagation of a generic chunk c_k over an overlay $G_o(V_o, E_o)$ is represented in Figure 3.4.

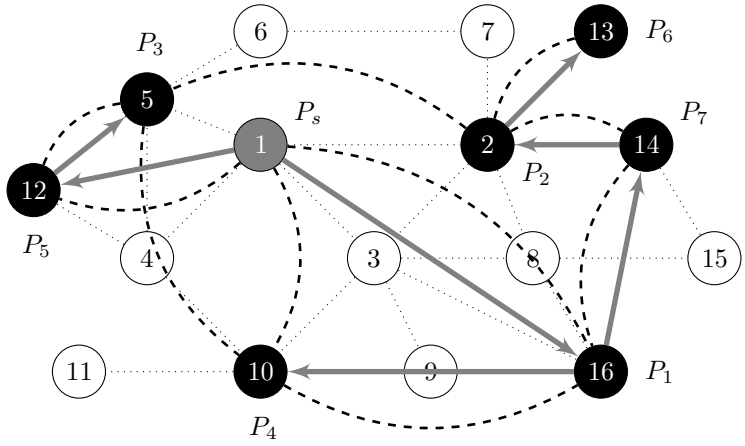


Figure 3.4 – Distribution tree (gray arrows) for a chunk c_k originated by the source P_s.

It is worth noticing that, if $\rho = 1$ then for the propagation of chunk c_k to happen it is necessary (but not sufficient) that

$$\frac{1}{|V_o|} \sum_i \tau_i \leq \tau_s \tag{3.1}$$

Intuitively, if the source is generating chunks at a rate faster than the average peer propagation rate, chunks cannot be forwarded to all peers as chunk buffer length is limited and, thus, they are eventually discarded.

To summarize, this work focuses on a offer/select push/pull scheme with timers τ_i and the use of peer first scheduling, so that when a peer P_i has to decide (P_j, c_k) for sending an offer it first executes a peer scheduling selecting P_j and secondly it selects c_k according to the *Latest-Useful* chunk selection policy which picks the newest chunk c_k that, presumably, P_j does not have.

Chunk scheduling has been extensively investigated [54, 72–75] and, under some assumptions, the existence of an optimal scheduling [76, 77] was proven. These works reveal the efficiency and robustness of the peer (possibly weighted) random selection strategy followed by the *Latest Useful Chunk* chunk scheduling strategy. This approach has recently been analyzed and extended under the name of High-bandwdith Peer First (HPF) [78].

Since the peer scheduling is performed randomly on a given distribution, it can be represented with a column stochastic adjacency matrix $\tilde{A}$ whose columns represent the

discrete probability distribution of peer selection. Hence, peer P_j schedules peer P_i with probability $\tilde{A}_{ij}$ (recall column $\tilde{A}_j$ indicates the probabilities of peer P_j sending a chunk to its neighbours). Note that, for a randomly injected chunk to be spread in the network, $\tilde{A}$ must be irreducible or, conversely, the graph resulting from $\tilde{A}$ has to be strongly connected.

Definition 3.1. *$\tilde{A} \in [0,1]^{|V|\times|V|}$ is a distribution matrix if*

- *$\tilde{A}$ is a column stochastic adjacency matrix,*
- *$\tilde{A}$ is irreducible.*

For the aforementioned neighbourhood rule (Section 3.2.1.2), it is requested that $\tilde{A}_{ij} = 0 \iff \tilde{A}_{ji} = 0$. In the trivial case (but in practice widely used) of uniform probability, the distribution matrix becomes $\tilde{A} = AI_d^{-1}$ where A is the adjacency matrix of $G_o(V_o, E_o)$ and $d = A\vec{1}$ is the peer degree sequence.

For the sake of ease modelling, τ_s is considered constant through time allowing the definition of the offer ratio $\theta_i = \frac{\tau_s}{\tau_i}$ which normalizes and expresses the amount of offers peer P_i sends per chunk time τ_s. Hence, it enables restating the necessary condition of Equation (3.1) in terms of offer ratios:

$$\sum_i \theta_i \geq |V_o| \tag{3.2}$$

Definition 3.2. *A distribution system for an overlay $G_o(V_o, E_o)$ is a couple $(\tilde{A}, \theta)$ such that:*

- *$\tilde{A}$ is a distribution matrix for $G_o(V_o, E_o)$,*
- *$\theta_i \geq 0 \ \forall i$.*

A distribution system $(\tilde{A}, \theta)$ is said minimal if $\vec{1}^T\theta = |V_o|$ and $\rho = 1$.

3.2.2 Performance measures

A live video streaming is characterized by a high sensitivity to reception delay. Generally, contents are considered live when delivered within few seconds. In any case, given its lively nature, peers are not requested to store the content indefinitely. Thus, the reception delay is ultimately impacted by the peer chunk buffer length which grossly determines, at steady state, the oldest content chunk obtainable.

Without considering underlay link losses, hence a chunk c_k cannot be downloaded by a peer P_i and it is considered lost if:

- it cannot be obtained from any neighbour $P_j \in N_i$,
- the c_k creation timestamp passed the system deadline;

with this in mind, two metrics are defined to measure the live distribution quality:

- (average) reception delay: the time interval from a chunk creation to its reception,
- (average) reception loss: the fraction of peers that received a chunk before it ceased to be forwarded.

However, the two metrics have not the same importance; distribution systems focus primarily on performing the actual distribution by improving the reception loss and, secondly, on improving the distribution quality by reducing the reception delay.

Chapter 4

Evaluation tools

This work is mainly about evaluating new approaches and communication strategies on mesh networks. For the sake of those evaluations comparison methods are required. The developed techniques are validate using different instruments at different abstraction levels.

A **testbed** is made of a real network reassembling a real-world scenario and it is the best choice for an on-the-field evaluation highlighting the major challenges to be tackled. An **emulator** makes it possible the experimentation with real software platforms in a controlled and designable environment, allowing corner case scenarios and validating the strategies on a much larger input set than the one available on testbeds. Finally, a **simulator** makes it possible to abstract everything but some specific algorithms and details, creating the best scenario for fine tuning and analysis and allowing comparisons without any interference.

In this work all of those three kind of tools are used and the actual ones are briefly introduced in the following of this section.

4.1 Community-Lab

Community-Lab[1] is a research infrastructure developed during the CONFINE project [79]. Unlike other popular testbeds like Emulab [80] or PlanetLab [81], it is meant to support the research on WCNs and foster their growth.

[1] https://community-lab.net/

Community-Lab is built starting from real WCN islands, the largest in Europe: guifi.net[2], FunkFeuer[3], Athens Wireless Metropolitan Network (AWMN)[4], Ninux[5] and Wireless België[6]. These "islands" are placed in different regions, respectively, Spain, Austria, Greece, Italy and Belgium, and they provide a variety of different infrastructures, communication schemes and protocols being a valid representative sample of the possible WCN setup available.

4.1.1 Experiments

Community-Lab is a network of *research devices* co-located in the aforementioned five WCNs. These research devices are special WCN nodes placed and set-up with the goal to offer researchers a controlled environment to launch experiments inside WCNs. These devices are generally not directly interconnected in the WCNs but linked to a real WCN node. In this way, research devices, and hence Community-Lab, provide realistic scenarios for experimenting inside the WCNs. The experiment managing functions are centralized and run from a specific testbed server that researchers can freely access.

Before running any actual P2P experiment at the application layer, the WCN properties and performance accessible through this testbed are evaluated. This step is needed so to evaluate the bias that other kind of experiments are subject to. A subset of research nodes are selected and their connectivity, in terms of link loss and link delay, is evaluated. Figure 4.1 shows the ICMP loss among the research devices. Data is averaged with respect to all the possible destinations (every other devices) and it shows the WCN links are quite reliable but for two badly connected devices.

Such difference can be further analysed considering the ICMP packet delay. Figure 4.2 presents the heatmap representation of the ICMP average RTT and its standard deviation among all devices. As it can be seen, devices seem to belong to different, well-intra-connected islands, but for the last two that appear badly connected with the all the others. In particular, research devices named from s1 to s10 belongs to a well intra connected group, s11 and s12 to another one which is not so well connected with the former while s13 and s14 data confirm their bad connectivity as showed in Figure 4.1.

[2]http://guifi.net
[3]http://funkfeuer.at
[4]http://www.awmn.net
[5]http://ninux.org
[6]http://www.wirelessantwerpen.be/

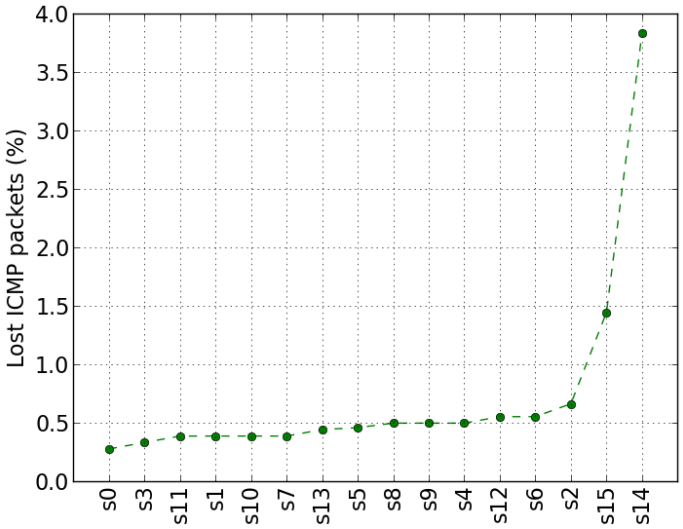


Figure 4.1 – ICMP traffic loss during experiments for each involved research devices. Copyright © 2014 IEEE.

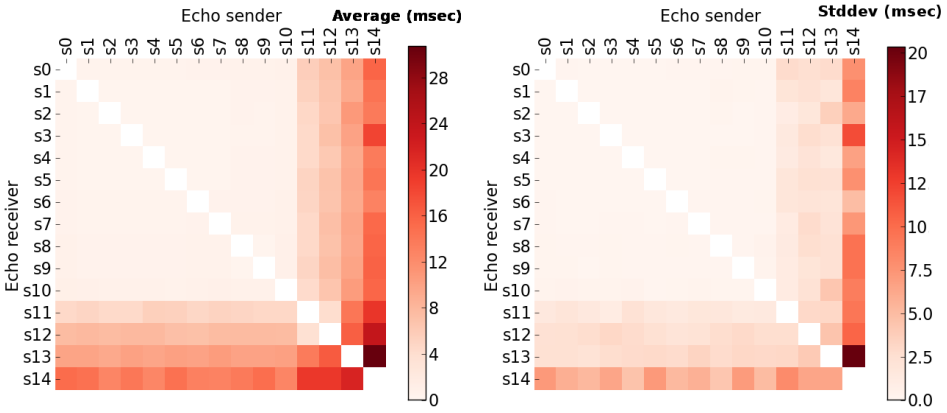


Figure 4.2 – Left plot: average RTT from each node to the others; right plot: related RTT standard deviation. Copyright © 2014 IEEE.

These results highlight the drawbacks of a real-word scenario, which, on the other side, offers researchers the opportunity to analyse and get insights before service deployments.

4.2 NePA TesT

Network Protocol and Application Testing Toolchain (NePA TesT) [5] is a development framework based on the Mininet emulator [82], created to rapidly prototype WCN oriented services on a broad ranges of realistic environments. NePA TesT also provides real-world datasets and topology generators, specific for WCNs. It is released as open-source software and it is freely available on-line[7].

The great advantage of emulators with respect to testbeds is their higher reliability and environment control, allowing the testing of corner cases as well the robustness assessment on huge numbers of scenarios. Moreover, they still permit the testing of ready to deploy code, as opposed to simulators.

Emulators like NEmu [83] and Naxim [84] take advantage from the QEMU [85] layer to virtualise an entire network. However, this kind of network emulation is much more resource hungry than *kernel namespace* ones, performed by Mininet, which allows the virtualization of thousands of nodes [82].

4.2.1 Mininet

With Mininet, emulated nodes share the same kernel, filesystem, memory and processor resources, but they have separated network environmnets. It has been created mainly for dealing with SDN experiments but it has been extended to ease the experimentation for WCN solutions.

4.2.2 Topology generator

The performance of communication systems are strongly affected by the network structure, i.e., by the underlying network topology they are relying upon. Characteristics like edge density, degree sequence and node number are just the simplest example of such determinant factors (see Section 3.1.1 for some advanced ones). Hence, it is mandatory to perform experiments on networks reassembling the real ones as much as possible. Moreover, most of the time, researchers want to abstract some key graph features, testing their algorithms on a large dataset of similar scenarios to assess their

[7] https://ans.disi.unitn.it/redmine/projects/community-newtork-emulator

solution robustness. To this end, they are interested in graph generators targeting some important properties.

Graph generation is not a trivial task and NePA TesT comes with a companion library capable of various kind of graph synthesizing including the well known Erdős-Rényi, Barabasi and Watts-Strogatz generators and the ones derived from real CN analysis performed by Milic and Malek [86] and Cerdá-Alabern [87].

4.2.3 WCN real data

Along with sound graph generators, NePA TesT comes with real world WCN topologies; snapshots of Ninux and FunkFuer had been taken and deployed so that reasearchers can test against them. It is worth noticing that NePA TesT supports input topologies in NetJSON format, which is a standard format many routing protocol implementations (including OLSRd and Batman-adv) export the topology with. Hence, researchers working with such protocols can export real world topologies and use them directly in the emulator for sandbox testing.

The link average loss rate is generally exported as meta-parameter of the topology itself so that emulation of losses can be implemented with standard distributions. Emulating a realistic link delay is generally more difficult. NePA TesT is shipped with a default CN delay model derived from a measuring campaign on the qMq Sants-UPC community network (a portion of Guifi.net)[8], hence, researchers can test networks with realistic topologies, loss and delay distributions.

4.2.4 Logging facilities

Simulating a system allows the abstraction of physical constraints as well. If the host machine cannot cope with a real-time simulation, simulations can slow down time and keep the simulation state sound. That does not apply to emulation in general, where measurements occur on real resources. Hence, if an emulator, or the machine hosting the emulation, do not have enough resources to support the intended scenario they can act as a bottleneck and influence the results. To avoid this possibility, NePA TesT comes with a logging module which keeps track of key indicators during the entire duration of the emulation. Such module can log CPU load, memory and swap usage. Researchers can analyze these logs a-posteriori and verify the emulation executed correctly.

[8] http://dsg.ac.upc.edu/qmpsu/index.php

4.2.5 Researcher interface

Each experiment is defined by its configuration file and the associated code and graph. Researchers are expected to write the test code to be executed in the emulated network extending the default test class. This code defines the network nodes behaviour and it usually consists in launching the target executable. Researchers have to select a network topology to be used; as mentioned, they can take advantage from NePA TesT network dataset, from graph generators or provide their own (in NetJSON or edge list formats). Finally they are required to write the experiment configuration file. This file specifies the test code to be executed, the network graph file and other experiment options.

Once the code, the graph and the configuration file are created, the experiment can be started through the python executable, as exemplified in Listing 4.1.

```
1 python nepa_test.py -f conf/peerstreamer.ini -t PSNRAND
```

Listing 4.1 – Invocation of NePA TesT using the configuration stanza PSNRAND defined in conf/peerstreamer.ini.

Figure 4.3 presents NePA TesT architectures and highlights the researcher interface part.

4.2.5.1 Test code

Researcher test code has to be integrated in the framework and follow some guidelines;

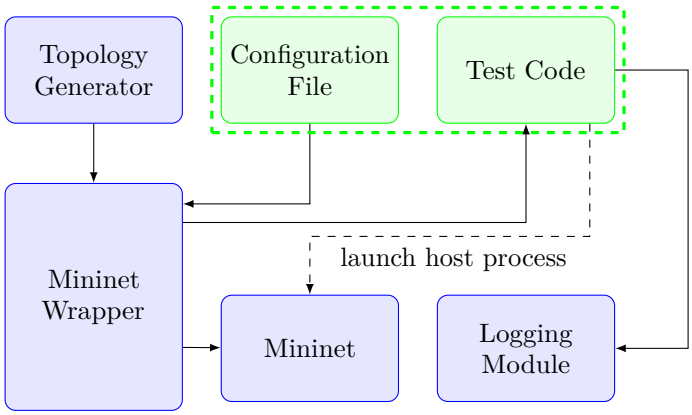


Figure 4.3 – NePA TesT architecture, the researcher interface is depicted in green.

- It has to be defined in a subclass of `MininetTest`,
- It has to implement the `runTest` function,
- It has to end the `runTest` function with a call to the *wait* function.

The wait function also takes optional parameters affecting the resource logging behaviour. *MininetTest* class also provides a set of important methods:

- `getAllHosts`: returns the list of all the network node objects,
- `getHostSample`: returns a random list of a subset of network nodes,
- `bgCmd`: executes a given shell command on a specified network node,
- `sendSig`: send a POSIX signal to a given network node,
- `killAll`: terminates all the precesses run by the network nodes,
- `setPrefix`: set the folder for storing experiment data and logs.

Researchers can also check the parameter values defined in the configuration file or passed through command line inspecting the class attribute `conf_args`. A typical experiment workflow follows the diagram showed in Figure 4.4.

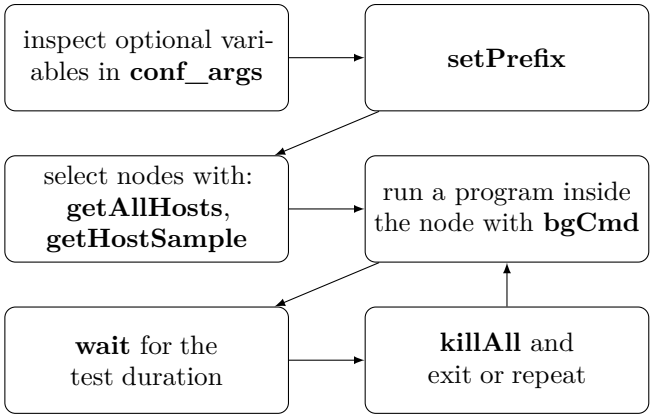


Figure 4.4 – Typical flow of the `runTest` function of a test class. Copyright © 2016 IEEE.

4.2.5.2 Configuration file

A NePA TesT configuration file follows the informal standard of *.ini* files. It comprises key values entries and sections. Those sections define NePA TesT configuration stanzas and relates to exactly one experiment. That means a researcher can define multiple stanzas in the same file. Moreover, these stanzas can inherit the configuration from other stanzas.

A configuration file example is given in Listing 4.2. To date, the following configuration parameters must be specified in any case: `testModule` which refer to the test code file name, `testClass` which specifies the test class in the test code file, `duration` expressing the experiment duration in seconds and `graphDefinition` indicating the network graph file name. The interested reader can find further setting details in the on-line project documents.

```
[PSNRAND]
testModule = peerstreamer
testClass = PSRandomNTest
graphDefinition = Nets/ninux0.edges
duration = 364
num_peers = 30
neigh_size = 10
times = 10
distoptimization = 0
xloptimization = 0
chunks_per_second = 175
chunks_per_offer = 30
log_chunks = 1
log_signals = 0
log_neighbourhood = 0
source_chunk_multiplicity = 1
push_strategy = 0
aframe_per_chunk = 5
chunk_buffer_size = 200
link_bw = 10
link_mean_delay = 8.0658ms
link_delay_sd = 55.7166ms
link_delay_distribution = qmp_delay_m8.0658_s55.7166
link_loss = wifi_loss
min_nodes_num = 10
max_nodes_num = 30
nodes_num_inc = 5
```

Listing 4.2 – An example configuration file; the fist line declares the stanza name and the following first four parameters are mandatory while the others are experiment specific (related to PeerStreamer)

4.3 SSSim

Simple and Scalable Simulator for P2P scheduling algorithms (SSSim) [88] is a P2P simulator designed with the goals of: flexibility, extensibility, performance (both in terms of CPU load and memory usage) and scalability. It has been mainly developed to test and compare different peer/chunk scheduling solutions (see Section 3.2.1.3 for details on P2P distribution and scheduling) for one-to-many real time streaming.

As comparing different distribution systems capable of scaling up to tens or hundreds of thousands of nodes, this work needs that kind of simulator because performance and scalability play a major role. Since the original release of SSSim did not allow the specification of user defined scheduling timers, it has been extended so that it is now able to entirely simulate complex P2P distribution processes like the one by Wu and Li [59] and the one presented in Chapter 7. All the code, is freely available on-line[9].

4.3.1 Contribution

This work adds the following features to SSSim:

- Event-driven scheduling simulation: enables researchers to test scheduling algorithms using fractions of chunk time (τ_s, see Section 3.2.1.3), hence, allowing a broader range of strategies to be implemented;
- Definition of the simulated topology through input file (edge list format): eases the experimentation with large graph datasets as well as with well-defined corner cases;
- Topology preprocessing as an adjacency matrix: allows the computation of the PageRank centrality metric for the topology nodes;
- Self-estimation of distribution convergence: simulation time is a very important matter when trying to establish the steady state behaviour of scheduling algorithms; the self-estimation of convergence feature of SSSim allows to end the simulation when steady state reaching is detected.
- Wu-Li scheduling optimization [59];
- PageRank scheduling optimization (Chapter 7).

[9] https://ans.disi.unitn.it/redmine/projects/sssim

Chapter 5

WCNs and video streaming

As introduced in Chapters 1 and 2 video streaming is a popular application, a service a modern network needs to sustain. At the same time, the P2P architecture and philosophy seem a perfect fit for the WCN structure and services.

This chapter answers two fundamental questions that arise from such considerations:

- Can WCNs support state-of-art P2P services designed for the internet?
- Is P2P video streaming feasible on WCNs? Under which conditions?

To answer these questions PeerStreamer, the NAPAWINE project outcome, is picked as the state-of-art P2P platform, and experiments are conducted using the Community-Lab testbed (see Section 4.1) to collect realistic results from real-world WCNs. A live P2P video streaming is considered successful in this context if the average chunk delay is withing few hundreds of milliseconds and the average chunk loss is lower than 10%.

5.1 Experiment setup

For experiments, nodes from two big WCNs, the Guifi.net and the AWMN ones, are taken into consideration. As described in Section 3.2.2 measurements relate to reception loss and delay and, additionally, to the number of overlay hops chunks perform on average to reach all the peers. That metric is of particular interest in mesh networks as link usage is costly and should be optimized.

Test networks comprise of from 24 to 28 nodes for the Guifi.net WCN and from 10 to 12 nodes for the AWMN WCN. Each experiment is composed of several runs (from

10 to 30 depending on the scenario stability) lasting for 10 minutes, out of which the central 5 minutes are analysed to avoid transient behaviours. During the experiment a $300\,\mathrm{kbit\,s^{-1}}$ encoded version of Big Buck Bunny[1] is streamed among the peers.

5.2 Parameter selection

Given the customizability of P2P systems and the research nature of PeerStreamer, the parameter space to be explored is dimensionally non-trivial and numerically unbounded. The following of this section lists the parameters and related ranges selected to be analysed (also summarized in Table 5.1).

With reference to the notation introduced in Section 3.2.1, $\tau_i = \tau_s - \epsilon,\ \epsilon \in [0, \tau_s)$ are set so that, on average, the offer rate is slightly larger than the chunk generation pace.

