UNIVERSITÉ DU
LUXEMBOURG

PhD-FSTC-2012-01
The Faculty of Sciences, Technology and Communication

DISSERTATION

Presented on 16/01/2012 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Guillaume-Jean HERBIET
Born on 11 February 1983 in Thionville (France)

SOCIAL NETWORK ANALYSIS OVER DYNAMIC GRAPHS AND APPLICATION TO URBAN MOBILE AD HOC NETWORKS

Dissertation defense committee

Dr Pascal Bouvry, dissertation supervisor
Professor, Université du Luxembourg, Luxembourg

Dr Frédéric Guinand
Professor, Université du Havre, France

Dr Marco Tomassini
Assistant Professor, Université de Lausanne, Suisse

Dr Steffen Rothkugel, Vice Chairman
Assistant Professor, Université du Luxembourg, Luxembourg

Dr Thomas Engel, Chairman
Professor, Université du Luxembourg, Luxembourg

Abstract

Wireless mobile ad hoc networks (MANET) are composed of mobile communicating devices that self-organize to ubiquitously exchange information over the air. Envisioned applications for such networks cover disaster relief situations, battlefield deployment and more generally any use-case that makes the deployment of wired or infrastructure-based networks too costly or simply unsuitable. Lately, new applications for MANETs are developing in the urban environment, namely mobile social networks and vehicular ad hoc networks. Despite this large spectrum of possible applications and the large number of wireless-capable devices available today, mobile ad hoc networks remain confidential. One important reason is the lack of reliability of wireless communications, especially on long path.

In this thesis, we propose to cope with this problem by creating virtual structures that will group together limited set of users that are densely and reliably connected in order to favor the dynamic and robust exchange of information. Our proposal uses the concept of community, that first appeared in social network analysis. After reviewing the main concepts taken from this branch of graph theory and justifying the application by underlining the specificities of MANETs in the urban context, we formally present our contribution, based on epidemic propagation of community labels. Then we exhibit its applications on concrete communication systems and how it can benefit more generally to improve the management of wireless ad hoc networks topology.

To the loving memory of my grandmother,
from whom I learned a lot.

To my son Valentin,
to whom I'll try to teach as much.

Acknowledgements

First, I would like to thank my advisor for his trust and confidence in my work. I had the opportunity to freely design not only the solutions presented here, but also to define the particular problem I wanted to assess and more generally to organize my work as I wished. This has been a formative and enriching process, far beyond the sole scientific aspect.

Another thank if for my team at the University of Luxembourg, especially those with which I shared so many fruitful and enjoyable discussions, either work-related or not. I also enjoyed our virtual or concrete games, despite that I was a sore player.

I also wish to express my gratitude to the LITIS lab from Université du Havre, and particularly Prof. Frédéric Guinand, who insufflated new ideas and motivation precisely when I was needing them. A special thought goes to the GraphStream development team for their help when developing new components and using their graph library.

Finally, a special thank goes to my family, in the broad sense. I highly regard my parents for their unconditional support in whatever I do and, last but not least, my beloved Claire. She provided all the love, care and energy that was required for me to stay on the tracks of the emotional roller-coaster that a thesis is.

Thesis committee

Committee Chairman
Prof. Dr. Thomas Engel
Université du Luxembourg, Luxembourg

Deputy Committee Chairman
A.-Prof. Dr. Steffen Rothkugel
Université du Luxembourg, Luxembourg

Committee Member
Prof. Dr. Frédéric Guinand
Université du Havre, France

Committee Member
A.-Prof. Dr. Marco Tomassini
Université de Lausanne, Suisse

Dissertation Supervisor
Prof. Dr. Pascal Bouvry
Université du Luxembourg, Luxembourg

Contents

List of Figures

List of Tables

List of Algorithms

List of Definitions

Contents

Chapter 1

Introduction

"Yet only through communication can human life hold meaning."
Paulo Freire, Pedagogy of the Oppressed.

The last century has seen more technical advances in means of communications than the two thousands years that separate it from the first traces of written text. It has been marked by inventions such as the Internet (Kahn, Cerf *et al.*, 1983), electronic mail (Tomlinson, 1971) and the World Wide Web (Berners-Lee, 1990) but also the introduction of mobile telephone services (in 1946 and later in 1977 by AT&T and the Bell Labs) and satellite communications (Masterman-Smith, 1958). Those breakthroughs not only radically changed the way we communicate with our fellow human beings, but they also changed the scale, or range, of those communications. By allowing users thousand kilometers away to instantly communicate, should we say to *get connected*, they have turned the world into what we reckon today as a global village.

More recently, the introduction of high speed wireless communications systems, such as IEEE 802.11 (Wi-fi) or 3G cellular protocols (like UMTS), along with the development of handheld communication-enabled devices (laptops, smartphone) came to complement this evolution. Previously, either stationary devices, like a desktop computer or a wired access-point, were most often required to access the communication network. Also, the users ability to communicate on the move was so restrained, due to the communication protocols design, the limited bandwidth or the terminal autonomy, that it almost sounded like stationarity. Those constraints can today be considered lifted, as we can hold a durable tremendous amount of computational and communication power in the palm of our hands.

The graph presented in Figure 1.1 below shows the CPU performances of the chipsets embedded in previous generation handheld communicating devices (the ARM chipset family at the leftmost), of last-generation terminals (namely the Cortex A8 and A9 respectively powering Apple iPhone 4 and iPad) compared with the Atom processors powering entry-level laptop computers. It shows that even the tiniest handheld device nowadays embeds a computational power comparable to "computers" as we usually define them. Besides, those communication terminals are generally capable of using both WiFi and cellular wireless communication technologies, combining a high-speed access on the local area and extended coverage with an acceptable bandwidth, using third generation cellular communication protocols (see Figure 1.2).

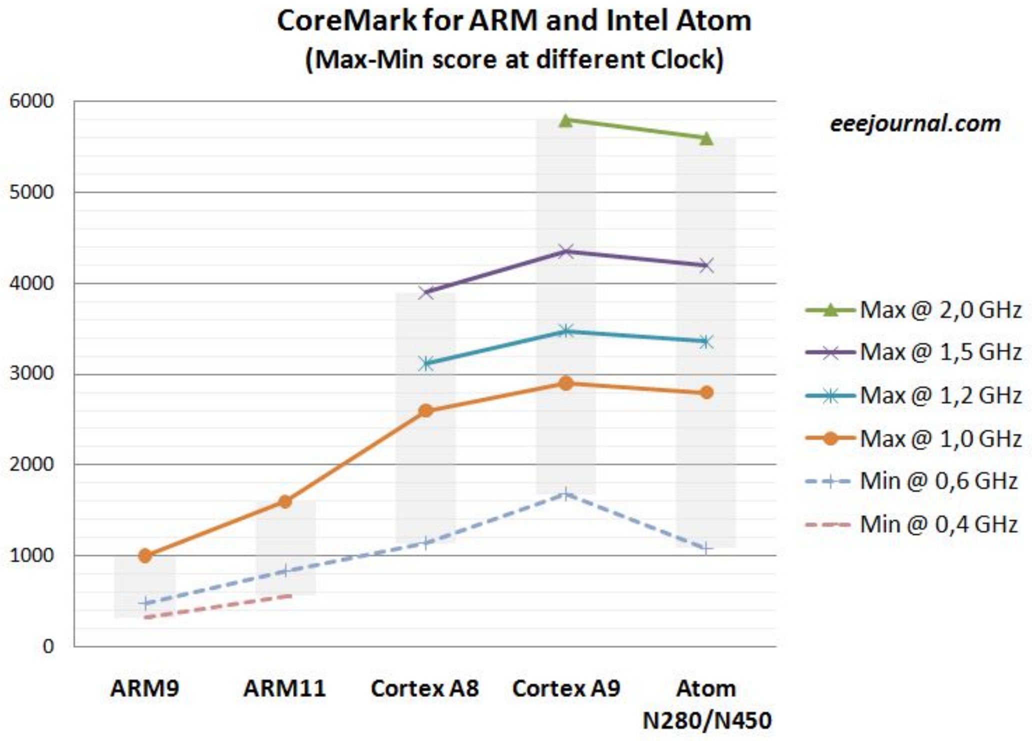


Figure 1.1: A comparison of mobile devices CPU performances using the CoreMark benchmark software (Source: EEEjournal.com)

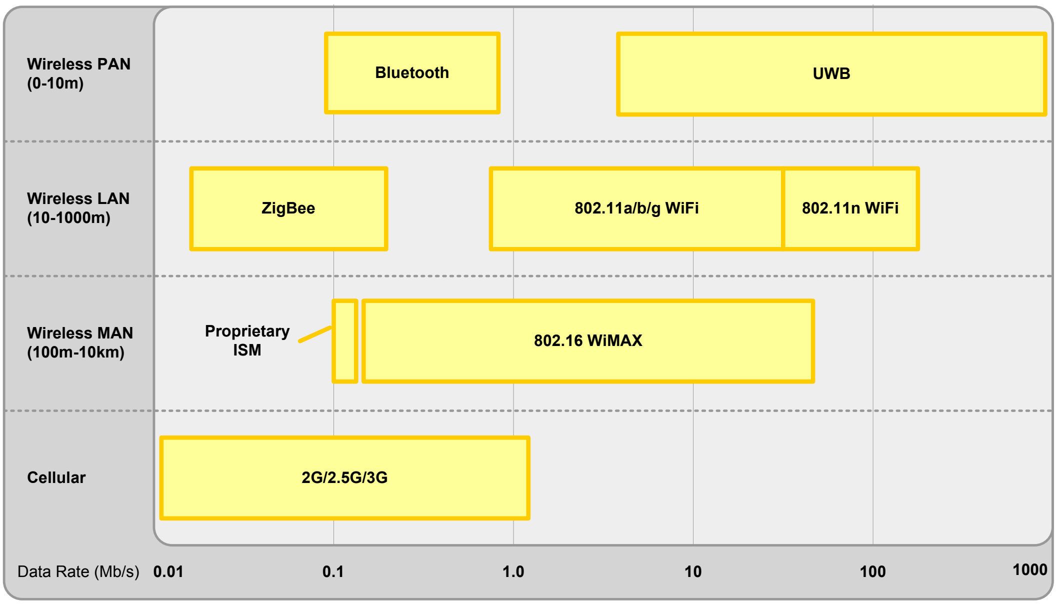


Figure 1.2: A comparison of the data rate of different wireless communication standards

1.1 From terminals to network entities

In this section, we will express the rationale for following a design different from what traditionally governs the operation of communication networks.

1.1.1 The client-server model

The development of the Internet as a global communication and service network was quickly marked by the emergence of the client-server model. Although this was not necessarily the original design[Bar64], the intelligence, computational and storage capacities were inside the network, leaving to the terminals at the edge of the network mainly with data representation and user interaction tasks. This was the paradigm that governed the mainframe systems of the early eighties and is, somewhat strangely, the approach followed nowadays with cloud computing: data is stored in a set of remote servers and the operating system can be limited to a browser that access services using the World Wide Web or to a set of applications that heavily rely on the cloud.

This pattern was also replicated in the design of cellular and wireless networks. As presented in Figure 1.3(a), those network rely only on wireless links for the last hop, i.e. the last relay of the messages. Besides, the wireless terminals are not able to communicate directly, all messages has to be sent through the access point, called base station for cellular networks. Most of the time, the wireless devices act as slave of access points that control the frequency, duration and data rate of the transfers occurring over the air.

1.1.2 A network of collaborating entities

However, when considering the computational and communication power, as well as the storage capacities of smartphones and other handheld communicating devices, it appears strange that

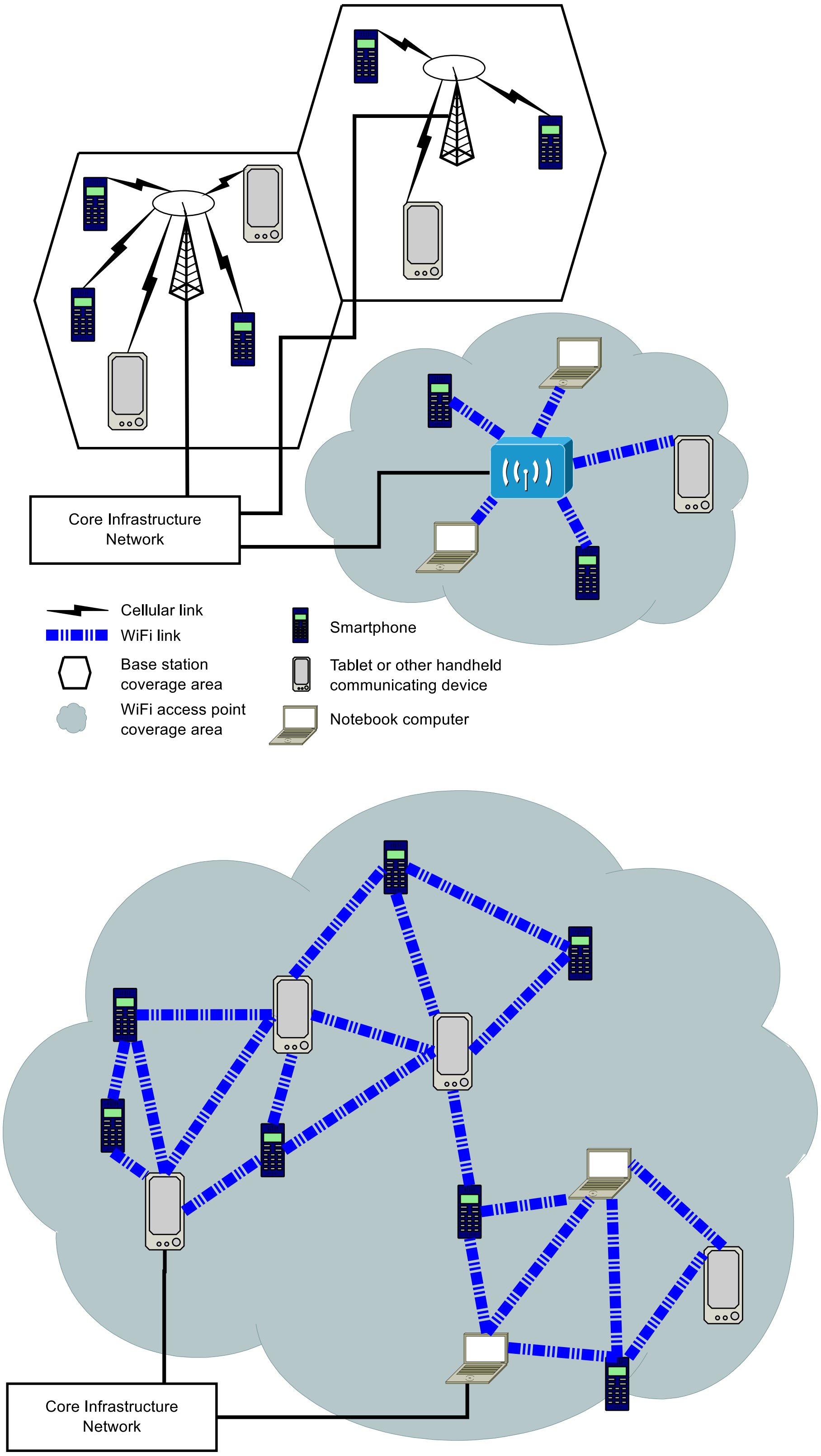

Core Infrastructure Network
Cellular link
WiFi link
Base station coverage area
WiFi access point coverage area
Smartphone
Tablet or other handheld communicating device
Notebook computer
Core Infrastructure Network

Figure 1.3: Centralized versus ad hoc wireless network

they remain "dumb" terminals, from the communication protocol and applicative point of view.