From the topological point of view, the target neighbour set size $N_N = |N_i|\ \forall i \in V,\ N_N \in \{5, 10, 20\}$; from the distribution point of view $sc = \rho = \sigma \in \{1, 3, 5\}$, the seeding multiplicity $m \in \{1, 3\}$ and the amount $fa \in \{1, 5\}$ of audio frame to pack in a single chunk. The last parameter is specific of PeerStreamer and relies on the fact that generally audio frames are quite small in size (at least compared to video ones) so it can be convenient to group them together. On the other hand, that has the drawback of increasing the reception delay of such data.

Table 5.1 – PeerStreamer parameter space for experimenting on WCNs

Parameter	Symbol	Range
Target neighbour set size	N_N	$\{5, 10, 20\}$
Offer/select chunk multiplicity	sc	$\{1, 3, 5\}$
Seeding chunk multiplicity	m	$\{1, 3\}$
Audio frames per chunk	fa	$\{1, 5\}$

5.3 Smart seeding

As noted in Section 4.1.1, certain WCN links can be quite lossy and this loss can be dangerous for a distribution system. The source chunk seeding is particularly sensitive; if the source P_S injects c_k in the overlay and that is immediately lost due to a lossy link, then no peer receives it and c_k has to be recovered through the offer/select process from the source P_s.

[1] https://peach.blender.org/

To avoid this possible corner scenario to happen the source seeding probability distribution is modified so that it is more likely the source sends chunks on reliable links. PeerStreamer chunk exchange mechanism includes a chunk receipt acknowledgement packet to be sent upon reception and such mechanism is used to maintain a moving estimation on successful chunk receptions. The peer scheduling is then weighted with such average probability.

Formally, P_s selects $P_i \in N_s(t)$ for chunk seeding at time t with probability

$$p_i^s(t) = \frac{w_i(t)}{\sum_{j:P_j \in N_s(t)} w_j(t)} \tag{5.1}$$

$$w_i(t) = \begin{cases} \alpha + w_i(t-1)(1-\alpha) & \text{if ack received by } P_s \\ w_i(t-1)(1-\alpha) & \text{if a timeout expires} \end{cases} \tag{5.2}$$

In the experiments $\alpha = 0.01$ and a timeout of $10\,\mathrm{s}$ are set. In the following two seeding strategies, the uniform (legacy) and the weighted, are indicated with Au and Aw respectively.

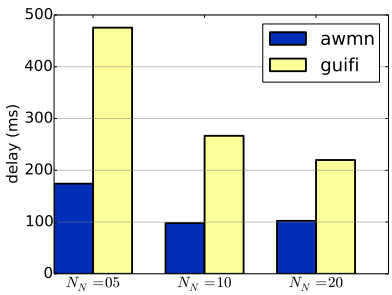
5.4 Comparative results

Figure 5.1 presents the averaged performance metrics obtainable varying N_N on both networks. It is worth noticing that the overlay in the AWMN comprises of only 11 nodes so the results for $N_N \in \{10, 20\}$ are straightforwardly identical. As it can clearly be seen, $N_N = 5$ is absolutely not appropriate in networks like Guifi.net as the delay and loss are quite high. This result is unexpected as, given the overlay size, $N_N = 5$ should be enough to grant connectivity and a good degree of offer redundancy; thus,it is probably due to poor underlay link quality. On the other hand, $N_N = 10$ grants to obtain a delivery success of almost 100% with a delay lower than $300\,\mathrm{ms}$.

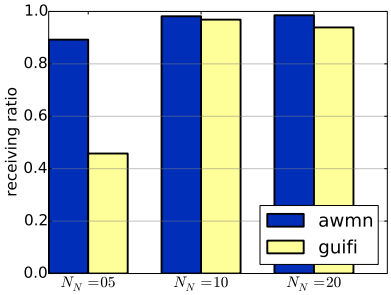
Figure 5.2 shows the impact of distribution parameter variation; as expected, increasing *fa* improves the receiving ratio (by reducing the total number of chunks per second) and it slightly increases the average reception delay. This increment is however tolerable within the context of live video streaming.

The role of *sc* is less obvious to analyse and a dramatic performance drop for $sc = 5, fa = 1$ is reported. When $fa = 5$ performance for both reception delay and loss improves linearly with *sc* but with $fa = 1$ there is an optimum at $sc = 3$.

Figure 5.3 reports the performance varying the seeding multiplicity m. As expected, increasing this multiplicity improves the performance (loss is always almost 0% and it

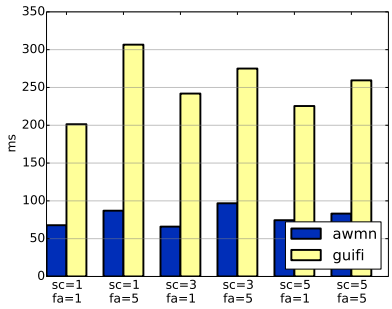


(a) Average chunk delay.

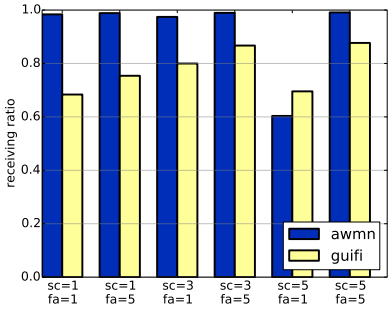


(b) Average chunk receiving ratio.

Figure 5.1 – Chunk level performance in the two WCNs as a function of N_N, $m = 1$. Copyright © 2015 Elsevier.



(a) Average chunk delay.

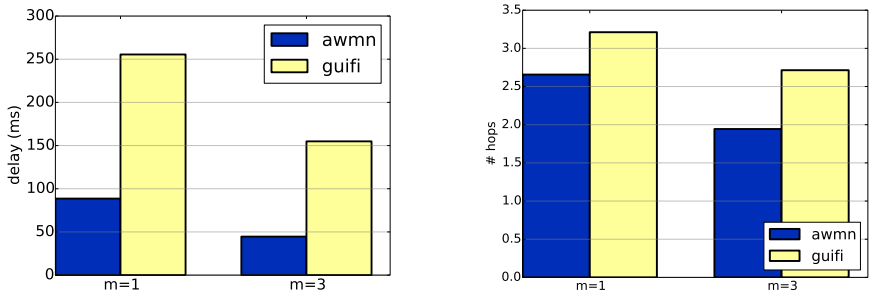


(b) Average chunk receiving ratio.

Figure 5.2 – Chunk level performance in the two WCNs, varying *sc* and *fa* with $N_N = 10$, $m = 1$. Copyright © 2015 Elsevier.

is not reported for brevity) and it reduces the diameter of the distribution tree (see Section 3.2.1.3) and, hence, the amount of hops chunks have to traverse to be fully distributed. This should not surprise as the amount of resources spent by the source P_s is increased.

Au and *Aw* performance are compared in Figure 5.4. Being Guifi.net quite a lossy network, P2P distribution gets a noticeable improvement both in reception delay and loss by using the strategy Aw. Conversely, being AWMN links quite reliable, performance variation is negligible. It is interesting to note that *Aw* on Guifi.net implies an increase of chunk distribution hops. The reason is that, with *Aw*, chunks are seeded less uniformly in the overlay and, on average, they have to be forwarded

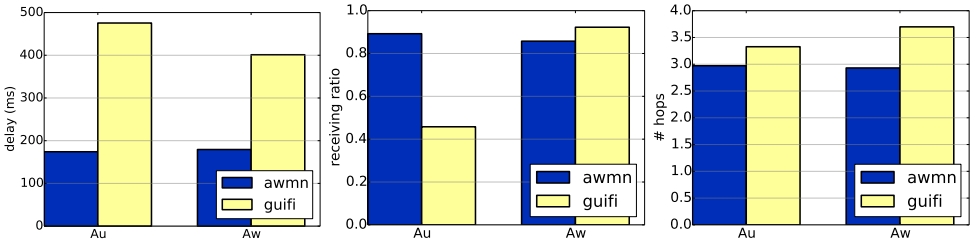


(a) Average chunk delay. **(b)** Average number of hops per chunk.

Figure 5.3 – Chunk level performance in the two WCNs, w.r.t. m; $N_N = 10, sc = 3, fa = 5$. Copyright © 2015 Elsevier.

more times to reach all the peers. This effect does not affect the key performance metrics as better transmission conditions avoid retransmissions and delays.

Finally, the answer to the two questions placed at the beginning of this chapter can be given: live video streaming on WCNs with a state-of-art platform is feasible as the average chunk delay is within few hundreds of milliseconds and the average chunk loss can be reduced below 10%.



(a) Average delay. **(b)** Average delivery ratio. **(c)** Average number of hops.

Figure 5.4 – Chunk level performance in the two WCNs, $N_N = 5, sc = 3$, and $fa = 5$. Copyright © 2015 Elsevier.

Chapter 6

Topology optimization

This chapter is devoted to P2P overlay optimization for mesh networks. Concepts from Section 3.2.1.2 are integrated and extended with those from graph theory (Section 3.1) to formally derive optimized distribution networks.

In particular, one of the most useful characteristics of WCNs and mesh networks in general is exploited: the availability of the network topology $G_u(V_u, E_u)$ (see Section 2.1.1). In this chapter, the focus is on networks managed with OLSR, as they allow nodes to export the complete topology in every node.

This can been used disruptively in P2P network optimization using cross-layer techniques. Hence, the focus is on the overlay rewiring problem: given the underlay $G_u(V_u, E_u)$ and a set of peers V_o, $V_o \subseteq V_u$, the goal is to define the most efficient and effective overlay $G_o(V_o, E_o)$ and a distributed, lightweight algorithm (meaning it requires negligible resource overhead) to implement it.

It is worth highlighting how this *cross-layer* approach does not require any modification of the lower levels, easing the deployment on large real-world scenarios. Furthermore, it does not introduce any level of centralization in the distribution system like *super-peers* or CDN-like approaches, realizing a fully decentralized strategy.

6.1 Overlays in mesh networks

Logical overlays $G_o(V_o, E_o)$ are built on top of the network underlay $G_u(V_u, E_u)$ and they constrain the distribution process; i.e., the actual data chunks will traverse the logical links E_o during the communication. Such constraining determines the stress over the underlay network resources (comprising of link usage time and bandwidth); so

we can identify a measure of optimality of such constraining considering the resulting resource stress.

To clarify this point, the toy example of Figure 6.1 presents a comparison between two overlays with respect their optimality in terms of resource stress; the nodes 1, 2 and 3 and the dotted lines represent $G_u(V_u, E_u)$, two possible overlays with $V_o = V_u$ are shown with dashed lines. The under optimized overlay a) forces the data chunks to go first from the source (node 1) to node 3 and then be retransmitted to node 2. That implies the underlay links $(1, 2), (2, 3)$ are used 1 and 2 times respectively. On the other hand, the optimized overlay b) forces the chunks to go from the source to node 2 first and then to be retrasmitted to node 3, thus using the underlay links optimally (only once each). Note these two overlays can be fairly compared as they have the same number of edges.

Generalizing, optimization should aim at reducing the underlay link overloading. Two optimization directions are identified:

- Logical links should map to the shortest possible underlay network path;
- Logical links should prefer underlay network path comprising of under-used underlay links.

The first bullet is straightforward and it derives from the observation of the toy example of Figure 6.1. The second bullet is more subtle and deals with the over-using of specific underlay links which could have a high edge betweeness (see Section 3.1.3).

From the point of view of the overlay, we seek to:

- Minimize the **load** (use) on the underlay links, favoring shorter paths;
- Maximize the **fairness** of underlay links usage, avoiding the overloading of one particular link.

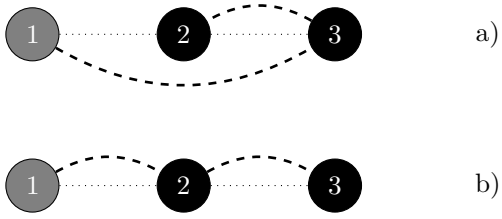


Figure 6.1 – a) Example of one under optimized overlay (nodes and dashed lines in upper plot) and, b) Example of one optimized overlay (nodes and dashed lines in lower plot) for the same underlay (nodes and dotted lines). Source node is represented in gray.

The mesh networks $G_u(V_u, E_u)$ are assumed to change slowly through time, meaning that its links and their attributes do not vary significantly. This, paired with the overlay optimization described, implies a convergence through time of the overlay $G_o(V_o, E_o)$ for a given $G_u(V_u, E_u)$.

6.2 Cross-layer Overlay metrics

Considering each single underlay link $l \in E_u$, a cost $w(l),\ w : E_u \to \mathbb{R}^+$ is assigned. This approach makes it possible to optimize through total cost minimization; moreover, it naturally fits routing protocol weighting systems that use the Expected Transmission Count (ETX) metric [89], such as, OLSR.

With respect to the definition given in Section 3.1 of a network Dijkstra function, $D_{G_u} : V_u \times V_u \to 2^{E_u}$, the cross-layer Dijkstra function $D^{\dagger}_{G_u} : V_o \times V_o \to 2^{E_u}$, is the function returning the underlay shortest path between two overlay nodes. The load $\mathcal{L}_o$ imposed by an overlay $G_o(V_o, E_o)$ on a underlay $G_u(V_u, E_u)$ can be formally restated as:

$$\mathcal{L}_o = \sum_{e \in E_o} \sum_{l \in D^{\dagger}_{G_u}(e)} w(l) \tag{6.1}$$

Let $\mathcal{H}_o(l)$ be the weighted number of logical links loading l:

$$\mathcal{H}_o(l) = |\{e \in E_o \ : l \in D^{\dagger}_{G_u}(e)\}| \cdot w(l) \tag{6.2}$$

The link Jain's fairness can be thus defined as

$$\mathcal{F}_o = \frac{\left(\sum_{l \in E_u} \mathcal{H}_o(l)\right)^2}{|E_u| \sum_{l \in E_u} \mathcal{H}_o(l)^2} \tag{6.3}$$

Note that $\mathcal{F}_o \in \left[\frac{1}{|E_u|}, 1\right]$ but $\mathcal{F}_o$ is not expected to reach its maximum as there are underlay links not supporting any overlay links.

Equations (6.1) and (6.3) express formally what it is introduced in Section 6.1 and define the metrics used to evaluate the overlay. Unfortunately, the two criteria are often opposing and thus they cannot be directly used to build an optimum overlay. A combined metric balancing both those aspects is thus needed.

6.3 Cross-layer link descriptor

In order to mathematically bind together the overlay $G_o(V_o, E_o)$ and the underlay $G_u(V_u, E_u)$ and to formally state the optimization problem, a cross-layer representation of the overlay edges is used. To this end, let's introduce an indexing faction assigning a unique identifier to each link of a simple graph. The *triangular element* indexing function $\text{triel}_V : V \times V \to [1, m_V] \cap \mathbb{N}$ is defined as:

$$\text{triel}_V(i,j) = \begin{cases} j - 1 + \frac{(i-1)(2|V|-2-i)}{2} & \text{if } i < j \\ i - 1 + \frac{(j-1)(2|V|-2-j)}{2} & \text{otherwise} \end{cases} \tag{6.4}$$

The triel_V function is bijective, hence it can be used as invertible mapping between the space of links E and the set of integers $[1, m_V] \cap \mathbb{N}$.

triel_{V_u} is used to give an ordering to the underlay links, $l_r = (i,j) \in E_u,\ r = \text{triel}_{V_u}(i,j),\ i,j \in V_u$. This ordering makes it possible to represent each underlay link l_r with a vector representing the underlay link occupied among all the possible ones in $V_u \times V_u$; e.g.,

$$\bar{l}_r = (0, \ldots, 0, 1, 0, \ldots, 0) \in \{0,1\}^{m_{V_u}}$$

Obviously, $\| \bar{l}_r \|_2 = 1,\ \forall r$.

Let's now define the core of the cross-layer framework: the cross-layer link descriptor; triel_{V_o} is used to give an ordering to the overlay links, $e_k = (i,j) \in E_o,\ k = triel_{V_o}(i,j),\ i,j \in V_o$. It is now possible to represent each overlay link e_k with a vector $\bar{e}_k$ encoding the underlay link elements involved along the shortest path between the peers $i,j \in V_o$:

$$\bar{e}_k = \left(\sum_{l_r \in D^{\dagger}_{G_u}(i,j)} \bar{l}_r \right) \in \mathbb{N}^{m_{V_u}},\ k = \text{triel}_{V_o}(i,j)$$

Summarizing, each element $\bar{e}_k$ binds the overlay edge $e_k = (i,j) \in E_o$ to the actual underlay elements $l_r = (s,t) \in E_u$ involved in the shortest path. Moreover, those vectors belong to an euclidean representational space with well defined metrics.

Considering again the scenario in Figure 6.1 ($m_{V_u} = m_{V_o} = 3$), the computation of the cross-layer edge descriptor for $(1,3) \in E_o$ is the following:

$$\begin{aligned}
&\text{triel}_{V_u}(1,2) = 1 \\
&\text{triel}_{V_u}(2,3) = 3 \\
&\text{triel}_{V_o}(1,3) = 2 \\
&\qquad \bar{l}_1 = (1,0,0) \\
&\qquad \bar{l}_3 = (0,0,1) \\
&\qquad \bar{e}_2 = (1,0,0) + (0,0,1) = (1,0,1)
\end{aligned}$$

The triel function and the optimization functions are efficiently implemented in a python module freely available on-line[1].

6.4 Cross-layer intersection graph

In this section a given overlay $G_o(V_o, E_o)$ is turned into an intersection graph (see Section 3.1.2). This step is needed to exploit the cross-layer edge descriptor potential and optimize the overlay as a whole.

Let's create a family of $|V_o|$ sets, $S_1, \ldots, S_{|V_o|}$, (maintaining the same ordering of the nodes) where:

$$S_i = \{\bar{e}_k : \ e_k \in E_o, \exists j \in V_o, \ \text{triel}(i,j) = k\}$$

S_i is the set of the cross-layer edge descriptors related to the links from node i to the nodes $j \in N_i$. Let's call $S = \bigcup_{i=1}^{V_o} S_i$ the set of all the cross-layer edge descriptors and $\Omega(S_1, \ldots, S_{|V_o|})$, or Ω for short, the cross-layer intersection graph resulting from $G_o(V_o, E_o)$. Figure 6.2 shows the intersection graph corresponding of the overlay of Figure 3.3.

Let's denote with $S'_i = \{\bar{e}_k : k = \text{triel}(i,j) \, \forall j \in V_o\}$ the complete set of edge descriptors for node i; $S'_i \supseteq S_i \ \forall i \in V_o$.

Let's call the underlay edge occupancy of Ω the vector $\bar{E} = \sum_{\bar{e}_k \in S} \bar{e}_k W$, where $W \in \mathbb{R}^{m_{V_u} \times m_{V_u}}$ is the cost matrix built starting from the cost function $w()$:

$$W_{r,q} = \begin{cases} w(s,t), \ r = \text{triel}_{V_u}(s,t) & \text{if } r = q \\ 0 & \text{otherwise} \end{cases}$$

[1] https://ans.disi.unitn.it/redmine/projects/overlay-theory/repository/overlay_optimization

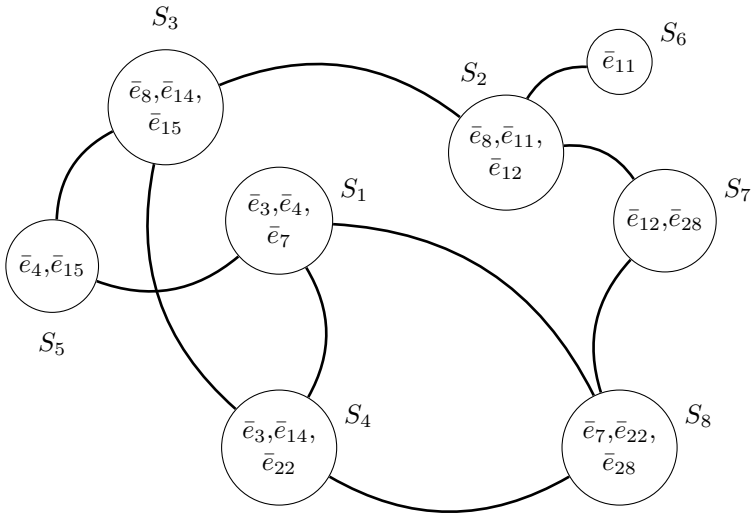


Figure 6.2 – Overlay intersection graph according to Figure 3.3. Elements $\bar{e}_k$ are the cross-layer overlay edge descriptors between the nodes. Copyright © 2016 IFIP.

With this in mind, Equations (6.1) and (6.3) can be restated in terms of the cross-layer intersection graph, respectively:

$$\mathcal{L}_\Omega = \vec{1}^T \bar{E} \tag{6.5}$$

$$\mathcal{F}_\Omega = \frac{(\sum_{k=1}^{m_{V_u}} \bar{E}_i)^2}{m_{V_u}(\sum_{k=1}^{m_{V_u}} \bar{E}_i^2)} \tag{6.6}$$

6.5 Overlay optimization

Given the underlay graph $G_u(V_u, E_u)$ and a set of participating peers V_o, the goal is to build the overlay $G_o(V_o, E_o)$, meaning building the sets $S_1, \ldots, S_{V_o}$, such that both $\mathcal{L}_\Omega$ and $\mathcal{F}_\Omega$ are minimal. That is a multi-objective combinatorial optimization problem; however our context allows the definition of a combined metric $\mathcal{C}_\Omega$ that express the cost of the overlay:

$$\mathcal{C}_\Omega = \left|\left| \sum_{k=1}^{|S|} \bar{e}_k \, W \right|\right|_2 \tag{6.7}$$

Note that Equation (6.7) takes into account both $\mathcal{L}_\Omega, \mathcal{F}_\Omega$ by exploiting the L_2-norm distance interpretation in the euclidean space of $\mathbb{N}^{m_{V_u} \times m_{V_u}}$. For the sake of simplicity, but without loss of generality, the cost matrix equal to the identity matrix is used, $W = I$. The overlay optimization problem can finally be stated as:

$$\arg\min_{z} \left|\left| \sum_{k=1}^{m_{V_o}} z_k \bar{e}_k \right|\right|_2 \tag{6.8}$$

subject to:

$$\sum_{\bar{e}_k \in S'_i} z_k \geq d; \quad \forall i = 1...|V_o| \tag{6.9}$$

where

$$z_k = \begin{cases} 1 & \text{if } \bar{e}_k \in \ S \\ 0 & \text{otherwise} \end{cases}$$

The problem variable vector $z \in \{0,1\}^{m_{V_o}}$ contains the indicator variables deciding which overlay edge e_k to keep. $d \in \mathbb{N}$ is the minimum node degree and must be set to avoid the trivial solution of $z = \vec{0}$. d must be large enough to grant the graph connectivity with high probability and it is usually set to $d > \log_2(|V_o|)$. Equation (6.9) presents a little abuse of notation for the sake of readability; the sum spans over all $\bar{e}_k \in S'_i$ but it is indeed a sum over $k : k = \text{triel}(i,j),\ j \in V_o$.

Equation (6.8) can be rephrased as: find the intersection graph Ω with minimum degree d defined by $S_1, \ldots, S_{|V_o|}$ so that Equation (6.7) is minimized.

Back to the toy example of Figure 6.1, we have $S' = \{\bar{e}_1, \bar{e}_2, \bar{e}_3\}$ with

$$\bar{e}_1 = (1,0,0), \bar{e}_2 = (1,0,1), \bar{e}_3 = (0,0,1)$$

If we set $d = 1$, the optimal solution to Equation (6.8) is $z = (1,0,1)$ corresponding to selecting the edge descriptors $S = \{\bar{e}_1, \bar{e}_3\}$ and, hence, to the overlay edge set $E_o = \{(1,2),(2,3)\}$, which in turn corresponds to the optimized case of b) in Figure 6.1.

6.5.1 NP-hardness

In this section it is shown that Equation (6.8) is a zero-one quadratic programming problem [90], similar to the quadratic knapsack one [91] and, hence, it is a NP-hard problem. This kind of problems are in the form of

$$\min_{z} c^T z + z^T Q z \tag{6.10}$$

subject to:

$$h_i^T z + z^T \Psi z > g_i \tag{6.11}$$

where $z = \{0,1\}^n$ is a vector of binary variables, $c, h_i \in \mathbb{R}^n$, $Q, \Psi \in \mathbb{R}^{n \times n}$ are symmetric and $g_i \in \mathbb{R}$.

Equation (6.8) can be expressed in the form of Equation (6.10) by picking Ψ as a zero matrix, $c = \vec{0}, g_i = -d$, h_i as a binary vector selecting the component of z pertaining S_i and $Q = \hat{E}^T \hat{E}$, where $\hat{E}$ is the matrix whose columns are the elements $\bar{e}_k$, in fact:

$$\arg\min_z \left|\left| \sum_{k=1}^{m_{V_o}} z_k \bar{e}_k \right|\right|_2 = \arg\min_z z^T \hat{E}^T \hat{E} z$$

Being NP-hard, Equation (6.8) cannot be easily computed but there are state-of-art algorithms making it tractable up to a certain size (m_{V_o}) using the branch-and-bound technique. However, since m_{V_o} grows quadratically with the number of peers $|V_o|$, those algorithms can be used only for very small overlays.

6.5.2 Relaxations

Being the goal the decentralized optimized overlay rewiring, in this section relaxed problems of Equation (6.8) are presented so to:

- Reduce the computational complexity;
- Make the resulting algorithm decentralized, enabling each peer $P_j \in V_o$ to select independently its neighbours $N_j \subset V_o$ according to the metrics in Equations (6.1) and (6.3) and communicate its choice.

Considering a generic peer P_j, the contribution of the edges selected by P_j is separated from all the other edge contributions in Equation (6.8),

$$\left|\left| \sum_{k=1}^{|S|} z_k \bar{e}_k \right|\right|_2 = \frac{1}{2} \left|\left| \sum_{i=1, i \neq j}^{V_o} \sum_{\bar{e}_k \in S_i'} z_k \bar{e}_k + \sum_{\bar{e}_k \in S_j'} z_k \bar{e}_k \right|\right|_2 \tag{6.12}$$

The factor $\frac{1}{2}$ is needed as each element $\bar{e}_k$ is considered twice as independently selected by two different peers $(P_i, P_j : \text{triel}_{V_o}(i,j) = k)$. Calling b_j the vector representing the

choices of all the peers in V_o but P_j, Equation (6.8) can be restated as:

$$\arg\min_{z} \left|\left| \sum_{k=1}^{|S|} z_k \bar{e}_k \right|\right|_2 = \arg\min_{z} \left|\left| b_j + \sum_{\bar{e}_k \in S'_j} z_k \bar{e}_k \right|\right|_2 \tag{6.13}$$

To go a step further in the decentralization of this rewiring optimization, the centralizing assumption that each peer P_j knows about other peers choices b_j is dropped . That means peer P_j has to guess as accurately as possible the value of b_j to do its neighbour choices. Hence, a reasonable approximation $\bar{b} \simeq b_j$ that node P_j can use in (6.13) is needed.

Consider the limit case in which every host in the underlay contains a peer ($|V_o| = |V_u|$) and let b^* be the value of $\bar{E}$ normalized to the total number of shortest paths:

$$b^* = \bar{E} \frac{2}{|V_o|^2 - |V_o|}$$

In this case, b^* is exactly the vector corresponding to the betweenness centrality of each link in E_u as defined in Section 3.1.3.

If $|V_o| < |V_u|$, b^* is an approximation of the link centrality vector; this is a well-known fact and a widely used technique to compute centralities in large networks comprising a number of nodes too high to compute all the shortest paths [92]. The convergence of b^* to the actual betweenness vector is pretty fast in power-low graphs, extremely frequent in communication networks and in large CNs [87]. Each peer P_j can thus reasonably approximate b_j following the shape of b^* which, representing link betweenness, gives a measure of the underlay link usage frequencies.

Given the constraint on the (minimum) overlay node degree d, it follows that $|E_o| \simeq \frac{d|V_o|}{2}$. This leads us to compute the approximation $\bar{b} \simeq b_j$ as:

$$\bar{b} = b^*(|V_o|-1)\frac{d}{2} = \bar{E}\frac{d(|V_o|-1)}{|V_o|^2-|V_o|} = \bar{E}\frac{d}{|V_o|} = \frac{d\sum_{k=1}^{m_{V_o}} \bar{e}_k}{|V_o|} \tag{6.14}$$

Computation of Equation (6.14) is polynomial in time and, in sparse graphs, it can also take advantage from a vast literature on efficient betweenness computation [93].

The first, distributed relaxed version of Equation (6.8) can now be stated:

$$\arg\min_{z} \left|\left| \bar{b} + \sum_{\bar{e}_k \in S'_j} z_k \bar{e}_k \right|\right|_2 \tag{6.15}$$

$$\text{conditioned to: } \sum_{\bar{e}_k \in S'_j} z_k \geq d \tag{6.16}$$

Despite Equation (6.15) can now be implemented distributively, it is still a zero-one quadratic minimization problem and, hence, NP-hard. However, its number of variables is now linear and not quadratic anymore with $|V_o|$ so a state-of-art solver can be used to tackle its solution for overlays comprising of hundreds of nodes. During experiments, the YALMIP library [94] is used for solving Equations (6.8) and (6.15).

The next relaxation aims to decrease the computational cost by considering now each single edge descriptor $\bar{e}_k$ separately, and weighting it with the expected link occupancy distribution by the other peers $\bar{b}$.

$$\arg\min_{z} \sum_{\bar{e}_k \in S'_j} z_k \left|\left|\bar{b} + \bar{e}_k\right|\right|_2 \tag{6.17}$$

$$\text{conditioned to: } \sum_{\bar{e}_k \in S'_j} z_k \geq d \tag{6.18}$$

For the sake of comparison it is worth considering the extreme case where $\bar{b} = \vec{0}$, in this case (and with $W = I$) the L_2-norm gives the same ordering as the L_1-norm and solving Equation (6.19) builds the peer neighbourhood with closest peers in terms of hop counts.

$$\arg\min_{z} \sum_{\bar{e}_k \in S'_j} z_k \left|\left|\bar{e}_k\right|\right|_2 \tag{6.19}$$

$$\text{conditioned to: } \sum_{\bar{e}_k \in S'_j} z_k \geq d \tag{6.20}$$

The role of $\bar{b}$ is hence to introduce a bias that peers consider when deciding which underlay link to occupy. Table 6.1 summarizes the different optimization strategies which are meant to be implemented in the rewiring module of the P2P systems (see Section 3.2.1.2). PeerStreamer is extended to feature the decentralized, polynomial ones.

Name	Symbol	Objective Function
Global Optimization	G_o	$\| \sum_{k=1}^{m_{V_o}} z_k \bar{e}_k \|_2$
Local Optimization	L_o	$\| \bar{b} + \sum_{\bar{e}_k \in S'_j} z_k \bar{e}_k \|_2$
Local Equalized Ranking	E_r	$\sum_{\bar{e}_k \in S'_j} z_k \| \bar{b} + \bar{e}_k \|_2$
Local Ranking	L_r	$\sum_{\bar{e}_k \in S'_j} z_k \| \bar{e}_k \|_2$

Table 6.1 – A summary of the optimization strategies with their symbols.

Name	Complexity	Decentralized	Number of variables
Glob. Opt.	NP-hard	No	$\|z\| = \|V_o\|(\|V_o\|-1)/2$
Loc. Opt.	NP-hard	Yes	$\|z\| = \|V_o\|-1$
Loc. Eq. Rank.	Polynomial	Yes	$\|z\| = \|V_o\|-1$
Loc. Rank.	Polynomial	Yes	$\|z\| = \|V_o\|-1$

Table 6.2 – A summary of the optimization strategies and their complexity attributes.

6.6 Simulation and emulation results

Table 6.1 reports a summary of the presented overlay optimization strategies along their symbols used in the following; Table 6.2 highlights the differences among those strategies from the complexity point of view. All strategies compared and evaluated with two different approaches; first with an ad-hoc simulator to assess them excluding any interfering mechanism, second emulating a real implementation of the strategies of interest, the decentralized ones, to measure the impact of the overlay optimization on P2P streaming performance. For the sake of comparison, the results from the random rewiring strategy are also reported, as it is largely used on Internet P2P applications (Section 3.2.1.2).

6.6.1 Simulations

The simulator is implemented with Python and the NetworkX library, specifically designed to deal with graphs. This simulator takes an underlay topology $G_u(V_u, E_u)$ and a set of peers $V_o \subset V_u$ as input, applies the different strategies and computes for each of them the load $\mathcal{L}_\Omega$ and the fairness $\mathcal{F}_\Omega$.

Validation of relaxations. Well known network models are taken into considerations, namely Erdős-Rényi and Barabási-Albert, to assess the validity of the relaxations. Comparing against G_o implies keeping $|V_o|$ small for the complexity reasons.

The results for $|V_o|= 20$, presented in Figures 6.3 and 6.4, validate the relaxation techniques being the resulting overlays, in terms of fairness and load, very close to the optimum one. It is worth mentioning all the presented strategies but L_r outperform the random one for the given metrics. This should not surprise as they take advantage from additional information on the underlay. L_r strategy results show it grants less link usage fairness than the random strategy; that is due to its overloading of specific links with high edge betweeness. Results look particularly promising in the Barabási-Albert case in Figure 6.4, which shows the decentralized polynomial strategy E_r very close to the first, NP-hard, relaxation L_o and, hence, to the global optimum G_o.

Scaling. Let's now test how strategies behave when increasing $|V_o|$. Experiment varies $|V_o| \in [20, 100]$ but, being results very similar, only the ones for $|V_o|= 100$ are shown. Being of greater interest, experiments focus on Barabási-Albert random topologies. Hardware limitations do not allow computation of the G_o strategy anymore.

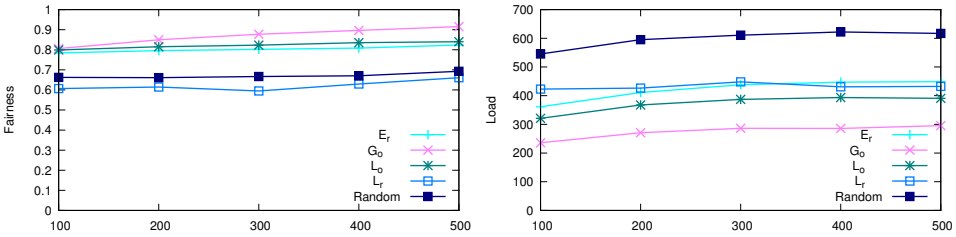


Figure 6.3 – Load and fairness of Erdős-Rényi random underlays with $|V_o|= 20$, varying $|V_u|$. Copyright © 2016 IFIP.

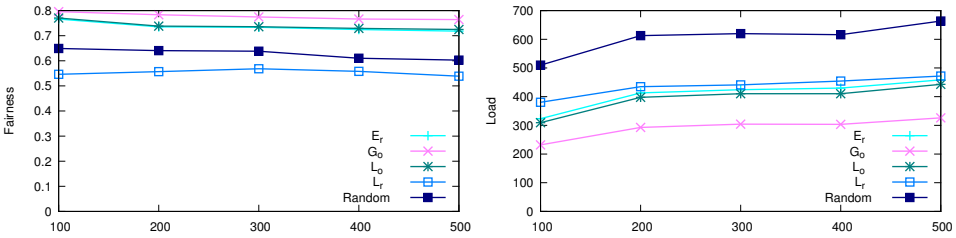


Figure 6.4 – Load and fairness of Barabási-Albert random underlays with $|V_o|= 20$, varying $|V_u|$. Fairness of E_r almost perfectly overlaps with the one of L_o. Copyright © 2016 IFIP.

Figure 6.5 confirms that E_r nicely obtains the same performance as L_o which are particularly distant from L_r especially in terms of fairness. It is interesting to note L_r obtains lower fairness with respect to the random strategy, this is mostly due to the fact random choices follow the underlay link distribution of b^* while L_r, being basically an hop-count based strategy, simply picks the shorter distances.

As previously introduced in Section 6.1, WCNs typically do not have backbones nor high capacity links, and neglecting link usage fairness simply leads to congestions and bottlenecks.

WCN-like topologies. Strategies are tested against realistic topologies and verify the preservation of the nice properties highlighted for the Barabási-Albert model. To this end Cerdá-Alabern networks are used, generated through the homonym algorithm [87] and resulting from a generalization of several real-world CNs. The same setup with $|V_o|= 100$ is used, and results shown in Figure 6.6 confirm the previous findings; again performance of E_r and L_o are very close while L_r fairness is lower than random. Note also that L_r load is considerevoly greater than the E_r one; that is due to the neighbourhood rule (Section 3.2.1.2) which makes the overlay built with L_r having more links.

6.6.2 Emulations

NePA TesT is used for the emulations (see Section 4.2) with two real-world WCN topologies taken from the Ninux and FunkFueier networks (respectively the Rome island comprising 131 nodes and the Wien island comprising of 236 nodes). The available real network ETX information is considered to emulate link loss, while considering a

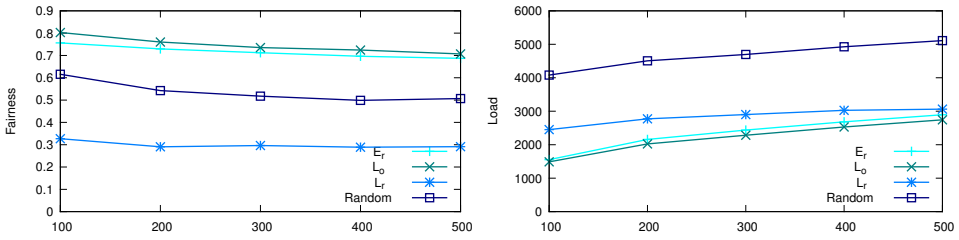


Figure 6.5 – Load and fairness of Barabási-Albert random underlays with $|V_o|= 100$, varying $|V_u|$. Copyright © 2016 IFIP.

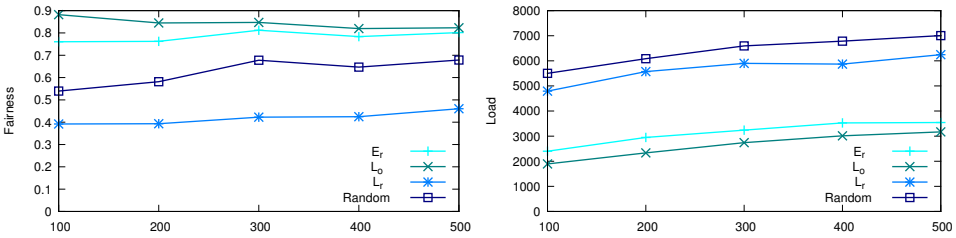


Figure 6.6 – Load and fairness of Cerdá-Alabern random underlays with $|V_o|= 100$, varying $|V_u|$. Copyright © 2016 IFIP.

constant link bandwidth of $10\,\mathrm{Mbit\,s^{-1}}$ and a uniform delay distribution with support [30,1000] µs .

PeerStreamer streams the Big Buck Bunny video with a bit rate of $300\,\mathrm{kbit\,s^{-1}}$ among $|V_o|= 30$ (randomly selected) peers and target degree $d = 10$.

Overlay metrics. PeerStreamer outputs the rewired overlay $G_o(V_o, E_o)$ generated during the emulation and it is possible to compute its metrics $\mathcal{L}_\Omega, \mathcal{F}_\Omega$. Without considering the absolute values, results shown in Figure 6.7 are perfectly compatible with the simulation outcomes for both the topologies, confirming E_r being the one with smaller load and higher fairness.

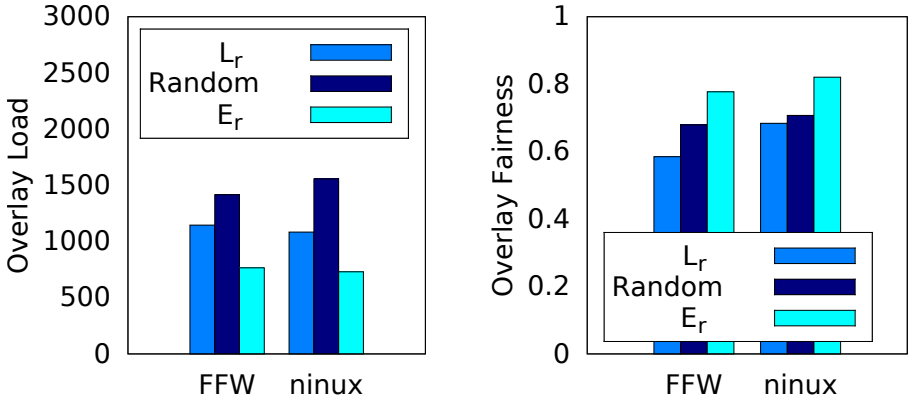


Figure 6.7 – Load and fairness of the overlay graph computed on FFWien and ninux topologies. Copyright © 2016 IFIP.

Sent data analysis. The overlay directly impacts the distribution, but to check the real link usage in terms of load and fairness, one must consider the actual data being sent. To this end, one can compute the distribution load as the total sum of megabytes sent on all the links and the fairness as the Jain fairness among the links considering the total amount of megabytes sent on each of them. Intuitively, such data should reflect the overlay load and fairness. Figure 6.8 presents that data; as can be seen, the load plot has the same shape as the corresponding in Figure 6.7 while the fairness plot is slightly different. The send data analysis confirms E_r strategy is the one that most offloads the links and fairly use the network resources.

Underlay link usage. Figure 6.9 further highlights the effort distribution property E_r strategy obtains. The plot presents the actual megabytes sent per underlay link, sorted according to the usage. It is worth noticing that: i) curves are nicely separated, meaning the total amount of data sent using E_r is smaller, ii) random strategy presents the highest peak while the other two significantly offload the more congested links iii) the E_r strategy alter the distribution to make it look more uniform, meaning it fairly distributes the resource usage.

6.7 Neighbourhood pruning

So far this chapter considered a *blind* overlay optimization, however, knowledge of the peers and their shortest paths, this information can be used smartly. The focus is now on overlays for applications without churn or with churn-preventing mechanism, e.g.,

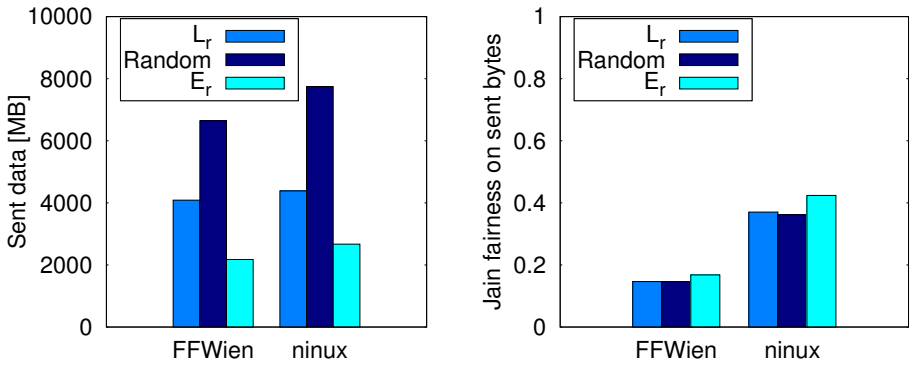


Figure 6.8 – Sent data and data fairness of the overlay graph measured with PeerStreamer on real topologies. Copyright © 2016 IFIP.

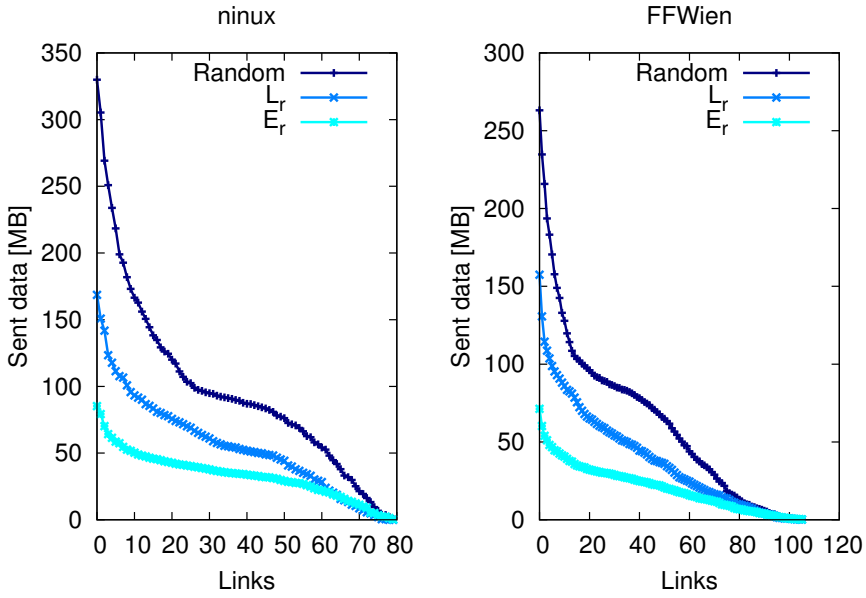


Figure 6.9 – Load measured on each link on real topologies. Copyright © 2016 IFIP.

video conferencing. Considering again Figure 6.1, the presented scenario implies that building an overlay with a fixed degree $d = 2$ means a complete waste of resources. In fact there is no point for node 1 to send data to node 3, as it would be transmitted through node 2 links without any benefit for the latter. A limitation on the possible neighbourhood choices for node 1 would grant it cannot include node 3 in its neighbour set at all.

The focus of this section is hence highlighting how, regardless of the underlay topology, when a peer P_j selects its neighbours N_j, if there is a peer P_z lying on the shortest path between P_j and P_i, there is no point for P_j to include P_i in its neighbour set N_j. Each peer should prefer as neighbours those peers that are first encountered along any shortest path route so that data packets do not "jump" suitable distribution destinations, hence wasting resources. The enforcement of this policy can be done by limiting the neighbour choices for each peer P_j.

From the intersection graph Ω point of view that means limiting the edge descriptors $\bar{e}_k$ each peer P_j can select when running the decentralized strategies of Table 6.1.