As we will see later in this manuscript (see subsubsection 4.2.2.2), relying on a centralized organization for wireless communications and for data access and distribution is neither a more efficient nor a more robust approach. This solution is acceptable only when it is dictated by technological constraints.

On the contrary, the computational and communication capacities of nowadays communicating devices allows them to play an active role not only in the coordination of communication, but also on the storage, dissemination, generation, merging of information.

Those collaborative entities, each with their individual intelligence, will organize themselves spontaneously and dynamically to form *ad hoc networks*. Those networks, referred to as *wireless ad hoc networks* or *mobile ad hoc networks* (MANET) when they involve mobile devices communicating over the air, are illustrated in Figure 1.3(b).

In this configuration, devices can dynamically transfer information between them, without any coordination point. The only restriction is that they are in the transmission range of each other. Besides, they also act as transmission relays (i.e. they can repeat, or retransmit, information that were not intended for them).

Such a paradigm can be envision as an alternative, or a complement, to the traditional centralized communication infrastructures (cellular and Wi-Fi based for instance).

1.2 The social impact of novel communication means

The question of the impact of a mean of communication, or the *communication network* that is formed by registering the interactions between individuals that occur using this particular communication form, is not new. Since the late sixties, sociologists were intrigued by the characteristics of human relations and their ability to favor, or refrain, the spread of trends, information, diseases.

1.2.1 It's a small world after all

As a forerunner, Stanley Milgram launched in 1967 a famous experiment that showed how a thousand randomly chosen americans can get in touch with a single ordinary person through an average of six intermediate connections. This observed phenomenon remained famous as the *six degrees of separation* property.

Connecting two individuals, randomly chosen in a large population, with only a few intermediate links is not necessarily trivial. It requires both that each individual has a lot of connections (at least enough so that it is possible to travel from any individual to another with a good probability) but also that they all have a sufficient set of *common* connections (which makes easy to find common relations between two unrelated individuals, diminishing the required number of intermediate acquaintances).

This made the accepted view of the social network of human acquaintances shift from a pure branching network to a graph exhibiting a high clustering factor (see Figure 1.4), often referred to as *small world* network. Later on, sociologists began investigating the importance of this *network structure* on *social organization*, hence opening a new field of research: *social network analysis* which we will investigate deeper in chapter 3.

From the analysis of human relationships, it appears that individuals are already connected in a global network that allows the diffusion of information and trends.

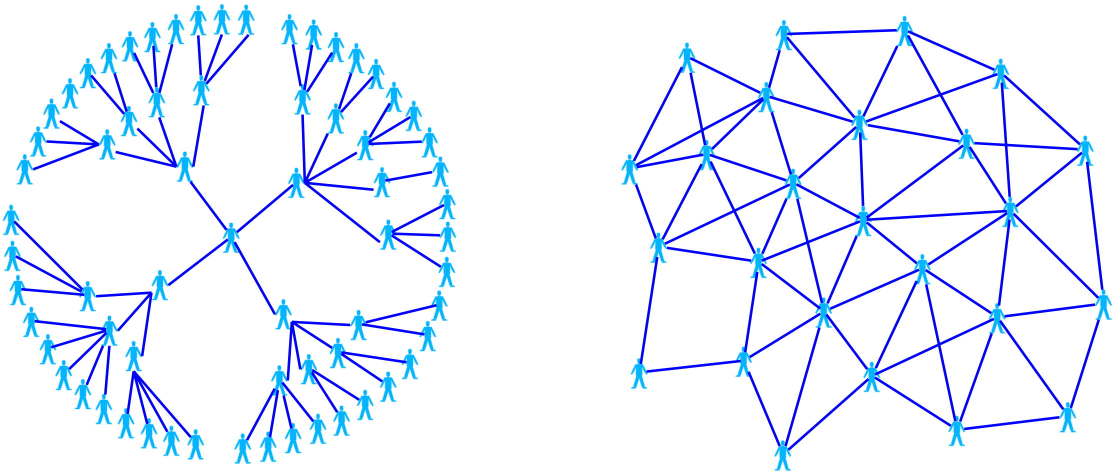


Figure 1.4: Branching network structure versus social network structure

1.2.2 Anytime, anywhere

The early experiments on the social graphs of the sixties revealed already dense networks able to favor the quick dissemination of new trends, or more sadly diseases, across the world. Yet, they were only relying on the media existing at that time: direct interactions, books, newspapers, radio, television, early forms of electronic communication.

Thanks, or because, of the last generation communicating devices presented in section 1.1 and after the extension of the *scale* of human communications, i.e. their relation to space, and the progressive concentration of links, we are now experiencing a dramatic change in their *availability*, i.e. their relation to time.

We are now in the situation where electronic communications can instantly happen not only across the world, but *anytime and anywhere*. They are many more opportunities for each of us to interconnect and we tend to use at least a good fraction of those opportunities to actually communicate.

1.2.3 From private to personal communication

This ease of access to performant communication vectors also changed the relation that, as human beings, we have with communication. When those opportunities were scarce and resource limited, they were dedicated to vital interactions. It is hence no surprise that radio communications, and later the grounds of the Internet (the ARPANET project, lead by US Department of Defense in 1969), were designed and first implemented by the military, before being used for government information transit, news report and finally all sorts of commercial and private communications.

The limited size of those smart communicating devices, as well as their design (from both hardware and software aspects) definitely make them personal. As a consequence, they open a new dimension to human communication, that goes beyond private communications. Whereas the latter are targeted to a very limited audience (family, friends and other relatives generally known from outside the communication system), this new kind of interaction, denoted as *personal communication* is all about broadcasting to the largest possible audience data about the user itself or what he is experiencing. Arch-examples of this new trend lie in social networks such as Facebook or micro-blogging services such as Twitter and its now famous request: "What's happening?".

The study of those new personal, yet public, interactions and their original ubiquitous nature under the light of social network analysis is an interesting open problem.

1.3 Contributions and organization

In this manuscript, the overall objective is to show that it is relevant to use social network theory paradigms to improve the performance and design novel services for wireless ad hoc networks. This is achieved by showing that such networks naturally exhibit properties similar to traditional interaction networks used in social network analysis, by proposing original solutions to perform a distributed and real-time online social structure mining over ad hoc networks and by describing examples of application that make an extensive use of the discovered structures.

1.3.1 Contributions

Therefore, the contributions of this thesis can be summarized as follows:

- An introduction to graph theory, especially concerning dynamic graphs, and to graph models and social network analysis;
- An extensive state-of-the-art of community detection techniques, a particular method of social network analysis, along with a suitability analysis of the referred solutions with wireless communication networks;
- A description of the modeling efforts to simulate wireless communication networks in a urban environment along with:
  - a rationale for using social network analysis techniques on such communication networks;
  - a proposal of a new social-aware mobility model, based on user profiles and designed for simulation of urban environments;
- The introduction of a family of distributed algorithms to perform community detection over communication graphs, amongst which are, to our knowledge, the first algorithms designed especially for dynamic graphs;
- The design or improvement, using our contributions to distributed community detection, of several algorithms for topology management of dynamic communication graphs.

1.3.2 Organization

As a consequence, the remaining of this document is organized as follows:

Part One introduces elements of graph theory and social network analysis. After a review of the different graph topology models and their characteristics, we will investigate the relevance of using this mathematical abstraction in order to study social interactions. Finally, we will concentrate on the notion of *community*, a particular structure exhibited by social network analysis, by presenting its specificities, relevance and determination techniques.

Part Two presents in further details the concept of *wireless ad hoc networks*. We will introduce the principles of operation and examples of use cases for those networks. We will also emphasize on the *distributed* aspect of those networks and more generally insist on the specificities of distributed systems. Finally, we will focus on the *dynamic* nature of ad hoc networks in order to introduce the necessary notions related to dynamic systems.