Algorithm 1 present the decentralized neighbour candidate pruning algorithm to be run before executing the strategies. $S_j^{'}$ is again the set of all possible overlay edges

1: $S_j^p = \emptyset$
2: **repeat**
3: $\quad i \leftarrow \arg\min_{i \in S_j^{'}} |D^{\dagger}_{G_u}(i,j)|$
4: $\quad S_j^p = S_j^p \cup \{\bar{e}_{\text{triel}(i,j)}\}$
5: $\quad S_j^{'} = S_j^{'} \setminus S_j^p$
6: $\quad S_j^{'} = S_j^{'} \setminus \{\bar{e}_{\text{triel}(z,j)} : \bar{e}_{\text{triel}(z,j)} \in S_j^{'}, (i,s) \in D^{\dagger}_{G_u}(z,j),\ s \in V_u\}$
7: **until** $S_j^{'} = \emptyset$

Algorithm 1: Neighbour candidate pruning algorithm for peer P_j

starting from P_j and the algorithm loops over to filter out the pruned set $S_j^p \subseteq S_j^{'}$. Note line 6 removes from the possible overlay edge set $S_j^{'}$ all the edges (z, j) whose shortest path includes passing through node i. Table 6.3 summarizes the strategies using the pruning techniques along with their formulas.

6.8 Pruning simulation results

Using the same simulator from Section 6.6.1, all the strategies are tested using the real-world WCN topologies of Ninux and FunkFeuer already used in Section 6.6.2

Results shown in Figures 6.10 and 6.11 are quite similar. $|V_o|$ is varied to assess the scaling of the P2P impact and it can be seen that, while the non pruned strategies have a load linear with $|V_o|$, the pruned ones are almost constant. That is due to the lowering of the mean degree, as shown in the *Mean connectivity* sub plots. Pruning also improves L_r fairness for $|V_o| \rightarrow |V_u|$ and offloads the most loaded underlay links as show in the *Maximum link load* plots.

In conclusion, pruning improves blind choices like the hop-count based especially when peer number gets close to the underlay node number.

Name	Sym.	Objective Function	Complexity	Decent.
Pruned Loc. Opt.	L_{op}	$\| \vec{b} + \sum_{\bar{e}_k \in S_j^p} z_k \bar{e}_k \|_2$	NP-hard	Yes
Pruned Loc. Eq. Rank.	E_{rp}	$\sum_{\bar{e}_k \in S_j^p} z_k \| \vec{b} + \bar{e}_k \|_2$	Polynomial	Yes
Pruned Loc. Rank.	L_{rp}	$\sum_{\bar{e}_k \in S_j^p} z_k \| \bar{e}_k \|_2$	Polynomial	Yes

Table 6.3 – A summary of the pruned optimization strategies and their attributes.

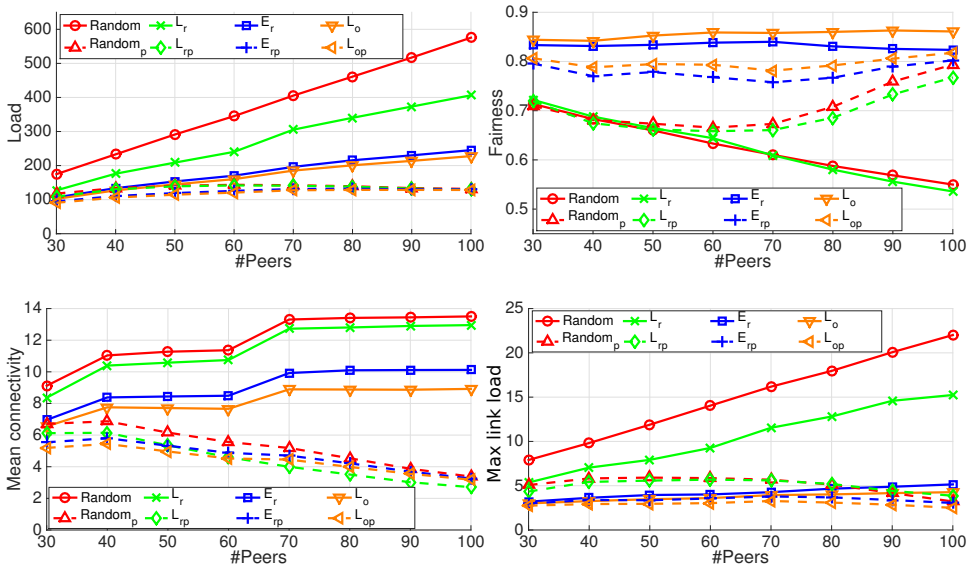


Figure 6.10 – Overlay optimization comparison on the Ninux network, varying $|V_o| \in [30, 100]$. Copyright © 2017 Elsevier.

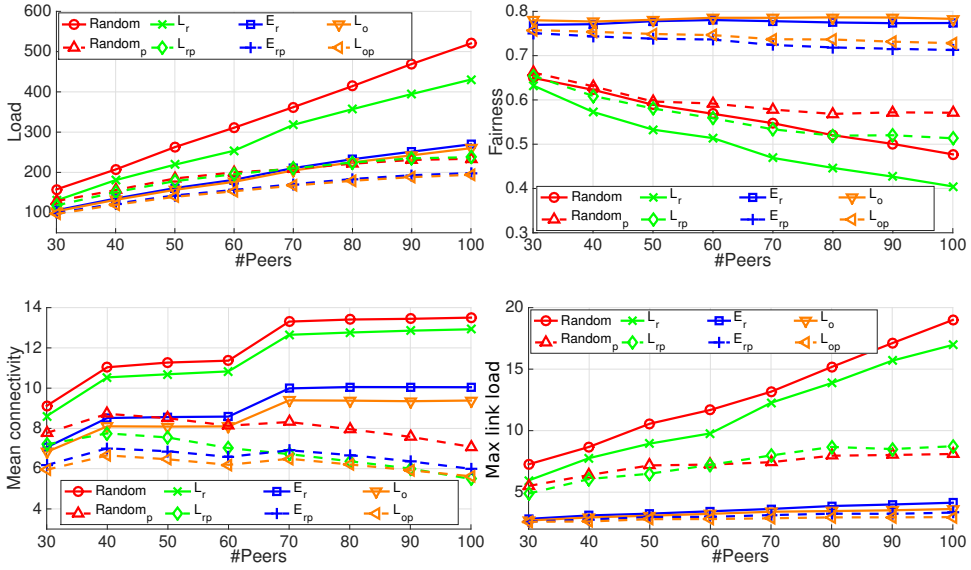


Figure 6.11 – Overlay optimization comparison on the FunkFeuer network, varying $|V_o| \in [30, 100]$. Copyright © 2017 Elsevier.

Chapter 7

Distribution optimization

This chapter focuses on the optimization of the P2P distribution process as described in Section 3.2.1.3. In the following, the actual overlay $G_o(V_o, E_o)$ is abstracted in favor of the derived distibution process $(\tilde{A}, \Theta)$. Note that $\tilde{A} \in \mathbb{R}^{|V_o| \times |V_o|}$ must be irreducible but it can be not symmetric. The goal is to optimize the distribution process parameters, meaning the sending probability matrix $\tilde{A}$ and the message rate vector Θ, with respect to the overall receiving delay and loss. Despite the original focus on live streaming, this distribution abstraction and the results presented can be applied on a broad range of communication scenarios related to data broadcasting in a mesh network, like sensor networks, vechicular networks, etc. In fact, the distribution process abstraction applies to all the systems realizing a decentralized communication where information has to be spread network-wide.

7.1 Reception-equal process

As introduced in Section 3.2.1.3, $(\tilde{A}, \Theta)$ completely characterizes the HPF scheduling which, in turns, realizes the distribution. In general, P2P systems assume a constant Θ and a column-uniform $\tilde{A}$. Indeed this case is the simplest, as each peer P_i sends Θ_i chunks per chunk time to one of its neighbour selected with uniform probability $\frac{1}{|N_i|}$, with $\Theta_i = \Theta_j,\ \forall P_i, P_j \in V_o$. This configuration is referred to with the name *sending equal* as all the peers send the same amount of information per chunk time. However, that means on average different peers can receive different amount of information.

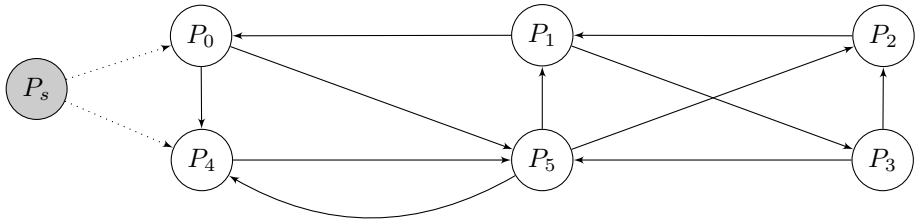


Figure 7.1 – Sample network of six nodes, source depicted in grey. Copyright © 2017 IEEE.

The following toy example clarifies this point. Given the strongly connected network presented in Figure 7.1 and the corresponding sending-equal distribution process:

$$\tilde{A} = \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{3} \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{3} \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} & 1 & 0 \end{bmatrix}; \Theta = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{7.1}$$

The average received amount of chunks $\Phi = \tilde{A}\Theta$ can be readily computed. It is worth noticing Equation (7.1) describes a minimal system (see Section 3.2.1.3), $\vec{1}^T\Theta = |V_o|$.

$$\Phi = \begin{bmatrix} 0.5 \\ 1.333 \\ 0.833 \\ 0.5 \\ 0.83 \\ 2 \end{bmatrix} \tag{7.2}$$

As the example reports in Equation (7.2) there are chances some peers receive, on average, more information (where $\Phi_i > 1$) while others *starve* (where $\Phi_i < 1$). Generally, this problem is overcome by linearly increasing the resources spent, so that each peer P_i sends an amount $\alpha\Theta_i,\ \alpha \in (1, \infty)$ of information per chunk time, resulting in a reception multiple of Φ, $\tilde{A}\alpha\Theta = \alpha\Phi$.

A better approach to improve fairness and peer cooperation is a distribution process $(A^\star, \Theta^\star)$, still minimal, for the same network granting a *reception-equal* property,

$A^\star\Theta^\star = \vec{1}$. That implies a redistribution of network resource usage optimizing the overall process.

7.2 Optimized distribution

Formally, given a distribution matrix $\tilde{A}$ and $|V_o|$, the goal is to find a couple $(\Theta^\star, A^\star)$ that satisfies:

$$\vec{1} = \Phi = A^\star\Theta^\star \tag{7.3}$$

$$|\Theta^\star| = |\vec{1}| = |V_o| \tag{7.4}$$

$$\vec{1}^T A^\star = \vec{1}^T \tag{7.5}$$

$$\tilde{A}_{ij} = 0 \iff A^\star_{ij} = 0 \tag{7.6}$$

Equation (7.3) imposes the reception-equal constraint; Equation (7.4) keeps the distribution system minimal and Equations (7.5) and (7.6) grant $A^\star$ to be irreducible and column-stochastic, hence a valid distribution matrix while preserving the same structure of $\tilde{A}$ (no edge is added or removed).

Theorem 7.1. *Let $G_o(V_o, E_o)$ be a strongly connected network graph and let $\tilde{A}$ be its associated distribution matrix. Then it is possible to find $A^\star$ and $\Theta^\star$ such that the conditions given by Equations* (7.3) *to* (7.6) *hold.*

Proof. Since $\tilde{A}$ is irreducible and column stochastic, the Perron-Frobenius states that the largest eigenvalue of $\tilde{A}$ is 1, and it exists a corresponding eigenvector x with all strictly positive entries that sum to 1:

$$\tilde{A}x = x;\ x^T\vec{1} = 1 \tag{7.7}$$

From the diagonal matrix definition in Chapter 3 it follows that

$$\tilde{A}I_x\vec{1} = x \tag{7.8}$$

and, since the elements of x are strictly positive:

$$I_x^{-1}\tilde{A}I_x\vec{1} = I_x^{-1}x = \vec{1} \tag{7.9}$$

If choosing $\Theta^\star = (\vec{1}^T I_x^{-1} \tilde{A} I_x)^T$ and $A^\star = I_x^\star \tilde{A} I_x I_{\Theta^\star}^{-1}$, then, Equation (7.3) is satisfied by:

$$A^\star \Theta^\star = I_x^{-1} \tilde{A} I_x I_{\Theta^\star}^{-1} \Theta^\star = I_x^{-1} \tilde{A} I_x \vec{1} = I_x^{-1} \tilde{A} x = I_x^{-1} x = \vec{1}$$

Equation (7.4) is satisfied by:

$$|\Theta^\star| = \Theta^{\star T} \vec{1} = \vec{1}^T I_x^{-1} \tilde{A} I_x \vec{1} = \vec{1}^T I_x^{-1} \tilde{A} x = \vec{1}^T I_x^{-1} x = \vec{1}^T \vec{1} = |\vec{1}|$$

Equation (7.5) is satisfied by:

$$\vec{1}^T A^\star = \vec{1}^T I_x^{-1} \tilde{A} I_x I_{\Theta^\star}^{-1} = \Theta^{\star T} I_{\Theta^\star}^{-1} = \vec{1}^T$$

To verify Equation (7.6) let us explicit each element of $\Theta^\star$ and $A^\star$:

$$\theta_j^\star = \sum_{k=1}^{|V_o|} \frac{a_{kj} x_j}{x_k}; \quad a_{ij}^\star = a_{ij} \frac{x_j}{x_i \theta_j^\star} \tag{7.10}$$

since elements of x are positive and at least one element per column in $\tilde{A}$ is larger than zero, then $\forall j \; \theta_j^\star > 0$. Therefore, $\forall i, j \; a_{ij}^\star \geq 0$ and $a_{ij}^\star = 0 \iff a_{ij} = 0$ which satisfies Equation (7.6). □

Moreover, if the system resources are linearly increased:

$$A^\star \alpha \Theta^\star = \alpha \vec{1}$$

The benefit spreads uniformly in the network and all the nodes receive on average α amount of information, which can guarantee streaming in a lossy environment.

7.2.1 Properties

The basic interpretation of the role of the eigenvector centrality x in the reception-equal optimization relies on the original PageRank model (Section 3.1.3). Let's suppose to have a chunk c_k exchanged in the overlay $G_o(V_o, E_o)$ under the assumption once a peer P_j sends it to a peer P_i it immediately deletes it. Then, the PageRank centrality x_i of the peer P_i represents, in a given moment, the probability c_k is sent to P_i from one of its neighbours.

After the optimization of Theorem 7.1, the eigenvector centrality of $A^\star$ is $\frac{\vec{1}}{|V_o|}$ so that, the steady state probability of all the peers of receiving c_k is uniform. That

modifies the resource distribution Θ so that each peer P_i tunes its activity and cope with the limitations of its neighbours.

Parameter interpretation. For the optimization each peer P_j has to tune Θ_j by computing its neighbour need $\eta_j = \sum_{k=1}^{|V_o|} \frac{\tilde{a}_{kj}}{x_k} = \sum_{P_k \in N_j} \frac{\tilde{a}_{kj}}{x_k}$ which is a weighted sum (by the given neighbour probability distribution) of the inverse of the neighbour centralities; the smaller x_k, $P_k \in N_j$, the greater η_j. In general, η_j is big if the neighbours of P_j are not central and small the other way round.

Using this interpretation, the optimized parameters given in Theorem 7.1 can be rewritten as:

$$\Theta_j^\star = \eta_j x_j;\ A_{ij}^\star = \frac{\tilde{a}_{ij}}{x_i \eta_j} \tag{7.11}$$

Hence, the amount of information/resources a peer P_j has to invest in the distribution is proportional to its centrality and to its neighbour need. The probability P_j sends information to one of its neighbour $P_i \in N_j$ is inversely proportional to its centrality x_i and weighted by the originally given probability $\tilde{a}_{ij}$ (η_j is just a normalization factor in this case).

Being $\Theta_j^\star$ proportional to x_j and $a_{ij}^\star$ inversely proportional to x_i, one may say the reception-equal distribution recalls a famous socialist concept: *from each according to his ability, to each according to his need* [95].

Resource bound. From Theorem 7.1 it follows $|\Theta| = |V_o|$ and the distribution is minimal but one may be interested in verifying the support of each Θ_j, $\forall P_j \in V_o$. Resources are limited in practice and it is of interest to estimate a peer resource demand as well as possibly to find a way to fine tuning it. The first issue can be addressed with the following proposition.

Proposition 7.1. *From Theorem 7.1, it follows:*

$$\theta_j^\star \leq |N_j|$$

Proof. From Equation (7.10) and Equation (7.7) it follows:

$$\theta_j^\star = \sum_{k=1}^{|V_o|} \frac{\tilde{a}_{kj} x_j}{x_k} = \sum_{k=1}^{|V_o|} \frac{\tilde{a}_{kj} x_j}{\sum_{i=1}^{|V_o|} \tilde{a}_{ki} x_i} = \tag{7.12}$$

$$\sum_{k=1}^{|V_o|} \frac{\tilde{a}_{kj} x_j}{\tilde{a}_{kj} x_j + \sum_{i=1; i \neq j}^{|V_o|} \tilde{a}_{ki} x_i} \tag{7.13}$$

Each term in the summation is lower or equal than 1 and it is 0 if and only if $\tilde{a}_{kj} = 0$. Since $\tilde{a}_{kj} = 0 \iff z_{kj} = 0$ then $\frac{\tilde{a}_{kj}x_j}{\tilde{a}_{kj}x_j + \sum_{i \neq j} \tilde{a}_{ki}x_i} \leq z_{kj}$ and it follows:

$$\theta_j^\star \leq \sum_{k=1}^{|V_o|} z_{kj} = |N_j| \tag{7.14}$$

□

Equation (7.14) states each participating peer spends a bounded amount of resources; moreover it gives us a direct parameter to tune them. Suppose P_j has been assigned by the optimization an amount $\Theta_j^\star$ of resources to be used; if P_j cannot provide them it can simply iteratively drop one of its neighbour and re-run the optimization. The limit case in which it has to drop all of its neighbours corresponds to the scenario in which its participation to the P2P network is not sustainable at all. Preventing $G_o(V_o, E_o)$ disconnection relies on rewiring policies (see Section 3.2.1.2) and it is out of the scope of this optimization.

7.3 Decentralized optimization

Working with decentralized systems, the goal is to obtain decentralized solutions for the algorithms. Each peer $P_j \in V_o$ is requested to individually compute its optimized parameters of Equation (7.11). To this end, P_j needs its own peer probability distribution (the j-th column of $\tilde{A}$), its centrality x_j, and the centralities of its neighbours $x_i, \forall P_i \in N_j$.

In the P2P context knowing P_j probability distribution is straightforward and exchanging information, like a centrality value, among neighbours is solvable using standard gossiping protocols. However, this schemes still demands each peer P_j to compute its own centrality x_j.

To solve this problem two solutions are proposed; the first takes advantage from a recent work on computing the PageRank centrality [96] with a decentralized approach, hence solving directly the issue; this solution combines the proposed optimization with their decentralized centrality computation realized through gossiping mechanisms granted to eventually converge to actual values. The second solution is simple and elegant to implement and to realize, even if it applies to a special case, as described in Section 7.3.1.

7.3.1 Symmetric, uniform case

Let's consider a very common case in P2P distribution systems; a bidirectional, not weighted overlay $G_o(V_o, E_o)$ which means that the peer scheduling probability function is uniform. The resulting adjacency matrix A is hence symmetric and the distribution matrix $\tilde{A} = AI_d^{-1}$ is column uniform. This is the case, for example, of the PeerStreamer distribution and, being straightforward and simple, broadly used.

This case has a very nice property to be used in the reception-equal optimization which relies on the following proposition.

Proposition 7.2. *Let $G_o(V_o, E_o)$ be an unweighted, non directed graph with adjacency matrix A and $\tilde{A} = AI_d^{-1}$ $(d = A\vec{1})$ its associated distribution matrix. Then $I_x^{-1}\tilde{A}I_x = \tilde{A}^T$.*

Proof. Let z_{ij} a binary variable indicating if $\tilde{a}_{ij} = 0$, since the graph is non directed, $z_{ij} = z_{ji}$. Each element of $\tilde{A}$ is defined by $\tilde{a}_{ij} = \frac{z_{ij}}{|N_j|}$. Let N be the column made of all the values of $|N_i|$, then N is an eigenvector of A , $AN = N$.

For the uniqueness statement of the Perron-Frobenius theorem $\exists k \in \mathbb{R} : N = kx$ hence, the element (i, j) of $I_x^{-1}\tilde{A}I_x$ is given by:

$$\left\{I_x^{-1}\tilde{A}I_x\right\}_{ij} = \tilde{a}_{ij}\frac{x_j}{x_i} = \frac{z_{ij}}{|N_j|}\frac{x_j}{x_i} = \frac{z_{ij}}{kx_j}\frac{x_j}{x_i} = \tag{7.15}$$

$$\frac{z_{ij}}{kx_i} = \frac{z_{ji}}{kx_i} = \frac{z_{ji}}{|N_i|} = \tilde{a}_{ji} \tag{7.16}$$

□

Proposition 7.2 states the similarity relationship between $\tilde{A}$ and $\tilde{A}^T$ and gives its corresponding change of base matrix I_x (I_x is called a symmetrizer). While there exist a large body of work on symmetrizers, this result has never been reported in the networking literature.

The special property of this particular case of distribution process can now be stated.

Corollary 7.1. *From Theorem 7.1 and proposition 7.2, it follows that if $G_o(V_o, E_o)$ is an unweighted, undirected graph with distribution matrix $\tilde{A} = AI_d^{-1}$ $(d = A\vec{1})$ then $\Theta^\star = \tilde{A}\vec{1}$ and $\tilde{A}^\star = \tilde{A}^T I_{\Theta^\star}^{-1}$.*

Hence, peer optimized parameters can be computed as:

$$\Theta_j^\star = \eta_j x_j = \sum_{P_i \in N_j} \tilde{a}_{ji};\ A_{ij}^\star = \frac{\tilde{a}_{ij}}{x_i \eta_j} = \frac{\tilde{a}_{ji}}{\Theta_j^\star} \tag{7.17}$$

To compute Equation (7.17) a peer P_j simply needs to know $\tilde{a}_{ji}\ \forall P_i \in N_j$ which in our case is given by knowing $|N_i|\ \forall P_i \in N_j$.

Summing up, the reception-equal optimization is obtainable for this special case by simply making peers exchange their neighbour set sizes $|N_j|$ on a neighbourhood scale, easily implementable by piggybacking such numbers on the P2P topology gossiping mechanism (Section 3.2.1.2).

7.4 Performance gains

The different strategies are simulated using SSSim (Section 4.3) to fairly compare them while avoiding any interference complex real P2P systems may introduce.

7.4.1 Experiment setup

The reception-equal optimization, indicated with *R-E*, is compared against the uniform peer selection HPF and against the state-of-art distribution optimization strategy by Wu and Li [59], indicated with *Wu-Li*. Wu-Li strategy solves a centralized problem for rate allocation and assigns the transmission rates to the peers keeping the distribution minimal. It is intended to be considered as an optimization lower bound for the proposed algorithm, however the limitation on the peer chunk buffers affects the optimality of its results. Solving a centralized optimization problem it is also computationally expensive and, in the experiments, it is run only for $|V_o| \leq 200$.

The formerly introduced resource amplification factor $\alpha \in (1, \infty)$ (see Section 7.1) is varied to compare strategies when resources available to peers change; in this context it multiplies the amount of chunk offers peers send per chunk time (Section 3.2.1.3). The chunk buffer length is set to 64. Loss and delay are measured as specified in Section 3.2.2 using a synthetic random dataset composed of Erdős-Rényi and power-law Barabási-Albert graphs. Graphs are passed as input to the simulator to be used as overlays $G_o(V_o, E_o)$.

The results for different runs and data are reported with a 99% confidence interval.

7.4.2 Experiment results

Figure 7.2 reports the loss rate for sparse networks, $\frac{|E_o|}{|V_o|} \simeq 2$ (lower plot) and denser networks, $\frac{|E_o|}{|V_o|} \simeq 5$ (upper plot). Results for both Erdős-Rényi (indicated with ER) and Barabási-Albert (indicated with PL) cases are very similar, but loss is generally higher with sparse networks. In all the cases, the reception-equal strategy produces a loss rate comparable to the centralized Wu-Li one and, with the sparser networks and $\alpha \geq 1.1$, is even better. That is probably due to a buffer overloading along the shortest path imposed by the Wu-Li technique, which ends up losing packets. The gain obtained by the reception-equal optimization with respect to the plain HPF is substantial with low α.

From Figure 7.2 results that $\alpha \geq 1.5$ is enough to obtain a loss rate of almost 0 for both reception-equal and Wu-Li strategies. $\alpha = 1.5$ can hence be kept while varying the overlay size $|V_o|$ to assess the scalability. Figure 7.3 shows loss rate variation increasing $|V_o|$ while keeping the graph density constant. With HPF the loss rate grows with $|V_o|$ while with the optimized techniques remains almost 0.

It is interesting to note what happens to the chunk delay distribution. So far reception-equal results show a good loss rate and that it likely depends on chunk buffers and packet dropping. It arises the question whether these chunk arrival distribute in time. To fairly compare arrival times, experiments are run with $\alpha = 2.5$ so the

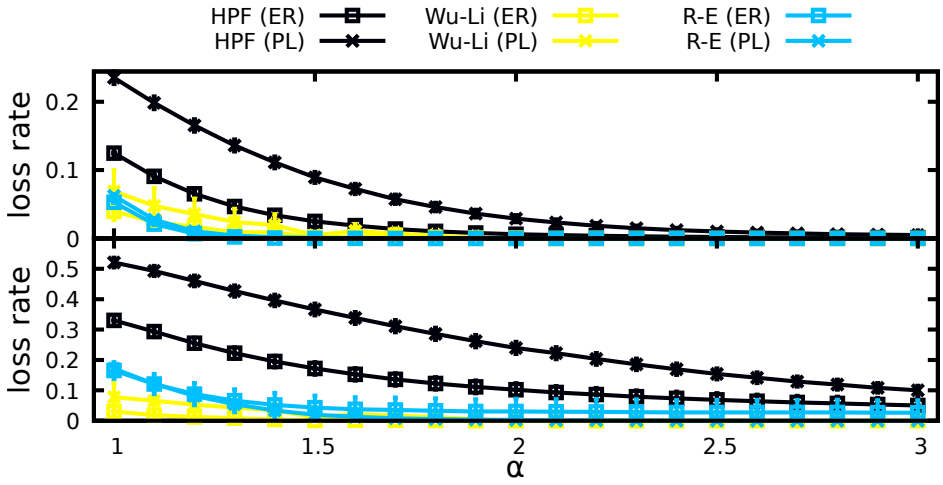


Figure 7.2 – Loss rate with 99% confidence interval as a function of α with $|V_o|$= 50; $|E_o|$= 264 (top) and $|E_o|$= 96 (bottom). Copyright © 2017 IEEE.

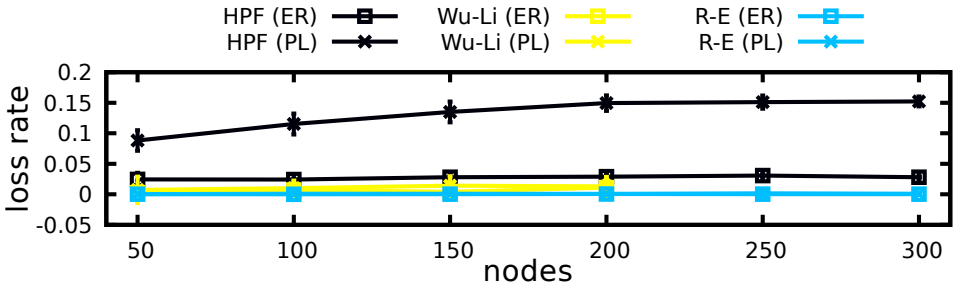


Figure 7.3 – Loss rate with 99% confidence interval as a function of $|E_o|$, $\alpha = 1.5$, $\frac{|E_o|}{|V_o|} \simeq 5$. Copyright © 2017 IEEE.

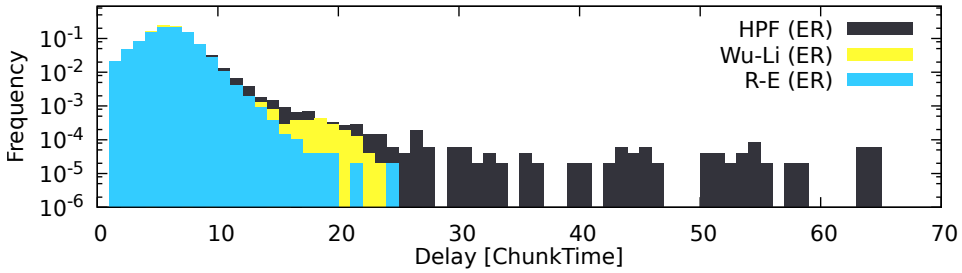


Figure 7.4 – Reception delay histogram for an Erdős-Rényi graph with $|V_o|$= 50, $|E_o|$= 264 and $\alpha = 2.5$. Copyright © 2017 IEEE.

loss ratio drops to 0 for plain HPF as well. The delay data of the run reported in Figure 7.4 indicates that HPF still has a very long distribution, meaning that less central nodes receive the chunks very late; conversely, the reception-equal strategy spreads the chunks in the network at faster pace, shrinking the delay distribution even better than the Wu-Li one.

7.5 Remarks on joint optimization with rewiring

The distribution optimization presented is independent from any overlay optimization, meaning that it can be executed either or not a rewiring process is applied. However, the two optimization techniques may have impacts on the goal of each other. In particular, in Chapter 6, a sending-equal behaviour is implicitly assumed as:

- Each overlay link is equally weighted (the $\tilde{A}_{ij}$);
- Each overlay node has fixed (target) degree (counting as Θ_i).

However, it is worth noticing if rewiring succeeds in creating the target output regular graph then applying Theorem 7.1 does not change the distribution process parameters $(\tilde{A}, \Theta)$, as all the overlay nodes have the same PageRank centrality.

In general, all the rewiring strategies presented in Chapter 6 tend to forge regular overlay graphs.

Chapter 8

PeerStreamer-ng

Despite PeerStreamer has successfully been used in live demos and tested in stressing scenarios (both in terms of network size and resource utilization) it has remained a research project, hence, suffering from the typical research code development issues (Section 8.1). Moreover, PeerStreamer and other popular P2P platforms have been designed for the internet. As introduced in Chapter 5, CNs are quite different from the internet in terms of structure and resource distribution. CN structure also offer new opportunities an application developed specifically for CNs can leverage. This is the case of micro-clouds infrastructure, currently under development in the Guifi.net network [97].

PeerStreamer-ng is a P2P live streaming platform using the latest state-of-art P2P algorithms, derived from PeerStreamer, specifically designed for CNs, in particular PeerStreamer-ng:

- Implements most of the strategies presented in Chapters 5 to 7'
- It is designed for specific CN use cases, see (Section 8.2).

PeerStreamer-ng is a fully-fledged and deployable product for CNs, realizing the transition from academic research to end-users benefits, generally called *innovation*.

The development of PeerStreamer-ng is in tight contact with the CN world and user expectations through the participation to European projects like CONFINE[1] and netCommons[2] which are specifically focused on CNs and include major European

[1] http://confine-project.eu/
[2] https://netcommons.eu

CN stakeholders. That grants a continuous feedbacks from the community users and involve them in the project, fostering a broad adoption of PeerStreamer-ng.

8.1 From PeerStreamer to PeerStreamer-ng

As introduced in Section 2.1.2, PeerStreamer is the outcome of the NAPAWINE project and it has been used as the reference platform in a large number of scientific publications working with P2P technologies [28,50,51,73,75]. Despite its success in the research community it has not gained a big momentum among actual video streaming users.

PeerStreamer was designed mainly for research purposes and, as such, it suffers from the typical research project issues:

- Un-friendly, not reliable interface;
- Several incarnations (for different user models);
- Functionalities operated by console scripts;
- Dead code;
- Functionalities not fully-integrated with the rest of the application;
- Mixed coding styles and conventions.

Those issues arise when researchers write code to test their latest algorithm but they do not really focus on its final deployments. Such contributions also come generally from a large, not organized, crowd. Moreover, PeerStreamer scope is quite broad, being a general purpose live streaming platform. Hence, no clear use case or user model have been designed to drive its development and there exist several mutually exclusive *incarnations*, meaning platforms performing the same things with different behaviours.

PeerStreamer-ng focuses on CN user cases and its design includes selecting the advanced P2P algorithms from PeerStreamer codebase and create a clean, robust and reliable live video platform.

8.2 Tailoring for WCNs

PeerStreamer-ng aims at implementing the latest technologies for P2P streaming on CNs. Its goal is to conclude the path started with initial data collection (Chapter 5),

evolved with specific research tools (Chapter 4) and new streaming optimization strategies (Chapters 6 and 7), by providing CN users a state-of-art tool for live streaming. The contact with CN users is of paramount importance to drive the development solving actual issues as well as to foster a broad adoption in the communities. In particular:

- First collaboration, founded by the OSPS project, aimed to identify and solve issues CN researchers involved in the CONFINE project were concerned about when talking about streaming;
- I personally had few meetings with the Italian Ninux.org WCN representatives;
- During the netCommons project, new insights and integration advices were collected again from the European CN research community.

Besides, this work started early to identify the target user scenarios in CNs [4].

8.2.1 Integration in Cloudy

Clommunity and netCommons projects both support Cloudy[3]. Cloudy is a GNU/Linux OS distribution with the goal of to easing the publishing and provision of network services in CNs. Its development started with the Clommunity European project in 2013 and it is now used in the Guifi.net network. Cloudy is supposed to be installed on CN user home nodes and it comes with pre-installed runnable services.

PeerStreamer-ng is developed in tight contact with the Cloudy developers, so that it has already been added as default CN service[4] and it is readily deployable on thousands of nodes.

Integrating PeerStreamer-ng with such kind of dedicated nodes eases the deployment in practice but it requires a flexible interface which can be de-coupled from the streaming engine. For this reason PeerStreamer-ng is designed to have a separated ReST interface, which can be run with any web browsing device, and a core to be installed on a specific machine.

In the Cloudy environment, services use the Serf system[5] to publish and retrieve the community services. PeerStreamer-ng comes with a streaming channel list management which supports several different retrieving modes, including channel advertisement through Serf. This way, channel list broadcasting among PeerStreamer-ng instances is performed directly with the dedicated Cloudy announcement service.

[3] http://cloudy.community
[4] http://cloudy.community/services/
[5] https://www.serf.io/

8.3 Design

Summing up, PeerStreamer-ng has been developed with the following goals:

- Polish PeerStreamer code and rewrite key components;
- Focus on specific user cases and one user interface;
- Support a full fledged friendly user interface;
- Being the most lightweight and portable as possible (to be ran on CN devices);
- Integration with Cloudy and Serf.

Figure 8.1 shows the architecture of PeerStreamer-ng. The **source** is a PeerStreamer-ng instance providing a chunk P2P flow and publishing its **channel** description through the **Serf** system. A user runs a web **browser** and communicates through HTTP

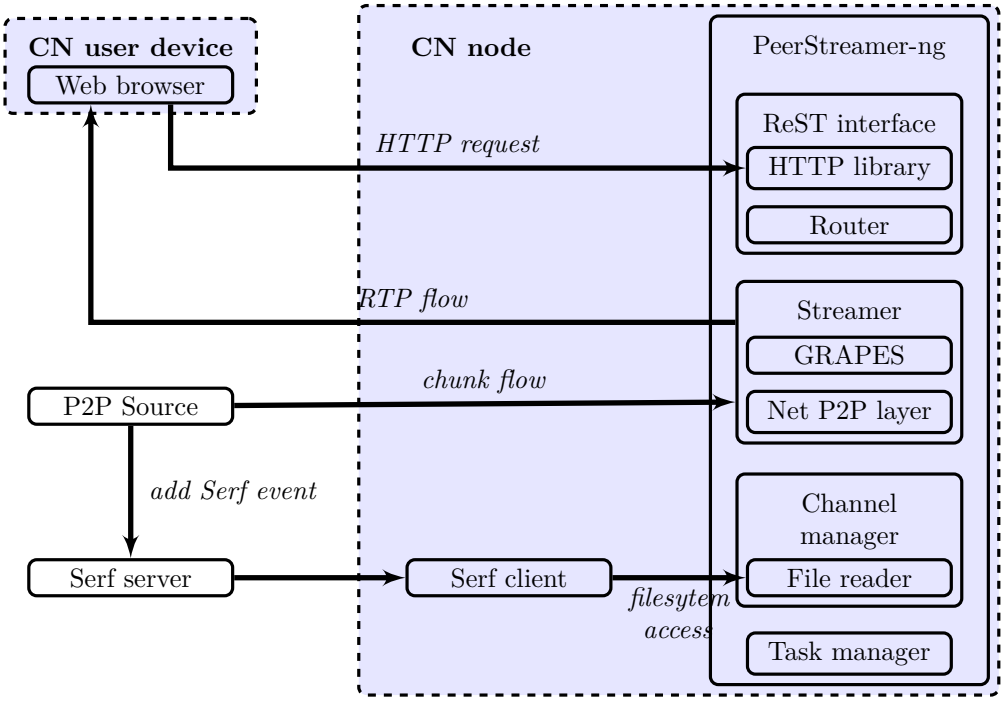


Figure 8.1 – PeerStreamer-ng architecture. Blocks represent software components which are run on different devices (light blue, dashed lines boxes). The source and the Serf server are run remotely in the network.

with PeerStreamer-ng **ReST interface**. Within this communication a user initiates a **Streamer** instance and the chunk flow is collected by PeerStreamer-ng and served as **RTP flow** back to the web browser. The user plays the RTP flow using standard browser plugins.

8.4 Module details

This section reports a deep description of the various PeerStreamer-ng modules.

8.4.1 Task manager

In PeerStreamer-ng there are many tasks that requires scheduling and a periodic execution. Examples include, Serf channel list refresh, HTTP server handling and all the P2P related periodic tasks like periodic offers, chunk injection and gossiping messages. PeerStreamer-ng is designed to be lightweight and to run on single cpu devices so all this periodic tasks are scheduled using the task manager module. The task manager offers a simple interface comprising of:

`task_manager_new_task()`, to add a new periodic task to the spool, specifying the callback function to be called when an event occurs (timeout expiration or file descriptor change of state), the reinit function to be called after timeout expiration and optional metadata to be passed to those functions;

`task_manager_poll()`, to make sleep the current process until an event occurs or the given timeout expires.

8.4.2 ReST Interface

The ReST interface is implemented on top of a HTTP library (mongoose[6]) granted to be lightweight and highly portable. The HTTP calls are first captured by mongoose and then managed by the router module. The router module provides developers with an easy interface to add and handle ReST calls:

`router_add_route()`, which add a new routing to the spool, specifying the HTTP method, the regular expression for matching the URL and the callback function to be called upon match, which takes in input the mongoose formatted HTTP message and it calls the callback function matching with the associated request;

`router_handle()`, which takes in input the mongoose formatted HTTP message and it calls the callback function matching with the associated request.

[6] https://cesanta.com/

The associated designed ReST interface is the following:

- `GET /player.html`: returns the default web application page;
- `GET /channels`: returns a JSON list of available channels;
- `POST /channels/<stream_id>?ipaddr=<sourceip>&port=<sourceport>`: creates the streaming resource <stream_id> and launches the streaming instance; it returns a JSON describing the resource attributes with which initialize the plugin player;
- `UPDATE /channels/<stream_id>`: heartbeat request, to be called frequently on <stream_id>;
- `GET /mysources.html`: returns the source html page;
- `POST /mysources/<channel_name>`: creates a new distribution overlay;
- `GET /mysources/<channel_name>`: returns the channel parameters and statistics web page;
- `DELETE /mysources/<channel_name>`: destroys the indicated distribution overlay.

8.4.3 Channel management and distribution

PeerStreamer-ng runs a periodic task for maintaining a fresh list of available streaming channels. This task can retrieve the list from different channel providers, specified through the command line. It can communicate with a local Serf instance using a text file comprising one line per channel, each line composed by the following comma separated fields: channel name, P2P source IP address and port, a quality string and a Service Description Protocol (SDP) file. The SDP file reports the metadata of the RTP flow and it is served (after modification of the source parameters) to the player in the browser.

8.4.4 Streamer

The streamer module is the one implementing the overlay and distribution management (Sections 3.2.1.2 and 3.2.1.3) and it has been the main target of this thesis work. Each time a user asks for a streaming channel, a streamer instance is created and joined to

the channel overlay. It implements the advanced strategies for P2P distribution and it takes advantage from two supporting modules, GRAPES and the network layer.

GRAPES [26] is a P2P toolkit providing the basic P2P structures and gossiping algorithms.

Network P2P layer is a standalone component managing advanced networking techniques not related to streaming such as: fragmentation, loss recovery and traffic shaping.

8.5 Showcase and impact on WCNs

During a netCommons project meeting in July 2017 there was the first showcase of PeerStreamer-ng and its integration in the Cloudy system. Figure 8.3 is a screenshot of Cloudy interface listing the available services on a demo network. Figure 8.2 shows PeerStreamer-ng interface of a running video streaming instance for a demo network. PeerStreamer-ng capabilities and details of integration are also reported in the project deliverables [98].

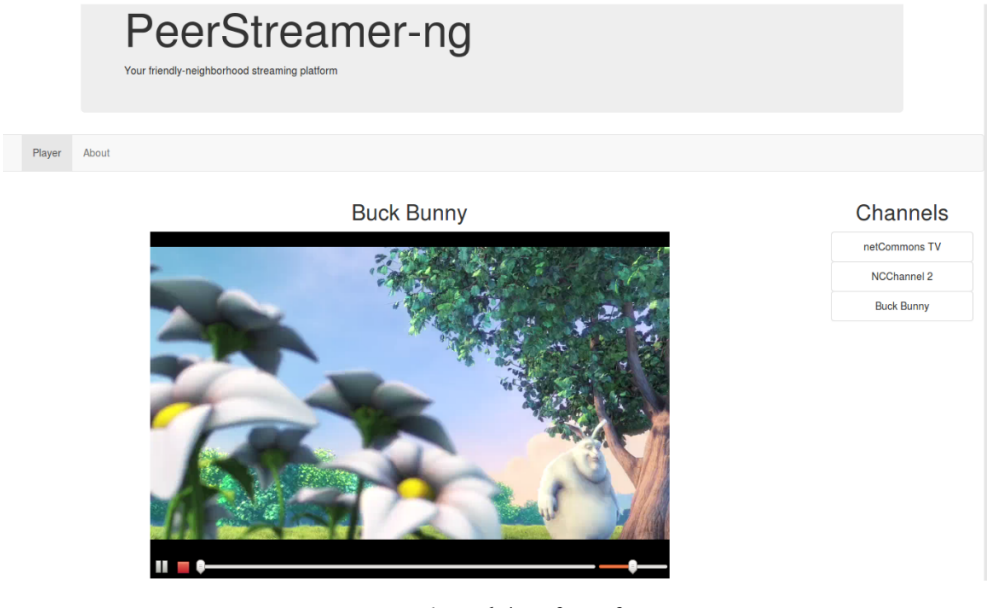


Figure 8.2 – PeerStreamer-ng web interface running during a demo.

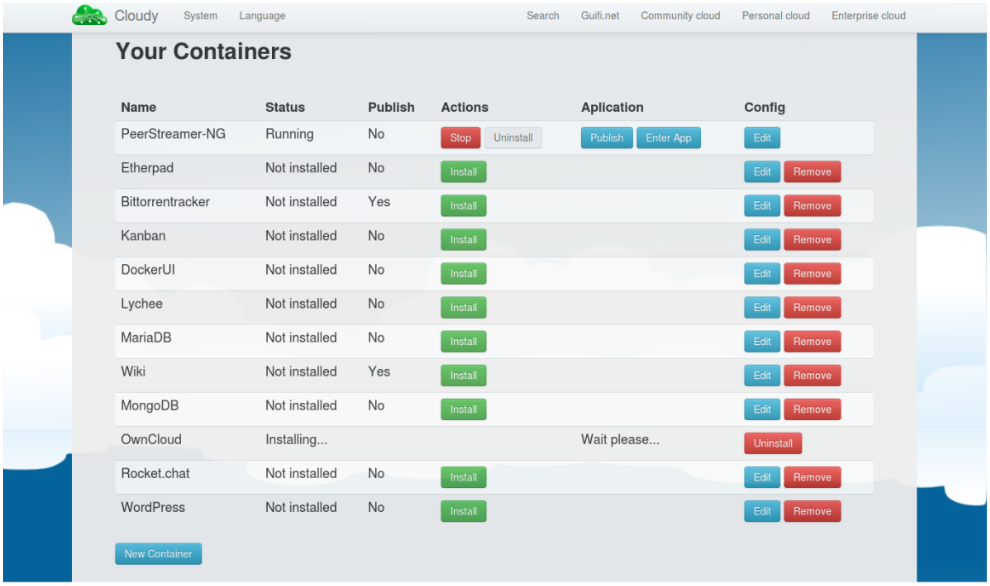


Figure 8.3 – Cloudy interface displaying PeerStreamer-ng among the available services.

Chapter 9

Spectral graph generation

This chapter deviates from the main topic of this thesis to focus on another important aspect of graph research: the generation of synthetic topologies with given properties.

Generating random graphs with certain prescribed properties is an increasingly important field [63, 99–102]. Researchers are often interested in data that can not always be fully disclosed, like social networks for privacy reasons. Synthetic graphs can be built starting from existing networks preserving the features of interest and acting as *proxies* for the real world data. Moreover, when developing communication algorithms, researchers want to assess the algorithm robustness and efficiency on a broad range of input data. This is typically not possible with real data due to lack of availability but synthetic generators can produce very large datasets of graphs sharing some common properties.