Part Three details the necessary *modeling* effort of wireless ad hoc networks in order to study them using graph theory and social network analysis techniques and why using those tools is relevant. We will first focus on a sound representation of the mobility of users of communicating handheld devices, with an emphasize on the social aspects of this phenomenon. Then, we will study how radio waves propagation and connectivity model impacts the structure of the communication graph. Finally, we will introduce notions of stability, robustness and other metrics related to the network dynamics.

Part Four presents a set of new social network analysis techniques, suitable for wireless ad hoc networks, and more generally dynamic communication systems. These distributed solutions are based around a novel heuristic that allows a sharper and more robust determination of social structures over dynamic graphs. We also exhibit results of thorough evaluations of these techniques, using both traditional social interaction graphs and original scenarios based on realistic uses cases. We will also envision a distributed social network application based on user proximity and an efficient car-to-car information dissemination and reporting to enhance traffic management.

Part Five reveals applications and benefits of the mining of social structure in dynamic networks, such as wireless communication networks. Namely, we will show how social structures can be used as a scalability helper for other topology management mechanisms generally used in wireless ad hoc networks, like spanning trees and virtual backbones. We will also see how community information can help in the energy-efficient broadcast process.

To conclude this manuscript we will review our contributions, and see how the mining of social structures can not only benefit to the wireless ad hoc communication networks, but can also be extended to benefit to other communication networks such as the Internet and associated services.

I

From graphs to dynamic social networks

Contents

Chapter
2

Elements of graph theory

"The origins of graph theory are humble, even frivolous..."
Biggs, Lloyd and Wilson, Graph Theory 1736-1936.

Throughout this thesis we focus on fundamental properties of wireless ad hoc networks, including the connectivity, the degree distribution and the hop count. To study those properties we will rely on a mathematical abstraction that represents communicating entities as vertices and communication links as edges of a graph. The study of graphs is known as graph theory.

In this early chapter, we will hence introduce the essential notions of the mathematical theory of graphs. After an historical introduction of the notion of graph and its usage to model problems, we will focus on giving proper definitions that will be used throughout this manuscript. Finally, we will also present how this theory can be extended so that graphs become suitable to abstract from communication networks where entities and communication links may evolve over time.

2.1 Graphs

The use of graphs is far from being limited to the modeling of wireless ad hoc communications. In fact many problems can be conveniently abstracted as a diagram consisting of a set of points, representing elements, actors or constraints of the envisioned problem and lines joining certain pairs of these points, figuring interactions, relations or transit possibilities between vertices.

2.1.1 Historical approach

According to Biggs, Lloyd and Wilson in [BLW77] the first scientific publication using graph theory is attributed to Leonhard Euler in 1736. It dealt about a problem that remained famous as the "Seven Bridges of Königsberg". The city included two large islands which were connected to each other and the mainland by seven bridges. The problem was to find a walk through the city that would cross each bridge once and only once. The city map is illustrated in Figure 2.1 and was abstracted by Euler as a graph of four vertices (the two parts of the city and the two islands) and seven edges (the bridges). Euler proved that the problem has no solution as a necessary condition for the walk to be of the desired form is that the graph is connected and has exactly zero or two nodes of odd degree. This lead to the notion of *Euler path* in graph theory.

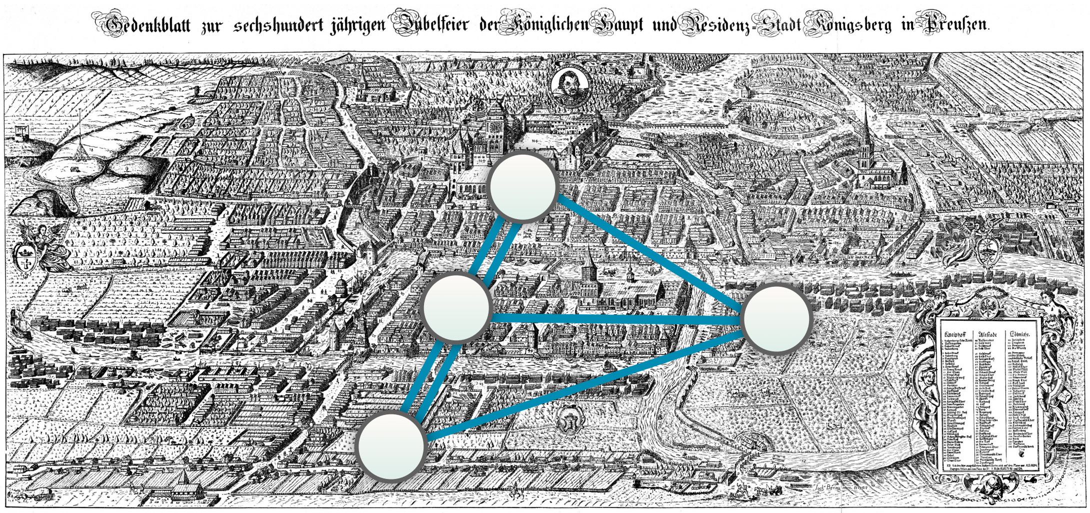

Gedenkblatt zur sechshundert jährigen Jubelfeier der Königlichen Haupt und Residenz-Stadt Königsberg in Preußen.

Figure 2.1: An illustration of the Konigsberg bridges abstraction by a graph

Later Cayley studies on differential calculus, that also had implications in chemistry, as well as fundamental results by Polya and others by De Bruijn lead to further formalization of graph theory, in the first half of the 20th century.

Another important problem in the development of graph theory is the "four color problem" asking for a solution to paint adjacent regions of a plane map with a different colors, while using only four different colors. First posed by Guthrie in 1852, it was solved by computer only in the late sixties.

Later, the introduction of probabilistic methods in graph theory, especially in the study of Erdös and Rényi of the asymptotic probability of graph connectivity, gave rise to yet another branch, known as random graph theory, which has been a fruitful source of graph-theoretic results.

Today, graphs are useful tools for the design of electronic devices, the study of interaction of species, the expression of genes in natural sciences, the modeling of communication networks, amongst other applications.

2.1.2 Definitions

We will now formally define graphs and their core elements, as well as a set of properties that we will refer to later through this manuscript.

Definition 2.1 (Graph)

A *graph* $\mathcal{G}$ is an ordered pair $(V(\mathcal{G}), E(\mathcal{G}))$, more concisely noted (V, E), consisting of a set $V(\mathcal{G})$ of vertices and a set $E(\mathcal{G}) \subset V(\mathcal{G}) \times V(\mathcal{G})$, disjoint from $V(\mathcal{G})$, of edges.

Graphs are generally represented with dots or circles figuring the vertices and straight or curved lines representing the edges, as in Figure 2.1.

Each edge $e \in E(\mathcal{G})$ is associated to a pair of vertices $\{u, v\}$ ordered or unordered, by an *incidence function* that represents the distribution of edges over the graph. If the pair of vertices u, v is ordered, the graph is *directed*, otherwise it is undirected. In most of this document, we will consider undirected graphs.

The same, we will consider $\mathcal{G}$ as being a *simple graph*, i.e. with a single non-loop edge that goes between any pair of vertices. This assumption makes sense for communication graphs where internal communications (inside the same communicating entity) are not considered.

The number of vertices of a graph $\mathcal{G}$, noted $|V(\mathcal{G})| = |V| = N$ is referred to as the graph $order$, while the number of edges, noted $|E(\mathcal{G})| = |E| = M$ is the graph $size$.

Definition 2.2 (Subgraph)

A subgraph $\mathcal{S}(\mathcal{G}) = (V_{\mathcal{S}}(\mathcal{G}), E_{\mathcal{S}}(\mathcal{G}))$ of a graph $\mathcal{G}$ is a graph so that $V_{\mathcal{S}}(\mathcal{G}) \subseteq V(\mathcal{G})$ and $E_{\mathcal{S}}(\mathcal{G}) \subseteq E(\mathcal{G})$.