Prior work has primarly focused on generating graphs with certain *local* properties, such as degree distribution and correlation, subgraph counts, etc. [63, 99, 101–103] with little attention to global properties. In this chapter the focus is on a global property exhibited by many comunication and social networks, the *community structure*, which captures the graph substructure. In fact, in these networks, nodes tend to aggregate, forming well-connected groups while being loosely connected with the rest of the network. This *community* [61] or *cohesive subgroup* [104] structure impacts network robustness and content diffusion. The graph generators focused on local properties do not necessarly preserve such structures and the resulting synthetic networks may be poor proxies of real world graphs.

9.1 Related work

A broad literature production is available on graph generation targeting local properties. Generators use local property descriptors like graphlets [99], motifs [100] and dK-series [101, 102] to preserve those structures starting from an initial realization or a vector of input statistics. Exponential family Random Graph Models (ERGMs) can be used to produce graphs with the same expected values of a series of features of interest, but, to date, they have been mainly employed for local properties. In contrast to the proposed approach, ERGMs preserve only expected statistics and not exact property values.

There exist only few approaches so far targeting graph modularity by design. Trajanovski et al. [105] approach generates random graphs with a prescribed modularity value while the Karrer and Newman stochastic block model [106] generates random graphs preserving both the modularity value and the community structure.

The Trajanovski et al. and Karrer and Newman generators are described in the following and used as baselines in the experiments.

The tight relation between eigenstructure and community structure has been investigated for community detection [61, 107], visualization [108] and analysis [109] purposes; this work proposes a graph generation method based on that. The proposed method, called Spectal Graph Forge (SGF), is the first general framework for synthetic graph generation by means of eigenstructure inputs.

9.2 Graph spectral structures

Modularity, as defined in Section 3.1.1, is based on the modularity matrix B of a graph and, hence, on its adjacency matrix A. A completely defines a simple graph and its eigenstructure is intimately related to its structure. As recalled in Section 3.1.4, A can be decomposed in a set of real eigenvalues λ_i and their associated eigenvectors v_i; the eigenvectors related to positive eigenvalues describe *core-periphery* structure [110] and the sign of their entries indicate good graph bi-partition. Conversely, the eigenvectors related to negative eigenvalues describe memberships in *bi-partitions* and the sign of their entries indicate node connection likelihoods.

This inner relationships between eigenstructure and the community description can be exploited to handle the manipulation of the latter in an automated fashion. Since the leading eigenvector (the one associated to the largest eigenvalue) is associated to the main connected component (the whole graph), it comes naturally to exclude it

from the analysis and to consider just $A - \lambda_1 v_1 v_1^T$. In general, v_1 is parallel to the degree vector d (they correlate) and, hence, the modularity matrix $B \simeq A - \lambda_1 v_1 v_1^T$ (see Section 3.1.1).

That leads to the common practice for community detection, considering the eigenstructure of B for splitting a graph according to the leading eigenvector entry signs. In particular, Newman showed that modularity is heuristically maximized when bi-partitioning a graph according to the leading eigenvector entries of B [107].

While this chapter focus primarily on graph communities and modularity, the spectral decomposition framework SGF is more general as the algorithms can be tuned to target specific types of structures of interest while randomizing the other graph properties. Specifically, the spectral-based graph generation is envisioned possible starting from different graph representations other than the modularity matrix, like the adjacency matrix or the clique co-membership matrix.

9.3 Spectral Graph Forge

The proposed approach, the SGF, works by dissecting a graph symmetric matrix derived from its adjacency matrix in its spectral components and deriving a class of random graphs sharing the same spectral properties, with a certain level of accuracy.

Formally, the procedure of the SGF is depicted in Figure 9.1. The input is a graph $G(V, E)$ represented through its adjacency matrix A and the output is a different adjacency matrix A' representing the output graph $G'(V', E')$ indicated with SGF(α).

The parameter $\alpha \in (0, 1)$ tunes the level of accuracy in targeting the original spectral properties; if $\alpha \simeq 0$ then SGF(α) is preserving only the highest level structures while, as α approaches 1, SGF(α) preservers the finest structure details. Its role is central in the low-rank approximation step and it is hence detailed in Section 9.3.2.

The SGF has been designed with flexibility in mind, ensuring the maximum customizability in experimenting with different solutions. Its pipeline blocks are independent and researchers can fine tune the components to best fit their needs. In the following their roles are presented in details.

9.3.1 Transformation

The transformation block applies the symmetry preserving real function of choice to A. As already mentioned such function includes identity, for working with A, modularity

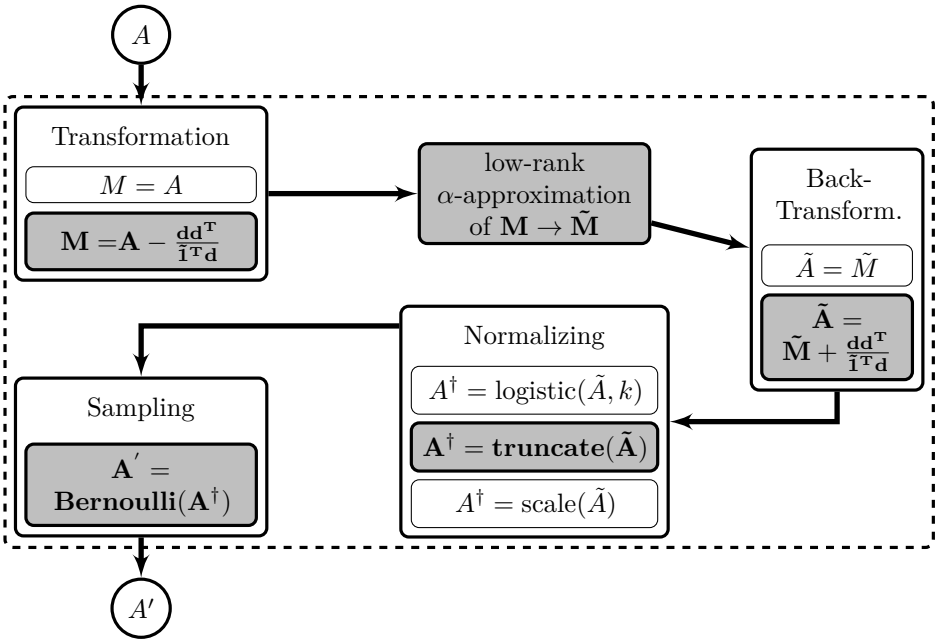


Figure 9.1 – The pipeline of the SGF framework. Given a simple graph adjacency matrix A as input, SGF outputs a "similar" one $A^{'}$ from which the corresponding graph can be built, called SGF(α). Sub-blocks indicate mutually exclusive options for each step. The main focus of the experiments is on using SGF to target modularity, by setting $M = B = A - \left(dd^T\right)/\vec{1}d)$, and the corresponding blocks are highlighted in grey. Copyright © 2018 IEEE.

transformation, for dealing with B. The transformation must produce a symmetric real matrix M in order to be used as input to the next block.

9.3.2 Low-rank α approximation

This block applies the low-rank approximation to the transformed matrix M as presented in Section 3.1.4 but driven by the parameter α, so that the output is

$$\tilde{M} = \sum_{i=1}^{\lceil \alpha |V| \rceil} \lambda_i v_i v_i^T, \ \alpha \in (0, 1)$$

Where $\lambda_1, \ldots, \lambda_{|V|}, v_1, \ldots, v_{|V|}$ are respectively the eigenvalues and the eigenvectors of M.

The distance between M and $\tilde{M}$ relates to the amount of information discarded during the approximation phase. This distance is measured using the Euclidean norm $\| M - \tilde{M} \|_2$, a well-known metric in the space of real matrices. It turns out such distance is readily computable by considering the spectral radius $\rho(\cdot)$ interpretation of the Euclidean norm of the difference matrix:

$$\| M - \tilde{M} \|_2 = \| \sum_{i=\lceil \alpha |V| \rceil + 1}^{|V|} \lambda_i v_i v_i^T \|_2 = \rho \left(\sum_{i=\lceil \alpha |V| \rceil + 1}^{|V|} \lambda_i v_i v_i^T \right) = \lambda_{\lceil \alpha |V| \rceil + 1} \qquad (9.1)$$

as the ordering discussed in Section 3.1.4 over eigenvalues and eigenvectors is adopted, such that $|\lambda_i| > |\lambda_j| \ \forall i < j$.

Equation (9.1) describes the approximation error convergence for $\alpha \to 1$ and it gives a readily method to estimate the resulting adjacency matrix drift from the original one.

9.3.3 Back-Transformation

This block takes $\tilde{M}$ as input and it applies to it the inverse of the transformation function applied in the block of Section 9.3.1. In the case of the modularity transformation this produces $\tilde{A} = \tilde{M} + \frac{dd^T}{\vec{1}^T d}$. Note that $\tilde{A} \in \mathbb{R}^{|V| \times |V|}$ is not an adjacency matrix as its components are real numbers not necessarily in $\{0, 1\}$. In order to obtain the output adjacency matrix A' two steps are still required.

9.3.4 Normalization

This block takes $\tilde{A}$ as input and normalizes its entries in $[0, 1]$, producing the block output matrix $A^\dagger \in [0, 1]^{|V| \times |V|}$. This is a necessary step in order to apply the stochastic function of the sampling block (Section 9.3.5) and hence introduce randomness in the graph generation process.

The normalization step is performed using a normalization function, formally a function $(\cdot) : \mathbb{R} \to [0, 1]$ to be applied element-wise to $\tilde{A}$; several solutions are applicable and three different techniques are proposed:

- $\text{logistic}(\tilde{A}_{ij}) = \frac{1}{1 + e^{(0.5 - \tilde{A}_{ij}) k}}, \ k \in [2, 10]$, where k is the parameter used to tune the function inclination.
- $\text{truncate}(\tilde{A}_{ij}) = \begin{cases} 1 & \text{if } \tilde{A}_{ij} > 1 \\ 0 & \text{if } \tilde{A}_{ij} < 0 \\ \tilde{A}_{ij} & \text{otherwise} \end{cases}$

- $\text{scale}(\tilde{A}_{ij}) = \frac{\tilde{A}_{ij} - \min_{s,t} \tilde{A}_{st}}{\max_{s,t} \tilde{A}_{st} - \min_{s,t} \tilde{A}_{st}}$ which equalizes the values preserving their proportions.

Figure 9.2 shows a comparison of the above functions in the interval $[-0.5, 1.5]$. Preliminary results have shown $k = 6$ gives the best performance for the applications of interest and it is used in the following.

For assessing the normalization function performance the spectral radius distance of Equation (9.1) is considered.

Figure 9.3 presents the Euclidean distances between A and $A^{\dagger}$ using different normalization techniques. Ideally, the normalization function should introduce the smallest variation, thus granting $\| A - A^{\dagger} \|_2 \simeq \| A - \tilde{A} \|_2$. The converging property on the trend $\| A - A^{\dagger} \|_2 \to 0$ as $\alpha \to 1$ is also of interest. In this and subsequent figures, a red dashed line represents the ideal target. Data shown in Figure 9.3 elects truncation as the best for this task.

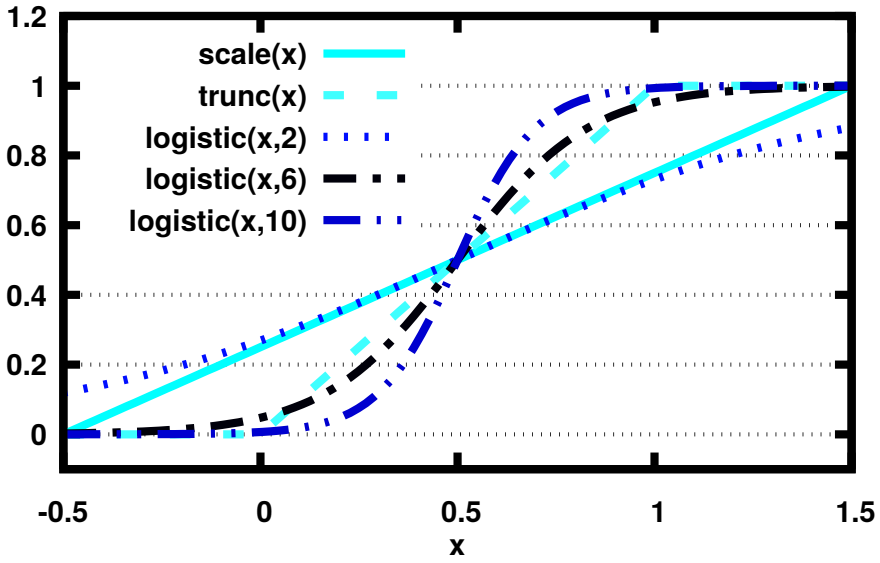


Figure 9.2 – Visualization of the presented normalization function in the interval $[-0.5, 1, 5]$.

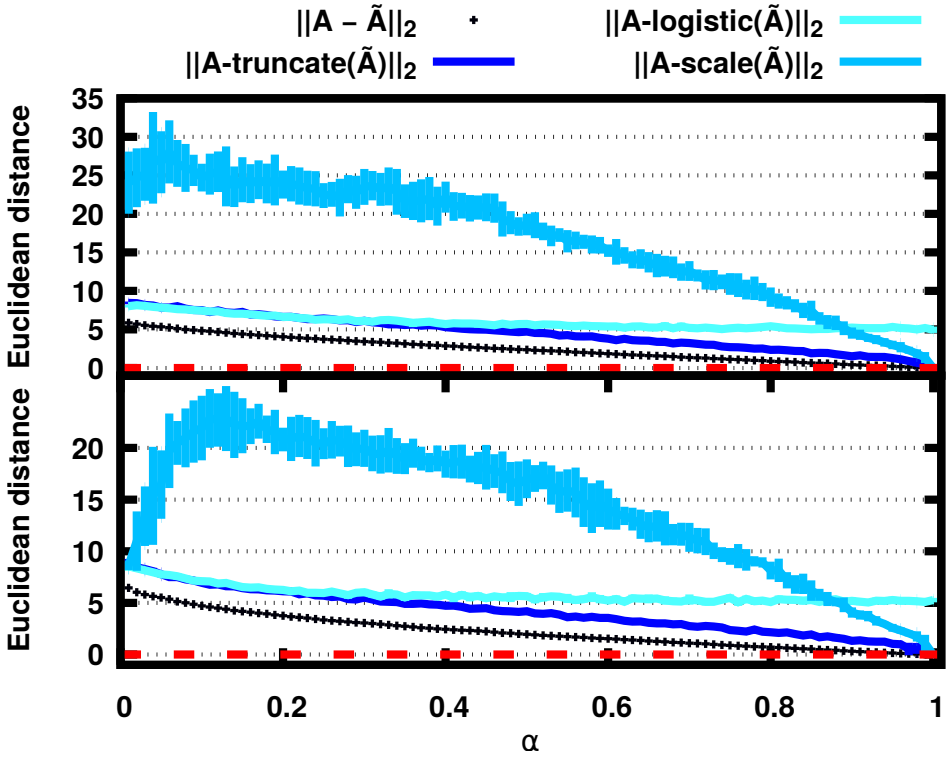


Figure 9.3 – Euclidean distance of $A^\dagger$ with respect to A using different normalization functions on Erdős-Rényi graphs (upper plot) and Barabási-Albert graphs (lower plot). Data is reported as 99% confidence intervals on different graphs. Copyright © 2018 IEEE.

9.3.5 Sampling

In this step a stochastic function to $A^\dagger$ is applied to introduce randomness and to create an adjacency matrix A'. In the current setup, the Bernoulli sampling applied element-wise to $A^\dagger$.

To preserve the simple graph properties, the elements on the diagonal are set to zero, the elements on the upper-right triangular portion are sampled while the remaining elements are specular.

- $A'_{i,i} = 0,\ \forall 1 \leq i \leq n$
- $A'_{j,i} = A'_{i,j},\ \forall j > i$

$A' \in \{0,1\}^{|V|\times|V|}$ is hence granted to be a random adjacency matrix of simple graph built from the original graph spectral components.

A' elements are readily sampled element-wise from the Bernoulli distributions with parameters $A^{\dagger}_{ij}$. This means the Shannon entropy of $A^{\dagger}$ family of matrices is easily computable. This measure describes how broad and skewed is the range of the possible output matrices A'; high entropy means very different output matrices are possible while low entropy reduces the chances of high variations. The entropy of the stochastic variable $\mathbf{A'}$ is indicated as

$$H(A^{\dagger}) = \frac{1}{S}\sum_{i=1}^{|V|}\sum_{j=i+1}^{|V|} -A^{\dagger}_{i,j}\log_2[A^{\dagger}_{i,j}] - (1-A^{\dagger}_{i,j})\log_2[1-A^{\dagger}_{i,j}], \tag{9.2}$$

which is readily computed from $A^{\dagger}$, where $S = \frac{-|V|(|V|-1)}{2}(\log_2\delta + \log_2(1-\delta))$ is the normalization factor and $\delta = \frac{2}{|V|^2-|V|}\sum_{i,j}A^{\dagger}_{i,j}$ is the expected graph density described by $A^{\dagger}$.

The possibility of computing the Shannon entropy over $\mathbf{A'}$ differentiates SGF(α) from all the other synthetic graph generators presented and it gives researchers a powerful tool for early test evaluation and theoretical a-priori estimations.

9.4 Related algorithms and test datasets

In this section the comparison algorithms introduced in Section 9.1 are presented as well as the datasets of interest used to make the experiments with.

9.4.1 Trajanovski et al. algorithm

Trajanovski et al. [105] present an algorithm for generating random graphs with a prescribed modularity which takes $m^*, Q^*, |E|$ as input (see Section 3.1.1). It first creates a graph with m^* well intra-connected communities $\Pi_1, \ldots, \Pi_{m^*}$ linked one another in a chain, thus obtaining the maximum modularity for the inputs $|E|, m^*$. It then performs three different rewiring techniques so to lower the modularity and targeting Q^*. However, since at each step the modularity variation is computed on the initial $\pi_1, \ldots, \pi_{|V|}$ and not on the optimal partitioning $\pi^*_1, \ldots, \pi^*_{|V|}$, the output graph maximum modularity as described in Section 3.1.1, can be very different from the target one Q^* given in input. In the following this approach is referred with the name *Trajanovski*.

9.4.2 DC-SBM

Stochastic Block Model (SBM) is a generative model for graphs preserving community structures [111]. Given the set of nodes V it places an edge (i, j) between nodes $i, j \in V$ with a probability which is function of $\pi_i, \pi_j \in \{\Pi_1, \ldots, \Pi_m\}$. These probabilities form a matrix of inter and intra community (block) probabilities. Generally SBMs can hardly reproduce real-world graph community structure [106] as they do not consider other form of properties often found in real networks, e.g., degree heterogeneity. The approach proposed by Karrer and Newman [106], Degree Corrected - Stochastic Block Model (DC-SBM) corrects this limitation yielding better performance on realistic topologies. In the following this approach is referred with the name *DC-SBM*.

9.4.3 Datasets

In the evaluation, synthetic datasets, specifically designed to deal with community structure research, and real-world dataset with relevant community properties, are used. The last category is very broad and includes all the networks involving interpersonal relationship information.

Synthetic. A synthetic graph generator which controls the modularity can be sketched as following; first create the node communities, secondly connect the communities and thirdly, keeping fixed the inter-community amount of edges, variate the inter-community edges until the desired modularity value is reached. This simple idea has been widely used since the beginning of community detection research; for example the Girvan and Newman model [112] creates graphs with 128 nodes, communities of the same size and almost constant node degree. Lancichinetti et. al generator [113] extends such algorithm allowing arbitral number of nodes and distributions for node degree and community size. Both generators are employed to create two datasets each comprising of ten randomly generated networks, called *Girvan* and *Lancichinetti* respectively in the remainder of this work.

Add-Health. The National Longitudinal Study of Adolescent to Adult Health (referred to as Add-Health in the following) is a U.S. national study on adolescents in grades 7-12. It is a very large study involving data collection spanning from 1994 to 2008 [114]. In the following, the dataset belonging to the first of the four waves is used, pertaining student friendship relationships. This data comprises 16 networks, each of which derived from students of a different schools, each node represents a student and edges represent friendship relationships. For each student, it is provided

the related data on gender, race/ethnicity, and grade which naturally correlates with the friendship partitioning.

Facebook. Ego networks are friendship networks created by considering all the friends of a subject and the friendship relationships among such friends (the subject node is not included). These networks form small graphs at the foundation of large-scale network data. In this work the Facebook ego network dataset by Mcauley and Leskovec [115] is employed, comprising of ten different networks with node numbers spanning from 52 to 1034.

9.5 Results

In this section the experiment results focusing on the modularity matrix B are reported (the dark gray block selection in Figure 9.1). An evaluation of specific SGF characteristics is given first, followed by a comparison of SGF(α) against Trajanovski and DC-SBM algorithms both considering direct community aspects as well as ancillary graph properties. Finally an evaluation of the potentiality of the SGF approach in terms of randomness capabilities is reported. When evaluating graph properties the ratio with respect to the input graph is computed; i.e., focusing on a graph property p for which the input graph has value p_i and the output graph has value p_o it is reported the property ratio $\frac{p_o}{p_i}$.

9.5.1 Insights on SGF

Let's first consider trivial graphs of 128 nodes and 2 or 8 communities. The maximum modularity ratio $\frac{Q_o^*}{Q_i^*}$ is measured and Figure 9.4 reports the results varying α. As expected the ratio approaches 1 as $\alpha \to 1$ but the speed of convergence varies with respect to the number of communities, the greater the number the slower the convergence. Figure 9.5 shows that the convergence speed also depends on Q_i^*; for this experiment in fact, the maximum modularity is controlled indirectly by varying the intra-community edge probability (0 for no intra community edges, 1 for communities as complete graphs). Again, the higher the input modularity value the better SGF(α) targets it.

9.5.2 Comparison against baselines

Let's now compare SGF(α) with the other approaches, still measuring the maximum modularity ratio $\frac{Q_o^*}{Q_i^*}$. All the results for all the datasets are reported in Table 9.1 while

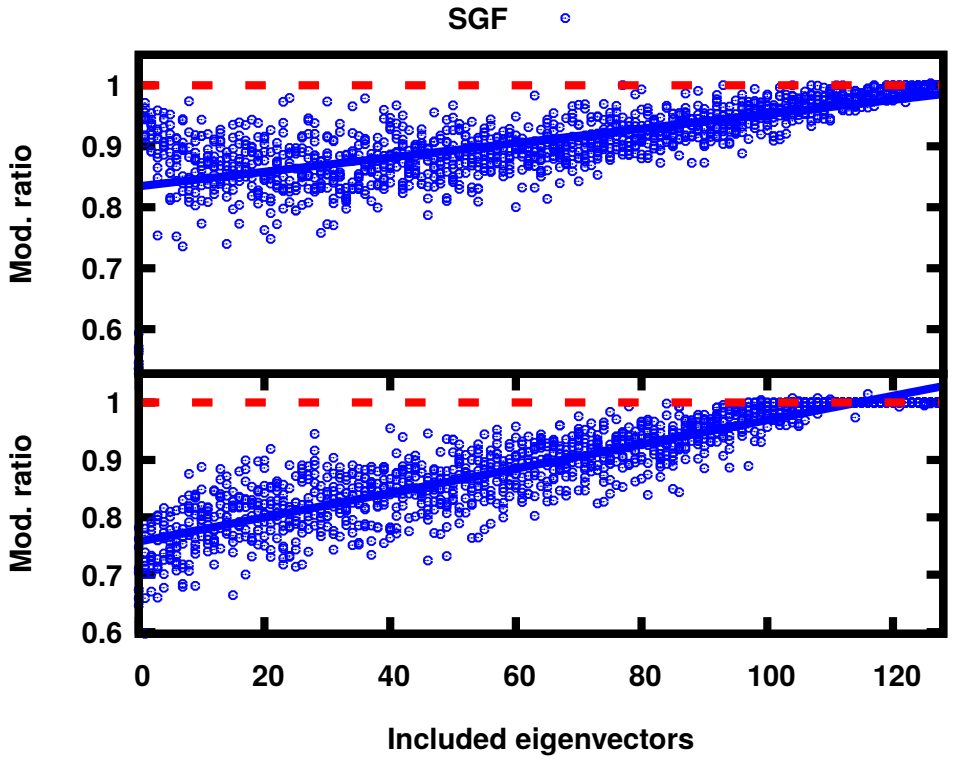


Figure 9.4 – Modularity ratio for SGF varying α on simple graphs with 2 communities (upper plot) and 8 communities (lower plot). Copyright © 2018 IEEE.

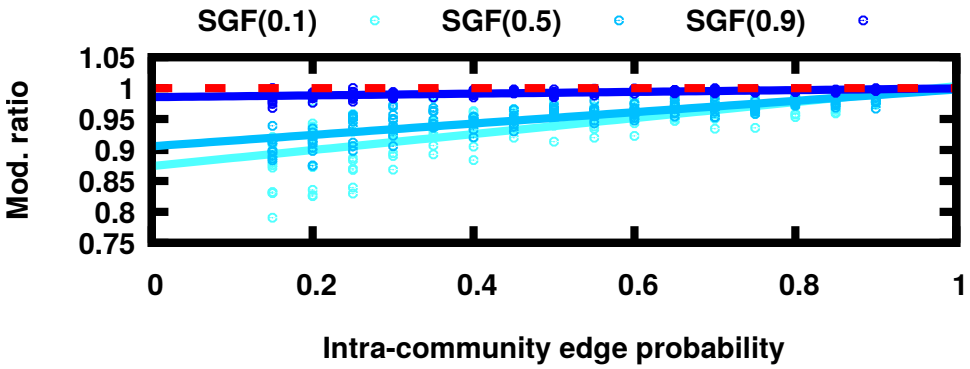


Figure 9.5 – Modularity ratio for SGF on simple graphs varying their connectivity with 2 communities. Copyright © 2018 IEEE.

Figure 9.6 show them graphically for $\alpha = 0.9$ for convenience. From Table 9.1 it can be seen SGF requires $\alpha \sim 0.7, 0.8$ to obtain results comparable to DC-SBM and that Trajanovski method results have a very high standard deviation.

As shown in Figure 9.6, $\alpha = 0.9$ grants SGF(α) to obtain always (within a 99% confidence interval) results very close to the target (ratio approx. 1) while Trajanovski always obtains worst results with a higher variance. DC-SBM performs better than Trajanovski but is farther than SGF(α) to the target and it has generally an higher variance. SGF(α) performs consistently across all the datasets under test.

Figure 9.7 tracks a different but related property, the number of detected partitions. As before it is reported the ratio between the measure on the output graph with respect to the input one so the target value is still 1. SGF(α) correctly reproduces the number of desired partitions for all the datasets but the Add-Health one for which it tends to split the communities. In the Add-Health case none of the solutions targets perfectly the number of partition with DC-SBM being the closest one.

9.5.3 Ancillary metrics

In addition to the targeted modularity property, the algorithms are tested whether they also preserve other important feature of the input graph, namely average clustering and degree sequence (see Section 3.1.1).

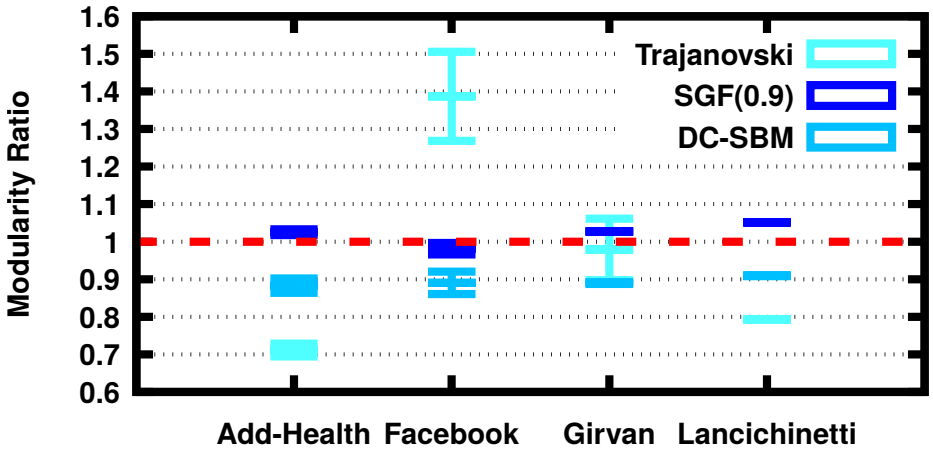


Figure 9.6 – Mean and 99% confidence interval for the modularity ratio, by method and data set. Copyright © 2018 IEEE.

Strategy	Dataset	Mean	Std	Dataset	Mean	Std
SGF(0.1)	Add-Health	0.76038	0.10479	Facebook	0.80551	0.07143
SGF(0.1)	Girvan	0.74602	0.01709	Lancichinetti	0.58268	0.00799
SGF(0.2)	Add-Health	0.71985	0.06740	Facebook	0.82773	0.08925
SGF(0.2)	Girvan	0.70728	0.01033	Lancichinetti	0.54349	0.00576
SGF(0.3)	Add-Health	0.75379	0.07496	Facebook	0.87341	0.08013
SGF(0.3)	Girvan	0.70690	0.01553	Lancichinetti	0.55989	0.00530
SGF(0.4)	Add-Health	0.80250	0.10278	Facebook	0.86587	0.06847
SGF(0.4)	Girvan	0.73735	0.00793	Lancichinetti	0.59954	0.00445
SGF(0.5)	Add-Health	0.82452	0.08711	Facebook	0.89184	0.06929
SGF(0.5)	Girvan	0.77007	0.01134	Lancichinetti	0.64635	0.00339
SGF(0.6)	Add-Health	0.86057	0.08472	Facebook	0.91149	0.06688
SGF(0.6)	Girvan	0.83302	0.01151	Lancichinetti	0.71476	0.00564
SGF(0.7)	Add-Health	0.88024	0.05368	Facebook	0.93420	0.07223
SGF(0.7)	Girvan	0.88050	0.01007	Lancichinetti	0.78831	0.00714
SGF(0.8)	Add-Health	0.93563	0.03820	Facebook	0.96249	0.02018
SGF(0.8)	Girvan	0.92902	0.00797	Lancichinetti	0.87126	0.00553
SGF(0.9)	Add-Health	1.02524	0.03708	Facebook	0.98131	0.05789
SGF(0.9)	Girvan	1.02722	0.00429	Lancichinetti	1.05135	0.00338
DC-SBM	Add-Health	0.88304	0.09229	Facebook	0.89041	0.11375
DC-SBM	Girvan	0.88989	0.00892	Lancichinetti	0.90944	0.00462
Trajanovski	Add-Health	0.71150	0.08293	Facebook	1.38715	0.45120
Trajanovski	Girvan	0.97946	0.31149	Lancichinetti	0.79338	0.00097

Table 9.1 – Modularity ratio for all the strategies on all the datasets.

Clustering. Average clustering is a measure of how much graph nodes tend to form cliques (triadic closures) and it is measured locally at each node and then averaged over all the graph. Despite SGF(α) does not explicitly target this local property it does well across all the datasets (plot shows the ratio between the output and input average clustering) while the other approaches tend to alter the triad structures as shown in Figure 9.8.

Degree sequence. Figure 9.9 reports the degree sequence correlation between the input graph and the output ones. While the eigenvectors of B are poorly connected to the input graph degree sequence (which instead correlates with the input adjacency matrix A) SGF(α) still performs better than the comparing algorithms. This is particularly surprising as DC-SBM actively attempts to preserve the degree distribution. However, it obtains results very close to the ones of SGF(α).

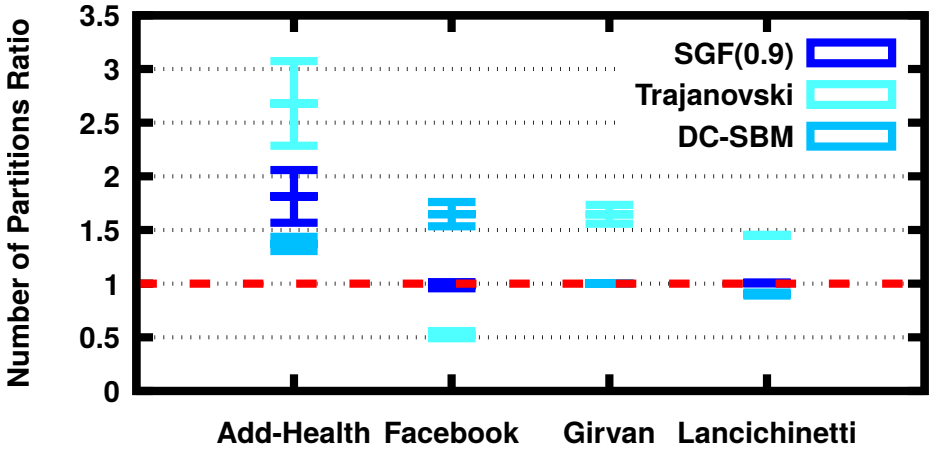


Figure 9.7 – Mean and 99% confidence interval for the partition number ratio, by data set. Note that the SGF(0.9) and DC-SBM algorithm results overlap with the Girvan data set graphs. Copyright © 2018 IEEE.

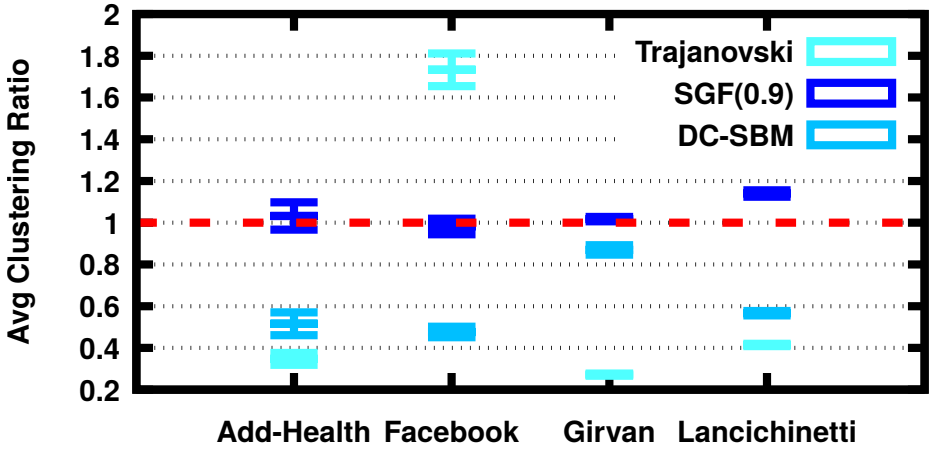


Figure 9.8 – Mean and 99% confidence interval for average clustering ratio, by data set. Copyright © 2018 IEEE.

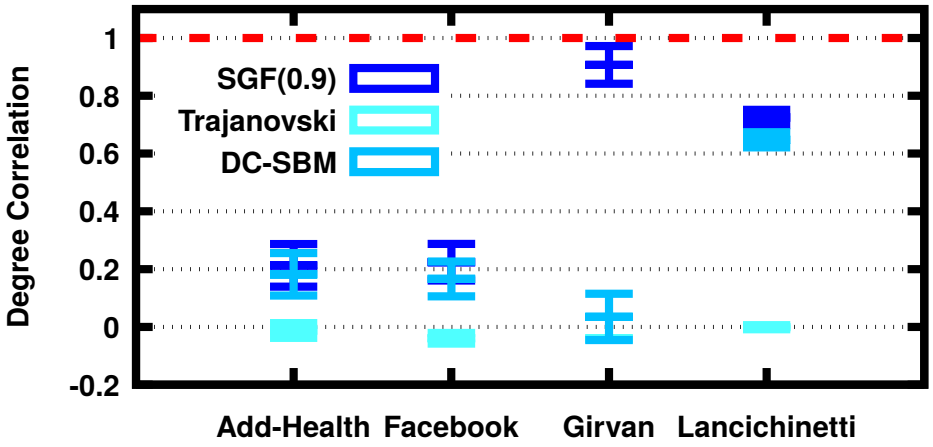


Figure 9.9 – Mean and 99% confidence interval for degree sequence correlation, by data set. Note that the DC-SBM result lower bound overlaps the Trajanovski results with the Girvan dataset. Copyright © 2018 IEEE.

9.5.4 Attribute modularity preservation

In this section one of the most interesting results of SGF are highlighted. The proposed approach not only successfully preserves the topological modularity described by Q^*, m^* but it also preserves other types of modularities which are based on node attributes and not related to the topology.

Considering a partition $\pi_1, \ldots, \pi_{|V|} \in \{\Pi_1, \ldots, \Pi_m\}$ on ethnicity, grade or gender, it is shown SGF(α) correctly preserves it while the other approaches have no way to keep such structure. The Add-Health dataset comes with such node attribute information, enabling the computation of the input ethnicity/grade/gender modularity value Q_i; after generating the graphs, that computation is repeated by labelling the nodes according to the input counter part and the resulting modularity ratios are reported in Figure 9.10.

As expected, Trajanovski and DC-SBM cannot reproduce these community structures while SGF(α) correctly preserves them even when they are complex and overlapping as in this case. This is the first method capable of creating efficient graph proxies with such complex community structures.

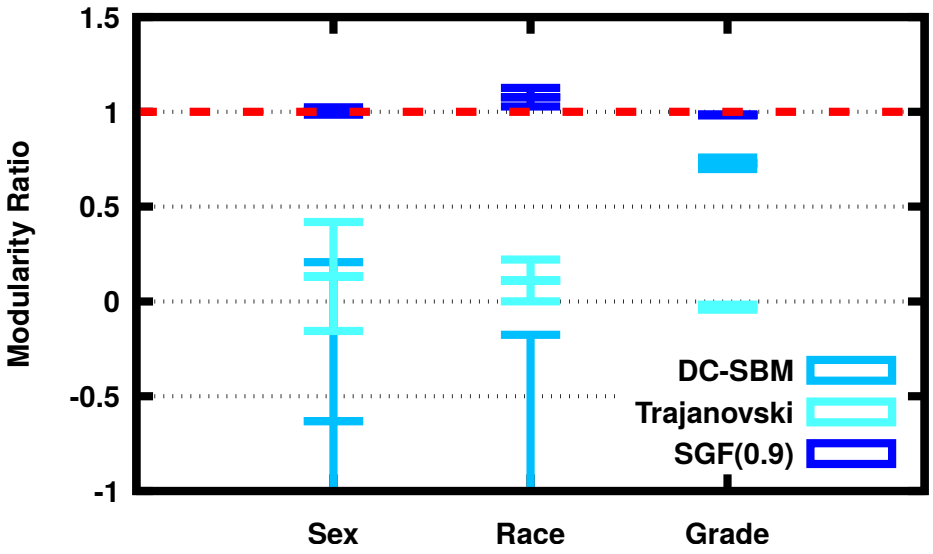


Figure 9.10 – 99% confidence interval on attribute modularity ratio for the Add-Health dataset. Copyright © 2018 IEEE.

9.5.5 On the randomness of SGF

In the previous section the performance of SGF(α) were described in terms of property targeting and that the closer α is to 1, the better the results are. However, two natural questions arise from such scenario:

- how concentrated (skewed) is the distribution of $\mathbf{A}'$? I.e., how likely are two subsequent generated graph to be similar?
- as $\alpha \to 1$, how similar is A' with respect to A?

Ideally, it is preferred a low level of concentration, so likely having distinct realizations and a reasonable distance from the input graph so that to grant a certain level of *randomness*.

In this work experiments, the Shannon entropy is used to measure the level of distribution concentration. To measure the distinctness with respect to the input graph, a pragmatic method is employed, mapping to a potential use case scenario, the resistance to de-anonymization attacks. In fact, SGF(α) is envisioned to be used to anonymize graphs with sensitive data while preserving other important structural features.

De-anonymization attacks attempt to identify nodes in a partially labelled graph by exploiting structural similarities, meaning that, if the identification fails the two graphs are topologically distinct. There is a broad literature production over graph de-anonymization attacks [116] and for this work experiments Distance Vector (DV) [116] is picked as it is proven to be robust, scalable and exploitative of the graph global characteristics [117] targeted by SGF. In the experiments, for each run, the DV attack is fed with 5% of output graph node identities as ground truth and the fraction of nodes successfully identified is reported.

The upper plot of Figure 9.11 refers to one connected graph from the Add-Health dataset. Even for low values of α the modularity ratio is close to 1 while the entropy, being close to 0 for $\alpha \to 1$, climbs steadily as α decreases. Interestingly, even for $\alpha = 0.9$ the DV success ratio is only around 60%; performance improves further for

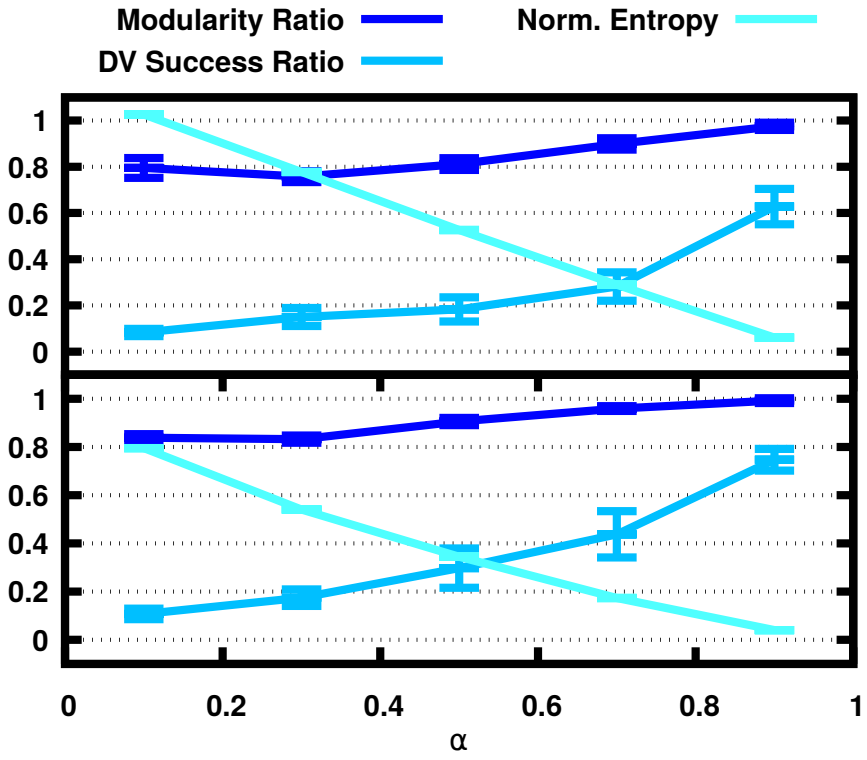


Figure 9.11 – Mean and 99% confidence interval for graph distribution entropy and DV de-anonymization success rates on one Add-Health graph (upper plot) and one Facebook graph (lower plot), by α.

$\alpha = 0.5$ granting a modularity value of 0.82 (according to Table 9.1) and a percentage of disclosed nodes of 20%. Given the strength of the DV attack, that means the output graphs significantly differ from the input one. At the same time the value of entropy grants low level of concentration for the output graphs.

The lower plot of Figure 9.11 reports similar results for a network from the Facebook dataset. In both cases for $\alpha = 0.1$ the de-anonymization attack can disclose only 10% of node identities (inclusive of the initial 5% of ground truth) while obtaining a modularity ratio around 0.8.

Chapter 10

Conclusions and long-term vision

Live streaming is a killer application for our communication networks and the trend is indicating its momentum is even increasing in the close future [12]. At the same time, CNs are an emerging and promising network scenario involving more and more people worldwide, thus spreading the adoption of mesh networks for large communication systems.

In this work, the challenges live streaming on CNs presents are shown and several strategies optimizing the efficiency and effectiveness of the distribution are derived. A specific framework is proposed to work in this context so to create new research directions and draw attention on the upcoming related challenges.

The derived streaming platform, PeerStreamer-ng, has already been integrated in a CN service system so to be readily deployed and used by CN users. This work goals, originally stated in Chapter 1, of:

- Rethinking killer applications for CNs;
- Modeling and deriving specific solutions in this context;
- Creating suitable streaming platform for CNs;
- Get CNs engaged in the process;

have been addressed and reached. Given the strong synergy between the CN philosophy, preferring distributed solutions and free data access, and the P2P architecture, relying on strong node cooperation and join effort orchestration, I envision this scenario to become a promising field for researchers; not only focusing on streaming but also rethinking entirely our way to communicate over networks.

Along the way, the following platforms have been designed and proposed to the community:

- NePA TesT, a mesh network emulator designed with WCNs in mind;
- PeerStreamer-ng, a live P2P streaming platform for mesh networks.

I see the adoption of PeerStreamer-ng as a first step in the *technologically open*, *freely accessible* and *end-user oriented* world that CNs represent.

The main scientific results of this work can be summed with:

- Cross-layer topology optimization (Chapter 6);
- Reception-equal distribution optimization (Chapter 7);
- Spectal Graph Forge (SGF) (Chapter 9).

Each of them represent a novel approach to solve a broad range of similar problems and they might also be combined to explore new solutions for issues not yet considered.

Cross-layer topology optimization can be applied in the context of *virtual network embedding* [118], allowing the optimization of virtual network allocation with respect to the actual underlay network; reception-equal optimization can be easily implemented in a broad range of existing decentralized networks with applications in information gathering in sensor networks, pertaining countryside pollution, weather conditions and vehicle traffic monitoring. Given the reception-equal optimization adaptability it can be used in contexts different from communication networks; the optimized sending rates can be though to improve systems with concurrent flow regulation, like vehicle traffic engineering. Roads and streets naturally map to a graph where concurrent vehicles compete to use the available resources and infrastructure devices, such as traffic lights represented through the graph nodes, regulate resource access. The reception-equal results might be applied in this context by imposing the optimized traffic light switching rates so to reduce the time vehicles spend in the roads (the time/delay to reach the destination). As already mentioned in Chapter 9, SGF shows interesting properties for anonymization applications and its applicability in this context has just started. Graph anonymization is becoming more and more important because of the growing size of the sensitive data daily produced by social network users.

While each of the aforementioned thesis outcomes can be exploited singularly, I also envision their combinations will be of interest in the next future. It is straightforward from this work that cross-layer topology optimization and reception-equal distribution

optimization can be combined for P2P live streaming applications. In the context of social networks, the reception-equal optimization can find the optimized parameters for spreading information (such as fake news) in the context of social networks.