2.1.2.1 Adjacency and paths

Edges are representing the *connectivity* of the graph, i.e. how the vertices are connected. It is hence important to characterize the impact of connectivity on the relation between vertices of the graph.

Definition 2.3 (Adjacent vertices)

A vertex $v \in V(\mathcal{G})$ is said *adjacent* to another vertex $u \in V(\mathcal{G})$ if and only if there exist an edge $e = (u, v) \in E(\mathcal{G})$ between u and v.

Note that, in an undirected graph if v is adjacent to u, then u is adjacent to v.

Definition 2.4 (Neighborhood)

The adjacent vertices of a vertex v of $\mathcal{G}$ are called *neighbors* of v. The set of neighbors $N(v)$ of a vertex v of $\mathcal{G}$ is called the *neighborhood* of v.

It is here useful to define the *two-hop* neighborhood of a node, as it is often used in distributed communication protocols, like in wireless ad hoc networks:

Definition 2.5 (Two-hop neighborhood)

The *two-hop neighborhood* of a vertex v of a graph $\mathcal{G}$ is the set $N^2(v)$ composed by the union of the neighborhoods of all the neighbors of v. That is $N^2(v) = \bigcup_{u \in N(v)} N(u)$.

Note that $v \in N^2(v)$ but $v \notin N(v)$.

By extension, we can define the *n-hop neighborhood*, which is defined by recurrence based on the *(n-1)-hop neighborhood*.

Definition 2.6 (Path)

A *path* $\mathcal{P}(\mathcal{G})$ in the graph $\mathcal{G}$ is a linear sequence of distinct vertices $\{v_0, ..., v_n\}$ so that two consecutive vertices in the path sequence are adjacent in the graph $\mathcal{G}$.

The *length* $l(\mathcal{P})$ of a path is simply the number of vertices in the sequence, the origin of the path excluded.

Definition 2.7 ((X,Y)Path)

An *(X,Y)Path* is a path $\mathcal{P}(X,Y)$ of the graph $\mathcal{G}$ that starts at vertex $X \in V(\mathcal{G})$ and ends at vertex $Y \in V(\mathcal{G})$.

The notion of path, and especially (X,Y)Path, is important in communication networks like wireless ad hoc networks as the existence of a path between two vertices means that those two entities will be able to exchange information. There are therefore many algorithms for finding a shortest path between two vertices (see for instance [For56, Bel58, Dij59]) as this allows to minimize the number of intermediate retransmissions.

Definition 2.8 (Shortest path)

A *shortest path* $\mathcal{P}_{min}(X,Y)$ between two vertices X and Y of the graph $\mathcal{G}$ is the (X,Y)Path with smallest length.
We note $l(X,Y) = l(\mathcal{P}_{min}(X,Y))$ its length.

The length of the longest shortest path of a graph $\mathcal{G}$, ie $\max_{(u,v) \in V(\mathcal{G}) \times V(\mathcal{G})} l(u,v)$, is the graph *diameter*.

Definition 2.9 (Connected component)

A connected component $\mathcal{S}_{cc}$ of the graph $\mathcal{G}$ is a subgraph of $\mathcal{G}$ in which there exists an (X,Y)Path for all pair of vertices (X, Y) in $V(\mathcal{S}_{cc}) \times V(\mathcal{S}_{cc})$.

2.1.2.2 Degree and density

In the sequel of this document, we will often refer to the number of neighbors that a vertex has, as well as the global implication of the number of neighbors, and hence edge, on network metrics. Hence, it is legitimate to introduce the following definitions.

Definition 2.10 (Degree)

The *degree* of a vertex v of the graph $\mathcal{G}$, noted $d_{\mathcal{G}}(v)$ or $d(v)$ is the number of edges of $\mathcal{G}$ incident to v. If $\mathcal{G}$ is a simple undirected graph, then this is simply the number of adjacent vertices or neighbors of vertex v.

Definition 2.11 (Graph average degree)

The *graph average degree*, often noted $\Delta(\mathcal{G})$ or Δ, is the average degree over all the vertices of the graph. That is:

$$\Delta(\mathcal{G}) = \frac{1}{N} \sum_{v \in (\mathcal{G})} d(v) = \frac{2M}{N} \tag{2.1}$$

Definition 2.12 (Graph density)

The *graph density*, often noted $\Gamma(\mathcal{G})$ or Γ, is the ratio of the number of vertices M of the graph $\mathcal{G}$ over the total number of possible edges. That is, for a simple undirected graph:

$$\Gamma(\mathcal{G}) = \frac{2M}{N(N-1)} \tag{2.2}$$

It is here important to note that the average degree and the density measures are global over the graph. As we will see later in this document (see chapter 3), network density is not necessarily homogenous throughout the graph.

2.1.3 Weighted graphs

The previously-presented definitions assumed that all vertices and edges are equal and support only two states : either they exist in the graph or they are absent from it.

However, the modeling of a problem may require that vertices and/or edges are given different importance, or that an *attribute* or *label* shall be given different values, depending on the considered vertex or edge. Such graphs are called *weighted graphs* and the corresponding attributes are often referred to as vertex or edge *weight*.

2.1.3.1 Examples of weighted graphs

For instance, if a weighted graph abstracts from different cities (the vertices) and their connections through roads (the edges), then the vertex weight can be the size of each city and the edge weight the distance between them. This instance of weighted graph can then be used for solving a modified version of the traveling salesman problem (see [Bel62, Kar82]) where biggest cities have to be visited as early as possible.

In communication networks, weights are often used to reflect the communication capacity or quality, or even other parameters depending on the use case and the optimization objectives. For instance, vertex weight can be the maximum throughput achievable by the considered communicating devices, or its remaining battery level in the case of mobile wireless networks. Edges weight is most of the time used to represent the throughput, transmission delay, or quality (bit error rate, signal-to-noise ratio for wireless networks).

2.1.3.2 Extensions of definitions

Most of the concepts introduced in subsection 2.1.2 can be easily extended to take into account weighted values on the various graphs elements.

For instance the notions related to adjacency and degree can be adapted as follows:

Definition 2.13 (Weighted neighborhood)

The set of tuples $N_w(v) = \{(u, w_u)/u \in N(v)\}$ that associates each adjacent vertices u of v with either the weight of vertex u or the weight of the edge between u and v is called the *weighted neighborhood* of v.

As for the unweighted neighborhood, the weighted neighborhood can be extended to *n-hops* (see subsubsection 2.1.2.1).

Definition 2.14 (Weighted degree)

The *weighted degree* of a vertex v of the graph $\mathcal{G}$, noted $d_{w,\mathcal{G}}(v)$ or $d_w(v)$ is the sum of weights of either all adjacent vertices of v in $\mathcal{G}$, or all edges of $\mathcal{G}$ incident to v.

Definition 2.15 (Weighted average degree)

By extension, the *weighted average degree* of a graph $\mathcal{G}$, noted $\Delta_w(\mathcal{G})$ or Δ_w, is computed as:

$$\Delta_w(\mathcal{G}) = \frac{1}{N} \sum_{v \in V(\mathcal{G})} d_w(v). \tag{2.3}$$

Definition 2.16 (Weighted path)
A *weighted path* $\mathcal{P}_w(\mathcal{G})$ in the graph $\mathcal{G}$ is a linear sequence of tuples $\{(v_0, w_0), ..., (v_n, w_n)\}$ where v_i is the i^{th} vertex in the associated path and w_i is either the corresponding weight, or the weight of the vertex from vertex v_{i-1} to vertex v_i.

The same way, *weighted paths* now associate the appropriate weight to each element of the path. This allows to consider not only shortest paths (see subsubsection 2.1.2.1) but also paths whose average, minimum or maximum weight is the smallest or the largest, depending on the chosen meaning of this attribute.