Summing up, future directions rely on applying and extending the main outcomes of these thesis in both:

- decentralized communication networks;
- social networks.

List of acronyms

CN Community Network

ISP Internet Service Provider

SDN Software Defined Network

WCN Wireless Community Network

P2P peer-to-peer

SGF Spectal Graph Forge

NePA TesT Network Protocol and Application Testing Toolchain

SSSim Simple and Scalable Simulator for P2P scheduling algorithms

AWMN Athens Wireless Metropolitan Network

OLSR Optimized Link State Routing Protocol

B.A.T.M.A.N. Better Approach To Mobile Adhoc Networking

ETX Expected Transmission Count

RTT Round Trip Time

HPF High-bandwdith Peer First

SDP Service Description Protocol

ERGM Exponential family Random Graph Model

SBM Stochastic Block Model

DC-SBM Degree Corrected - Stochastic Block Model

DV Distance Vector

OSPS Open Source P2P Streaming

PSN Pocket Switched Network

List of Tables

List of Figures

Bibliography

[1] L. Baldesi, L. Maccari, and R. Lo Cigno, "Live P2P streaming in CommunityLab: Experience and insights," in *13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, June 2014, pp. 23–30.

[2] L. Baldesi, L. Maccari, and R. Lo Cigno, "Improving P2P streaming in community-lab through local strategies," in *10th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2014, pp. 33–39.

[3] L. Baldesi, L. Maccari, and R. Lo Cigno, "Improving P2P streaming in Wireless Community Networks," *Computer Networks*, vol. 93, no. Part 2, pp. 389 – 403, 2015.

[4] L. Maccari, L. Baldesi, R. Lo Cigno, J. Forconi, and A. Caiazza, "Live Video Streaming for Community Networks, Experimenting with PeerStreamer on the Ninux Community," in *ACM Workshop on Do-it-yourself Networking: An Interdisciplinary Approach*, ser. (ACM Co-located with MobySis), 2015, pp. 1–6.

[5] L. Baldesi and L. Maccari, "NePA TesT: network protocol and application testing toolchain for community networks," in *12th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, Jan 2016, pp. 1–8.

[6] L. Baldesi, L. Maccari, and R. Lo Cigno, "Optimized cooperative streaming in wireless mesh networks," in *IFIP Networking Conference (IFIP Networking) and Workshops*, May 2016, pp. 350–358.

[7] L. Maccari, N. Facchi, L. Baldesi, and R. Lo Cigno, "Optimized P2P streaming for wireless distributed networks," *Pervasive and Mobile Computing*, 2017.

[8] L. Baldesi, L. Maccari, and R. Lo Cigno, "On the Use of Eigenvector Centrality for Cooperative Streaming," *IEEE Communications Letters*, vol. 21, no. 9, pp. 1953–1956, Sept 2017.

[9] L. Baldesi, A. Markopoulou, and C. Butts, "Spectral Graph Forge: Graph Generation Targeting Modularity," in *IEEE International Conference on Computer Communications - INFOCOM*, 2018, Accepted, to appear.

[10] L. Ghiro, "Logic topology adaptation strategy comparison for P2P video distribution on wireless community netwroks," 2014, University of Trento, Italy.

[11] R. Francescato, "Feasibility study of an audio conferencing system based on PeerStreamer," 2015, University of Trento, Italy.

[12] Cisco, "Cisco visual networking index: Forecast and methodology, 2016-2021," https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html, 2016.

[13] J. Seedorf, S. Kiesel, and M. Stiemerling, "Traffic Localization for P2P-Applications: The ALTO Approach," in *IEEE International Conference on Peer-to-Peer Computing (P2P-12)*, Seattle, WA, US, Sept. 2009.

[14] H. Xie, R. Y. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4P: provider portal for applications," in *ACM SIGCOMM*, Seattle, WA, US, Aug. 2008.

[15] C. Fuchs, "Theoretical foundations of defining the participatory, co-operative, sustainable information society," *Information, Communication & Society*, vol. 13, no. 1, pp. 23–47, 2010.

[16] P. Tasca, S. Liu, and A. Hayes, "The Evolution of the Bitcoin Economy: Extracting and Analyzing the Network of Payment Relationships," http://dx.doi.org/10.2139/ssrn.2808762, 2016.

[17] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Elsevier Computer Networks*, vol. 47, no. 4, pp. 445 – 487, 2005.

[18] D. Benyamina, A. Hafid, and M. Gendreau, "Wireless Mesh Networks Design; A Survey," *IEEE Communications Surveys Tutorials*, vol. 14, no. 2, pp. 299–310, 2012.

[19] E. Alotaibi and B. Mukherjee, "A survey on routing algorithms for wireless ad-hoc and mesh networks," *Elsevier Computer Networks*, vol. 56, no. 2, pp. 940 – 965, 2012.

[20] S. Jain and D. P. Agrawal, "Wireless community networks," *IEEE Computer*, vol. 36, no. 8, pp. 90–92, 2003.

[21] B. Braem, C. Blondia, C. Barz, H. Rogge, F. Freitag, L. Navarro, J. Bonicioli, S. Papathanasiou, P. Escrich, R. Baig Viñas *et al.*, "A case for research with and on community networks," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 68–73, 2013.

[22] ZTE Corporation, "Driving the Convergence of the Physical and Digital World," http://wwwen.zte.com.cn/en/products/bearer/201402/P020140221415329571322.pdf.

[23] T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," Tech. Rep., 2003.

[24] D. Johnson, N. Ntlatlapa, and C. Aichele, "Simple pragmatic approach to mesh routing using batman," 2008.

[25] R. Birke, E. Leonardi, M. Mellia, A. Bakay, T. Szemethy, C. Kiraly, R. Lo Cigno, F. Mathieu, L. Muscariello, S. Niccolini, J. Seedorf, and G. Tropea, "Architecture of a network-aware P2P-TV application: the NAPA-WINE approach," *IEEE Communications Magazine*, vol. 49, no. 6, 2011.

[26] L. Abeni, C. Kiraly, R. Russo, M. Biazzini, and R. Lo Cigno, "Design and Implementation of a Generic Library for P2P Streaming," in *Workshop on Advanced Video Streaming Techniques for Peer-to-Peer Networks and Social Networking*, Florence, Italy, Oct. 2010.

[27] A. Russo and R. Lo Cigno, "Delay-aware push/pull protocols for live video streaming in P2P systems," in *IEEE International Conference o nCommunications (ICC)*. IEEE, 2010, pp. 1–5.

[28] R. Birke, C. Kiraly, E. Leonardi, M. Mellia, M. Meo, and S. Traverso, "Hose rate control for P2P-TV streaming systems," in *IEEE International Conference on Peer-to-Peer Computing (P2P)*, Aug 2011, pp. 202–205.

[29] D. Katsaros, N. Dimokas, and L. Tassiulas, "Social network analysis concepts in the design of wireless ad hoc network protocols," *IEEE network*, vol. 24, no. 6, 2010.

[30] S. Dolev, Y. Elovici, and R. Puzis, "Routing betweenness centrality," *Journal of the ACM (JACM)*, vol. 57, no. 4, p. 25, 2010.

[31] L. Maccari and R. Lo Cigno, "Waterwall: a cooperative, distributed firewall for wireless mesh networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, p. 225, Sep 2013.

[32] L. Maccari and R. Lo Cigno, "Betweenness estimation in olsr-based multi-hop networks for distributed filtering," *Elsevier Journal of Computer and System Sciences*, vol. 80, no. 3, pp. 670–685, 2014.

[33] A. Vázquez-Rodas and J. Luis, "A centrality-based topology control protocol for wireless mesh networks," *Elsevier Ad-Hoc Networks*, vol. 24, pp. 34–54, 2015.

[34] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay-tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 11, pp. 1576–1589, 2011.

[35] J. Niu, Y. Liu, L. Shu, and B. Dai, "A P2P query algorithm based on Betweenness Centrality Forwarding in opportunistic networks," in *IEEE International Conference on Communications (ICC)*. IEEE, 2013, pp. 3433–3438.

[36] N. Kourtellis and A. Iamnitchi, "Leveraging peer centrality in the designof socially-informed peer-to-peer systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2364–2374, 2014.

[37] K. Bryan and T. Leise, "The $25,000,000,000 eigenvector: The linear algebra behind Google," *Siam Review*, vol. 48, no. 3, pp. 569–581, 2006.

[38] Y. Andreopoulos, N. Mastronarde, and M. V. D. Schaar, "Cross-layer optimized video streaming over wireless multihop mesh networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 2104–2115, Nov 2006.

[39] J. Yuan, Z. Li, W. Yu, and B. Li, "A cross-layer optimization framework for multihop multicast in wireless mesh networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 2092–2103, Nov 2006.

[40] J. Wen, J. Cao, K. Xie, and R. Li, "User density sensitive P2P streaming in wireless mesh networks," *Journal of Parallel and Distributed Computing*, vol. 71, no. 4, pp. 573 – 583, 2011.

[41] Z. Guan, T. Melodia, and D. Yuan, "Optimizing cooperative video streaming in wireless networks," in *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*. IEEE, 2011, pp. 503–511.

[42] J. Rückert, J. Blendin, and D. Hausheer, "Software-Defined Multicast for Over-the-Top and Overlay-based Live Streaming in ISP Networks," *Springer Journal of Network and Systems Management*, vol. 23, no. 2, pp. 280–308, Jul. 2014.

[43] J. Ruckert, J. Blendin, and D. Hausheer, "RASP: Using OpenFlow to Push Overlay Streams into the Underlay," in *IEEE International Conference on Peer-to-Peer Computing (P2P-13)*, Sept. 2013, pp. 1–2.

[44] A. Payberah, H. Kavalionak, V. Kumaresan, A. Montresor, and S. Haridi, "CLive: Cloud-assisted P2P live streaming," in *IEEE International Conference Peer-to-Peer Computing (P2P-12)*, Sept. 2012, pp. 79–90.

[45] J.-L. Kuo, C.-H. Shih, C.-Y. Ho, and Y.-C. Chen, "A cross-layer approach for real-time multimedia streaming on wireless peer-to-peer ad hoc network," *Elsevier Ad Hoc Networks*, vol. 11, no. 1, pp. 339–354, 2013.

[46] E. Setton, T. Yoo, X. Zhu, A. Goldsmith, and B. Girod, "Cross-layer design of ad hoc networks for real-time video streaming," *IEEE Wireless Communications*, vol. 12, no. 4, pp. 59–65, 2005.

[47] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, 2004, pp. 15–26.

[48] T. E. Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, 2002, pp. 170–179.

[49] Y. Chen, X. Wang, C. Shi, E. K. Lua, X. Fu, B. Deng, and X. Li, "Phoenix: A weight-based network coordinate system using matrix factorization," *IEEE*

Transactions on Network and Service Management, vol. 8, no. 4, pp. 334–347, 2011.

[50] R. Fortuna, E. Leonardi, M. Mellia, M. Meo, and S. Traverso, "QoE in pull based P2P-TV systems: Overlay topology design tradeoffs," in *IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 2010, pp. 1–10.

[51] S. Traverso, L. Abeni, R. Birke, C. Kiraly, E. Leonardi, R. Lo Cigno, and M. Mellia, "Neighborhood Filtering Strategies for Overlay Construction in P2P-TV Systems: Design and Experimental Comparison," *IEEE/ACM Transactions on Networking*, 2014.

[52] M. Conti, E. Gregori, and G. Turi, "A Cross-layer Optimization of Gnutella for Mobile Ad Hoc Networks," in *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2005, pp. 343–354.

[53] J. A. Oliveira, F. Prado, F. M. de Lima, M. Rubinstein, and A. Sztajnberg, "Improving peer neighborhood on P2P video distribution networks using Push/Pull protocol," *Computer Communications*, vol. 61, pp. 17–33, 2015.

[54] J. Zhang, W. Xing, Y. Wang, and D. Lu, "Modeling and performance analysis of pull-based live streaming schemes in peer-to-peer network," *Elsevier Computer Communications*, vol. 40, pp. 22–32, 2014.

[55] B. Bellalta, E. Belyaev, M. Jonsson, and A. Vinel, "Performance evaluation of IEEE 802.11 p-enabled vehicular video surveillance system," *IEEE Communications Letters*, vol. 18, no. 4, pp. 708–711, 2014.

[56] K. Mokhtarian and H. A. Jacobsen, "Minimum-delay multicast algorithms for mesh overlays," *IEEE/ACM Transactions on Networking*, vol. 23, no. 3, pp. 973–986, June 2015.

[57] D. S. Lun, N. Ratnakar, R. Koetter, M. Medard, E. Ahmed, and H. Lee, "Achieving minimum-cost multicast: a decentralized approach based on network coding," in *IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3, March 2005, pp. 1607–1617 vol. 3.

[58] Z. Li and B. Li, "Efficient and distributed computation of maximum multicast rates," in *IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3, March 2005, pp. 1618–1628 vol. 3.

[59] C. Wu and B. Li, "Optimal rate allocation in overlay content distribution," in *IFIP Networking*, 2007, pp. 678–690.

[60] F. Harary, *Graph theory.* Addison-Wesley, Reading, MA, 1969.

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

[62] M. Gjoka, M. Kurant, and A. Markopoulou, "2.5k-graphs: From sampling to generation," in *IEEE INFOCOM*, April 2013, pp. 1968–1976.

[63] C. Orsini, M. M. Dankulov, P. Colomer-de Simón, A. Jamakovic, P. Mahadevan, A. Vahdat, K. E. Bassler, Z. Toroczkai, M. Boguñá, G. Caldarelli, S. Fortunato, and D. Krioukov, "Quantifying randomness in real networks," *Nature Communications*, vol. 6, p. 8627, oct 2015.

[64] J. S. Coleman, *Introduction to mathematical sociology.* London Free Press Glencoe, 1964.

[65] L. Freeman, "The development of social network analysis," *A Study in the Sociology of Science*, vol. 1, 2004.

[66] R. A. Horn and C. R. Johnson, *Matrix analysis.* Cambridge university press, 2012.

[67] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[68] L. Mirsky, "Symmetric gauge functions and unitarily invariant norms," *The quarterly journal of mathematics*, vol. 11, no. 1, pp. 50–59, 1960.

[69] D. Carra, R. L. Cigno, and E. W. Biersack, "Stochastic graph processes for performance evaluation of content delivery applications in overlay networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 2, pp. 247–261, Feb 2008.

[70] S. Voulgaris, D. Gavidia, and M. Van Steen, "Cyclon: Inexpensive membership management for unstructured P2P overlays," *Springer Journal of Network and Systems Management*, vol. 13, no. 2, pp. 197–217, 2005.

[71] N. Tölgyesi and M. Jelasity, "Adaptive peer sampling with newscast," in *European Conference on Parallel Processing*, 2009, pp. 523–534.

[72] Y. Sakata, K. Takayama, R. Endo, and H. Shigeno, "A chunk scheduling based on chunk diffusion ratio on P2P live streaming," in *Network-Based Information Systems (NBiS), 2012 15th International Conference on*, 2012, pp. 74–81.

[73] C. Kiraly, R. Lo Cigno, and L. Abeni, "Deadline-based differentiation in P2P streaming," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, 2010, pp. 1–6.

[74] K.-L. Hua, G.-M. Chiu, H.-K. Pao, and Y.-C. Cheng, "An efficient scheduling algorithm for scalable video streaming over P2P networks," *Elsevier Computer Networks*, vol. 57, no. 14, pp. 2856–2868, 2013.

[75] L. Abeni, C. Kiraly, and R. Lo Cigno, "Robust scheduling of video streams in network-aware P2P applications," in *Communications (ICC), 2010 IEEE International Conference on*, 2010, pp. 1–5.

[76] Y. Liu, "On the minimum delay peer-to-peer video streaming: how realtime can it be?" in *15th ACM international conference on Multimedia*. ACM, 2007, pp. 127–136.

[77] L. Abeni, C. Kiraly, and R. Lo Cigno, "On the optimal scheduling of streaming applications in unstructured meshes," in *International Conference on Research in Networking*. Springer, 2009, pp. 117–130.

[78] J. Zhang, C. Yang, and X. Zhang, "A high-bandwidth live streaming model in mesh-based peer-to-peer networks," *IEEE Communications Letters*, vol. 20, no. 12, pp. 2390–2393, 2016.

[79] A. Neumann, I. Vilata, X. Leon, P. E. Garcia, L. Navarro, and E. Lopez, "Community-lab: Architecture of a community networking testbed for the future internet," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on*, 2012, pp. 620–627.

[80] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 255–270, 2002.

[81] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, 2003.

[82] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, p. 19.

[83] V. Autefage and D. Magoni, "Network emulator: a network virtualization testbed for overlay experimentations," in *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2012 IEEE 17th International Workshop on*, 2012, pp. 266–270.

[84] K. Nakajima, S. Kurebayashi, Y. Fukutsuka, T. Hieda, I. Taniguchi, H. Tomiyama, and H. Takada, "Naxim: A fast and retargetable network-on-chip simulator with qemu and systemc," *International Journal of Networking and Computing*, vol. 3, no. 2, pp. 217–227, 2013.

[85] F. Bellard, "Qemu, a fast and portable dynamic translator." in *USENIX Annual Technical Conference, FREENIX Track*, 2005, pp. 41–46.

[86] B. Milic and M. Malek, "Npart-node placement algorithm for realistic topologies in wireless multihop network simulation," in *Proceedings of the 2nd international conference on simulation tools and techniques*, 2009, p. 9.

[87] L. Cerda-Alabern, "On the topology characterization of guifi. net," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on*, 2012, pp. 389–396.

[88] L. Abeni, C. Kiraly, and R. Lo Cigno, "SSSim: a simple and scalable simulator for P2P streaming systems," in *Computer Aided Modeling and Design of Communication Links and Networks, 2009. CAMAD'09. IEEE 14th International Workshop on*, 2009, pp. 1–6.

[89] M. E. M. Campista, P. M. Esposito, I. M. Moraes, L. H. M. Costa, O. C. M. Duarte, D. G. Passos, C. V. N. De Albuquerque, D. C. M. Saade, and M. G. Rubinstein, "Routing metrics and protocols for wireless mesh networks," *IEEE network*, vol. 22, no. 1, 2008.

[90] H. D. Sherali and J. C. Smith, "An improved linearization strategy for zero-one quadratic programming problems," *Springer Optimization Letters*, vol. 1, no. 1, pp. 33–47, 2007.

[91] D. Pisinger, "The quadratic knapsack problem—a survey," *Elsevier Discrete applied mathematics*, vol. 155, no. 5, pp. 623–648, 2007.

[92] U. Brandes and C. Pich, "Centrality estimation in large networks," *World Scientific International Journal of Bifurcation and Chaos*, vol. 17, no. 07, pp. 2303–2318, 2007.

[93] R. Puzis, P. Zilberman, Y. Elovici, S. Dolev, and U. Brandes, "Heuristics for speeding up betweenness centrality computation," in *International Conference on Privacy, Security, Risk and Trust (PASSAT) and International Conference on Social Computing (SocialCom)*. IEEE, 2012, pp. 302–311.

[94] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *IEEE International Symposium on Computer Aided Control Systems Design*. IEEE, 2004, pp. 284–289.

[95] K. Marx, *Critique of the Gotha program*. Wildside Press LLC, 2008.

[96] H. Ishii and R. Tempo, "Distributed randomized algorithms for the pagerank computation," *IEEE Transactions On Automatic Control*, vol. 55, no. 9, pp. 1987–2002, Sept 2010.

[97] M. Selimi, F. Freitag, R. P. Centelles, and A. Moll, "Distributed storage and service discovery for heterogeneous community network clouds," in *IEEE/ACM 7th International Conference on Utility and Cloud Computing*, Dec 2014, pp. 204–212.

[98] L. Maccari, L. Baldesi, N. Facchi, R. Lo Cigno, F. Freitag, M. Karaliopoulos, M. Panagiota, and A. Pilichos, "D3.4: Release of new open source software for all applications," https://netcommons.eu/?q=content/deliverables-page, net-Commons project, June 2017.

[99] Ö. N. Yaveroğlu, N. Malod-Dognin, D. Davis, Z. Levnajic, V. Janjic, R. Karapandza, A. Stojmirovic, and N. Pržulj, "Revealing the hidden language of complex networks," *Nature Scientific reports*, vol. 4, 2014.

[100] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *American Association for the Advancement of Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[101] M. Gjoka, B. Tillman, and A. Markopoulou, "Construction of simple graphs with a target joint degree matrix and beyond," in *INFOCOM*, 2015, pp. 1553–1561.

[102] P. Mahadevan, D. Krioukov, K. Fall, and A. Vahdat, "Systematic topology analysis and generation using degree correlations," *ACM SIGCOMM Computer Communication Review*, vol. 36, pp. 135–146, 2006.

[103] R. Aldecoa, C. Orsini, and D. Krioukov, "Hyperbolic graph generator," *Computer Physics Communications*, vol. 196, pp. 492–496, 2015.

[104] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*. Cambridge university press, 1994, vol. 8.

[105] S. Trajanovski, F. A. Kuipers, J. Martín-Hernández, and P. Van Mieghem, "Generating graphs that approach a prescribed modularity," *Elsevier Computer Communications*, vol. 36, no. 4, pp. 363–372, 2013.

[106] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *American Physical Society Phys. Rev. E*, vol. 83, no. 1, p. 016107, Jan 2011.

[107] M. E. Newman, "Modularity and community structure in networks," *National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[108] W. Richards and A. Seary, "Multinet for windows 4.76 [computer program]," 2006.

[109] P. D. Hoff, A. E. Raftery, and M. S. Handcock, "Latent space approaches to social network analysis," *Taylor & Francis Journal of the American Statistical Association*, vol. 97, no. 460, pp. 1090–1098, 2002.

[110] S. P. Borgatti and M. G. Everett, "Models of core/periphery structures," *Social Networks*, vol. 21, no. 4, pp. 375 – 395, 2000.

[111] T. A. Snijders and K. Nowicki, "Estimation and Prediction for Stochastic Blockmodels for Graphs with Latent Block Structure," *Journal of Classification*, vol. 14, no. 1, pp. 75–100, Jan 1997.

[112] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[113] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *APS Physical review E*, vol. 78, no. 4, p. 046110, 2008.

[114] K. M. Harris, C. T. Halpern, E. Whitsel, J. Hussey, J. Tabor, P. Entzel, and J. R. Udry, "The national longitudinal study of adolescent to adult health: Research design," *See http://www. cpc. unc. edu/projects/addhealth/design (accessed 9 April 2015)*, 2009.

[115] J. J. Mcauley and J. a. Leskovec, "Learning to discover social circles in ego networks," in *Advances in neural information processing systems*, 2012, pp. 539–547.

[116] S. Ji, W. Li, P. Mittal, X. Hu, and R. A. Beyah, "Secgraph: A uniform and open-source evaluation system for graph data anonymization and de-anonymization." in *USENIX Security Symposium*, 2015, pp. 303–318.

[117] M. Srivatsa and M. Hicks, "Deanonymizing mobility traces: Using social network as a side-channel," in *ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 628–637.

[118] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual Network Embedding: A Survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, 2013.