2.2 From graphs to dynamic networks

As we have seen, graphs are a useful abstraction for communication networks, especially wireless ad hoc networks. In this case, the graph represents the network topology, i.e. the communicating devices and the set of wireless links that they can use to communicate with each other.

However, the sole graph definition from graph theory, as presented in definition 2.1 is not sufficient to capture completely the complexity of wireless ad hoc networks. Particularly, the communication capabilities are highly dependent on the geographical distribution of nodes (as radio waves are used as communication channel).

Besides, we insisted on the fact that the involved communicating devices are mobile. As a consequence, and because communication links are correlated with the distance between their endpoints, the graph is very likely to evolve over time. Ad hoc wireless networks are also subject to fluctuations in the communication quality over time and the appearance and disappearance of communicating entities, for instance due to battery drain.

2.2.1 Wireless communication networks

The term of network is used in many different context. In pure graph theory, it is often associated to flows problems. It is also sometimes referring to graphs in which dynamics are considered[Ber05], not to mention the various notions related to the application of graph theory to telecommunications, natural networks, etc.

For sake of clarity, we will formally define the notion of *network* as graphs where the position of nodes is important, complementing our abstraction based on graphs.

Definition 2.17 (Network)
A *network* $\mathcal{N}$ is a couple $(\mathcal{G}, L)$ where $\mathcal{G}$ is a graph and $L : V(\mathcal{G}) \rightarrow \mathcal{R}$ is a *location function* that associates each vertex v of $\mathcal{G}$ to its position in $\mathcal{R}$, a d-dimensional space.

The network $\mathcal{N}$ is said to be *bounded* to region $\mathcal{R}$. When real-world networks are abstracted, $\mathcal{R}$ is generally of dimension $\mathrm{d} = 3$ or $\mathrm{d} = 2$ (the height or altitude of communicating devices is then ignored).

In a network, vertices are generally called *nodes*, while the edges are referred to as *links*.

2.2.1.1 Mobility

The location function L does not necessarily assign nodes to a fixed position, especially when the network abstracts from a scenario when communicating nodes are mobile. In this case, the location function assigns a different position to nodes at each time $t \in \mathbb{T}$ and is called a *mobility model*.

Definition 2.18 (Mobility model)
A *mobility model* for a network $\mathcal{N} = (\mathcal{G}, L)$ is a function $L_{dyn} : V(\mathcal{G}) \times \mathbb{T} \to \mathcal{R}$ that assigns a time-dependent position in $\mathcal{R}$ to each node of $\mathcal{N}$.

We will dedicate chapter 7 to the analysis of mobility models, especially in an urban environment.

2.2.1.2 Transmission function

In subsection 2.1.2 we have introduced the notion of incidence function that distributes edges over a graph $\mathcal{G}$. As the location function, this can be refined to integrate the dependence on node positions of the wireless transmissions. We therefore introduce the *transmission function*.

Definition 2.19 (Transmission function)
A *transmission function* T for a network $\mathcal{N} = (\mathcal{G}, L)$ is a function that for each pair of located nodes $((u, L(u)), (v, L(v)))$ of the network $\mathcal{N}$ decides whether or not there exists a link between u and v.

The transmission function takes into account not only the characteristics of the transmission channel, which is determined by the chosen propagation model, but the complete transmission chain that makes propagation of information possible over the air.

Although this does not appear directly, the transmission function can be time-dependent, two connected nodes u and v can be disconnected at a different time, whether or not their respective locations $L(u)$ and $L(v)$ are unchanged.

The simplest transmission functions are solely distance-based. The simplest example is called the *Unit Disk Graph* (UDG) model introduced by Huson and Sen in [SH96, SH97]. In this model edges are set if and only if the distance between its two endpoint nodes is lesser than the unitary distance measure. A network based on the unit disk graph model is illustrated in Figure 2.2.

We will investigate the importance of transmission models in chapter 6.

2.2.1.3 Formal definition

Finally we can define formally our model for wireless communication network:

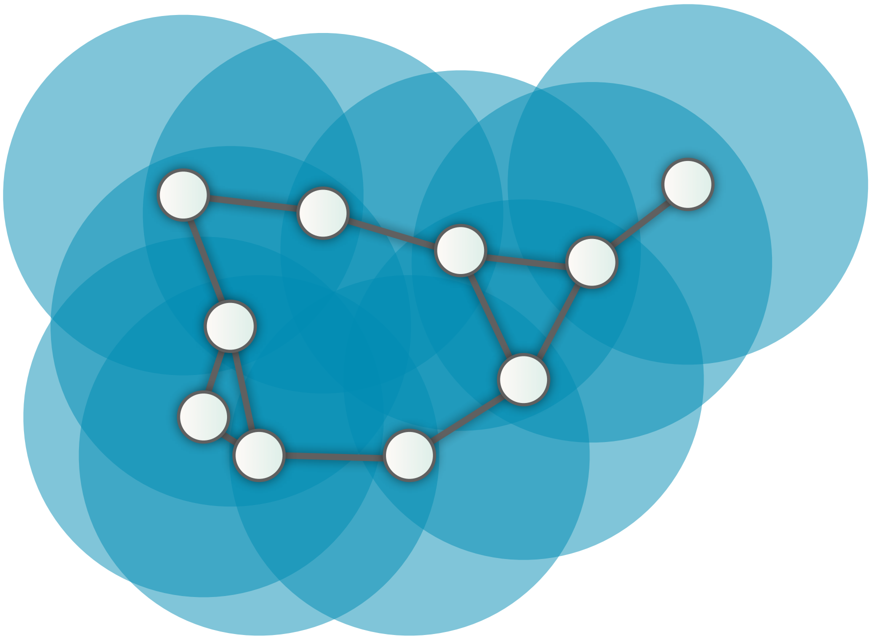


Figure 2.2: A network based on the unit disk graph model.

Definition 2.20 (Wireless communication network)
A *wireless communication network* is an ordered three-tuple $(\mathcal{G}, L, T)$ where $\mathcal{G}$ is a graph, L is a location function and T a transmission function so that the vertices $V(\mathcal{G})$ are placed according to L and the edges $E(\mathcal{G})$ are established using the transmission function T.

From now on, and for the remainder of this document, we will indifferently refer to wireless communication networks as networks or graphs.

2.2.2 Dynamic graphs

The previous definition of a wireless communication network will, when it involves mobile nodes, rely on a time-dependent location function L_{dyn}. Besides, we have reckoned that the transmission function T can be also modeled as being time-dependent.

Those components are necessary to capture the following aspects of wireless ad hoc networks in the abstraction model:

- the **mobility of nodes** that will create and tear down wireless communications channels based on the relative distance between communicating nodes;
- the **arrival and disappearance of nodes** that may enter the network or stop operating due to failure or battery drain for instance;
- the **changed in channel conditions** that are due to sporadic interferences or modifications of the nodes environment.

2.2.2.1 Definition

Graphs, as fixed mathematical structures found in graph theory, fail to capture the interactions of evolving systems in many scientific fields such as physics, sociology or chemistry.

They would require a time reference (or a precedence relation) and the ability to evolve over time to become dynamic graphs. Those structures are characterized by *simple dynamics* (the evolution of the valuation of vertices and edges over time) and *meta-dynamics* [Ber05] (the appearance and deletion of components of the graph).

This induces the following definition for a dynamic graph, formalized by Ferreira in [Fer02]:

Definition 2.21 (Dynamic Graph - Pigné (Pig08))
A dynamic graph $\mathcal{G}$ is a three-tuple $(\mathcal{G}_0, \mathbb{T}, P)$ so that :

- $\mathcal{G}_0 = (V_0, E_0)$ is the initial static graph, potentially empty;
- $\mathbb{T}$ is a temporal base, discrete or continuous;
- P is an evolution process.

The static image of the graph $\mathcal{G}$ à time t is noted $\mathcal{G}_t$.

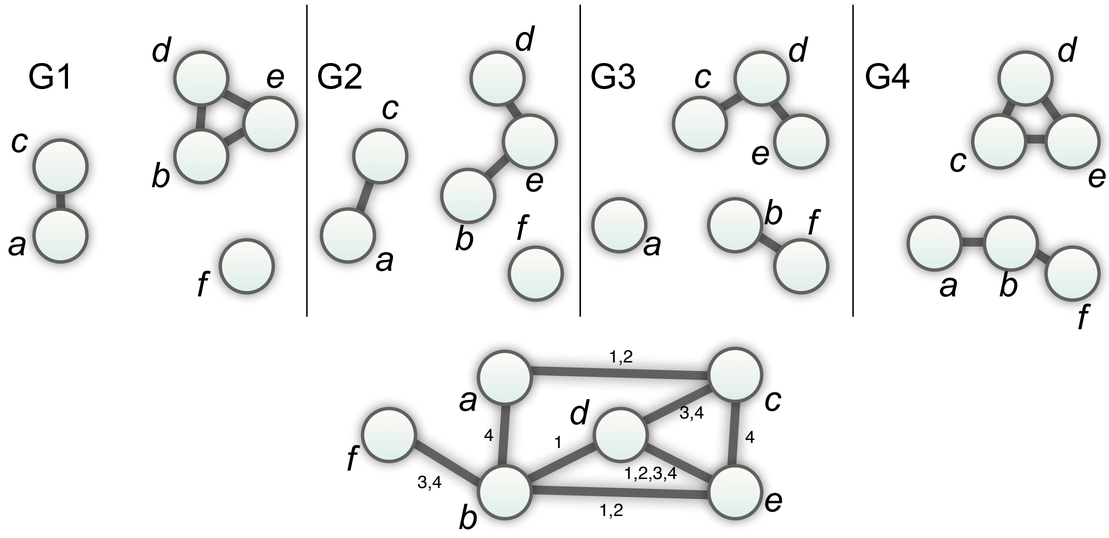


Figure 2.3: Example of static graphs (top) representing snapshots of a dynamic graph (bottom)

Once extended to networks, and by setting the evolution process P as the composition of a mobility model L_{dyn}, presented in definition 2.18, and a transmission function T, also dynamic -see subsubsection 2.2.1.2-, it is obvious that this abstraction is perfectly suitable to model dynamicity in wireless communication networks.

2.2.2.2 Models

Many practical models have been designed to favor the usage in various dynamic problems. They are extensively presented in subsubsection 2.2.1.2. We must however consider:

- **Space-Time Networks** were used by Pallottino and Scutella [PS97] to find the time-optimum journey on roads with varying travel time. However this network models allows only simple dynamics (only weights valuation can change) and is not suitable for modeling wireless communication networks;

- **Cumulative Graphs**, presented by Cortes *et al.*[CPV03], were created to model a large wire-based telecommunication system. This model only allows the addition of new nodes and links; however it can be extended so that graph elements can be ignored if their valuation is above or below a given threshold;

- **Evolving graphs** introduced by Ferreira *et al.*[Fer02, BXFJ03] allow complete dynamics as both vertices and edges can be added or removed and their valuations changed over time;

Evolutive graphs appear as a suitable model for wireless communication networks. They are defined as follows:

Definition 2.22 (Evolutive graph)

Let $\mathcal{G} = (V, E)$ a graph. Let $\mathcal{S}_{\mathcal{G}} = \{\mathcal{G}_0, \mathcal{G}_1, ..., \mathcal{G}_k\}$ an ordered set of subgraphs of $\mathcal{G}$. Then $(\mathcal{G}, \mathcal{S}_{\mathcal{G}})$ is an evolutive graph.

This practical model can hence be interpreted as a set of snapshots of the considered graph taken at instants $0, 1, ., k \in \mathbb{T}$, either regularly spaced in time or not. However, it requires the temporal base $\mathbb{T}$ to be discrete.

2.2.2.3 Extensions

As they introduce the notion of time in the graph abstractions, dynamic graphs open new considerations for the history of graph elements as well as their evolution over time.

In definition 2.6 we have introduce the notion of path, i.e. the possibility for two nodes to communicate through intermediate nodes. This required all links between intermediate nodes, from the original node to the destination to exist at the same time. In dynamic graph a more generic definition, allows for messages to travel along a path at different time, when the link to the next intermediate node is available. This is the notion of *journey*, introduced by Bui-Xuan, Ferreira *et al.* in [BXFJ03]

Definition 2.23 (Journey)

Let $\mathcal{J} = \{e_0, e_1, e_2, ..., e_k\}$ a set of edges of a dynamic graph $\mathcal{G}$. Let $\mathbb{T}_{e_j}$ the set of time intervals where edge e_j exists in $\mathcal{G}$. Then $\mathcal{J}$ is a *journey* if and only if there exists $\{t_0, t_1, t_2, ...t_k\}$ so that $t_0 \leq t_1 \leq t_2 \leq ... \leq t_k$ and $t_j \in \mathbb{T}_{e_j}, \forall j \in [0..k]$.

The notion of *journey* is important in a subset of wireless ad hoc networks, called *delay tolerant networks* where information can be stored for some time and transmitted later, in order to improve the number of nodes that have received the messages, even after a large delay since original emission.

2.2.2.4 Metrics

As well, the monitoring evolution of the graph over time allows the definition of new metrics, aimed at measuring the longevity or stability of the graph elements over time. Those metrics can be used as *weight* functions on those elements. Most of them were introduced by Pigné in [Pig08].

Definition 2.24 (Age)
The *age* of an element of $\mathcal{G}$ in the difference between the current date in $\mathbb{T}$ and the date at which this element appeared. If the element is not existing in $\mathcal{G}$ at the current date, its age is 0.

Definition 2.25 (Cumulated age)
The *cumulated age* of an element of $\mathcal{G}$ is the sum of the length of all time intervals of $\mathbb{T}$ during which it exists in the dynamic graph $\mathcal{G}$.

Definition 2.26 (Volatility)
The *volatility* of an element of $\mathcal{G}$ is the ratio of the number of times it appeared in $\mathcal{G}$ and its cumulated age.

Definition 2.27 (Average age)
The *average age* of an element of $\mathcal{G}$ is the inverse of its volatility.

All those metrics can be extended to any subgraph of $\mathcal{G}$ by taking the average values of the chosen metric over all the elements of the considered subgraph.

2.3 Highlight

- In this chapter, we have introduced the key notions of **graph theory** that will be used throughout this document;
- We also extended these notions to **networks and dynamic graphs**, so that they are suitable for modeling wireless communication networks;
- This extension requires a definition of a **mobility model** that will determine the position of the networks nodes over time;
- Based on the relative node position, a **transmission function**, that can be time-dependent, will govern the existence of communication links between nodes;
- Considering the time-based evolution of elements in dynamic graphs allows for the definition of **stability metrics** that reflect the history of those components.

We will now further investigate some particularities of graphs and their implications on social network analysis, as well as the associated techniques.

Contents

Chapter 3

Social network analysis

> "Our acquaintances -not our friends- are our greatest source of new ideas and information."
>
> Mark Granovetter, The Strength of Weak Ties.

As we have seen in subsection 2.1.1, most work in graph theory was developed to answer scientific questions. Graph enumeration allowed chemists to classify molecular structures, and biologists use coupled dynamical systems to study everything from neural networks to the synchronous chirping of crickets. Graphs have also appeared in sociology, as in Harary work on social status in a hierarchy[Har59] and Granovetter work on weak ties[Gra73].

Those studies not only offered noticeable breakthroughs in their respective domains, but they also allowed to further understand and describe the graph structure itself. The problem is to create relevant graph topologies, that virtuously capture the reality, so it is possible to draw righteous conclusions, and design appropriate solutions, by reasoning on the mathematical structure.

This better understanding of graph topology models in mainly due to social network analysis, a sociological research field that uses graphs as abstraction of human relations to study their impact on various social phenomena.

Particularly, we will focus on the concept of *community* that emerged from social network analysis. Later in this manuscript, namely in chapter 6 and chapter 7, we will see how it is relevant to the modeling of wireless ad hoc networks in a urban environment.

3.1 Graph topology models

For the study of network characteristics in general, different graph models may be proposed. In this chapter we consider random graph models, the regular lattice model and the scale-free model. Although knowledge of all these models is essential completeness, it will become clear that not all of these models are equally suitable to characterize wireless multi-hop ad hoc networks.

3.1.1 Random models

Random graph models a generic method to easily generate graph with certain global properties: order, size, average degree, etc. The generation function can generally be expressed as a mathematical stochastic function, which makes it easy to generate large graphs with the help of the computer.

However, those models, despite their interest in formal analysis of properties, fail to capture the subtlety of complex systems like social interactions.

3.1.1.1 Erdös and Rényi random graph model

Although many random graph models exist, the most straightforward and best studied is the one proposed by Erdös and Rényi in [ER60] and presented in Figure 3.1.

Definition 3.1 (Random Graph Model - Erdös and Rényi (ER60))
A *random graph* $\mathcal{G}_{er}$ whith N vertices and M edges is constructed by starting with N vertices and zero edges. Then M edges are chosen randomly and independently from the $\frac{N(N-1)}{2}$ possible edges.

As per definition 3.1, there are at most $\binom{N(N-1)/2}{M}$ equiprobable graphs $\mathcal{G}_{er} = (V, E)$ with $|V| = N$ and $|E| = M$.

Random graphs are also often referred to by using the probability p for an edge to be created, and noted $\mathcal{G}_p(N)$.

An interesting aspect of random graphs is the existence of a critical probability p_c at which a giant cluster forms. A giant cluster $\mathcal{S}_{c,max}$ is a connected component (see definition 2.9) of $\mathcal{G}$ that contains most of vertices of $\mathcal{G}$. This means that at low values of p, the random graph consists in isolated subgraphs. When the value of p increases, above a threshold value a giant cluster emerges that spans almost the entire network. This phenomenon is referred to as *percolation transition*.

3.1.1.2 Chung and Lu random graph model

Chung and Lu provide another generalized random graph model with rigorous analysis in [CL02, CL03]. The newly added formalism is to construct random graphs with given arbitrary expected degrees: they assign weights to the nodes and have an independent random link between any two nodes with a probability proportional to the product of the weights of these two nodes.

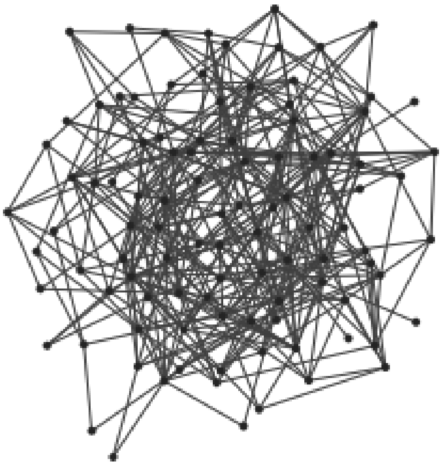


Figure 3.1: An example of random graph with the Erdös-Rényi model

Thus, the expected degree of each node is specified by its weight. Chung and Lu then extend their approach to also model the clustering effect. They propose a hybrid model where they couple a power-law random graph, as a global graph, with a specific local graph rich of local links [CL04].

3.1.1.3 Geometric random network models

Our definition of a wireless ad hoc network consists in a set of nodes, dispersed on a geographical region, connected to other devices in their vicinity, depending on the capabilities of the radio channel used for communication. As a consequence, those networks can not be modeled as pure random networks. Geometric random network models try to cope with this limitation in introducing a correlation between the distance between two vertices and the probability that they are connected.

Definition 3.2 (Geometric Random Network)

A *geometric random network* is a network $\mathcal{N}_{p(r_{u,v})} = (\mathcal{G}_{p(r_{u,v})}, L)$ of N nodes where $p(r_{u,v})$ is the probability of having a link between node u and node v at a distance $r_{u,v}$.

We assume in a geometric random network that the N nodes are uniformly distributed over the entire service region $\mathcal{R}$ of $\mathcal{N}_{p(r_{u,v})}$.

Note that geometric random network model is simply a subclass of networks (as presented in definition 2.17) where the location function L is random, and the transmission function T is stochastic and depends on $p(r_{u,v})$. If $p(r_{u,v})$ is so that $p(r_{u,v} \leq r_{max}) = 1$ and $p(r_{u,v} > r_{max}) = 0$, we have an *Euclidean graph model*. If $r_{max} = 1$, this is the unit disk graph model[SH96].

Geometric random network appear as more suited to represent wireless networks, but they also hold meaning in wired networks. For example, Faloutsos *et al.*[FFF99] observed in the Internet topology, that a ball of neighbors within distance r has size approximated by r^{β}, $\beta \in \mathbb{R}^{+}$, when the radius distance r is small enough. Yook *et al.* also observed a correlation between Internet nodes and the population density in the areas that are economically homogeneous[YJB02].

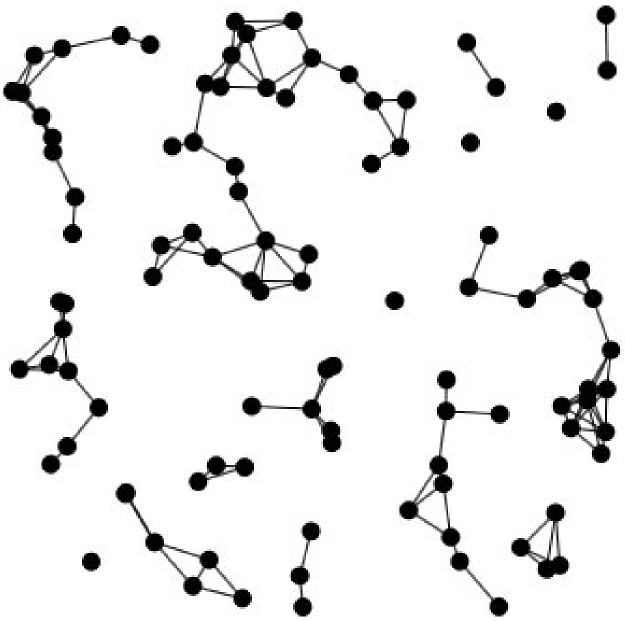


Figure 3.2: An example of a Random Euclidean graph

3.1.2 Regular lattice graphs

Regular lattice graphs, often referred to as *grid graphs*, take the opposite approach: nodes are not placed at random any more but are regularly spread on a grid.

Definition 3.3 (Regular lattice graph)
A *regular lattice graph* $\mathcal{G}_{lat}(N, p)$ whith N nodes is constructed with nodes on a regular grid structure. Grid-adjacent nodes (sharing the same row or column of the grid) are all equidistant. The probability that two adjacent nodes are connected is p.

Figure 3.3 shows an example of such a regular lattice graph for various values of p.

We see that the lattice model and ad hoc networks share the notion that the distance between nodes influences the link probability. From this point of view, the lattice model is more suitable to represent an ad hoc network than the random graph model discussed previously. However, the position of nodes in an ad hoc network is generally not fixed to a regular lattice. Further, in radio communication the distance over which nodes can be connected to each other is not a fixed value.

3.1.3 Scale-free networks

While other models presented so far allowed to set various network parameters (size, order, average degree), the stochastic processes made formally no difference between the nodes in the attribution of edges. This binomial degree distribution, found in random graphs, seems to be an unrealistic assumption for large real-world networks, like the Internet[FFF99], that follow a

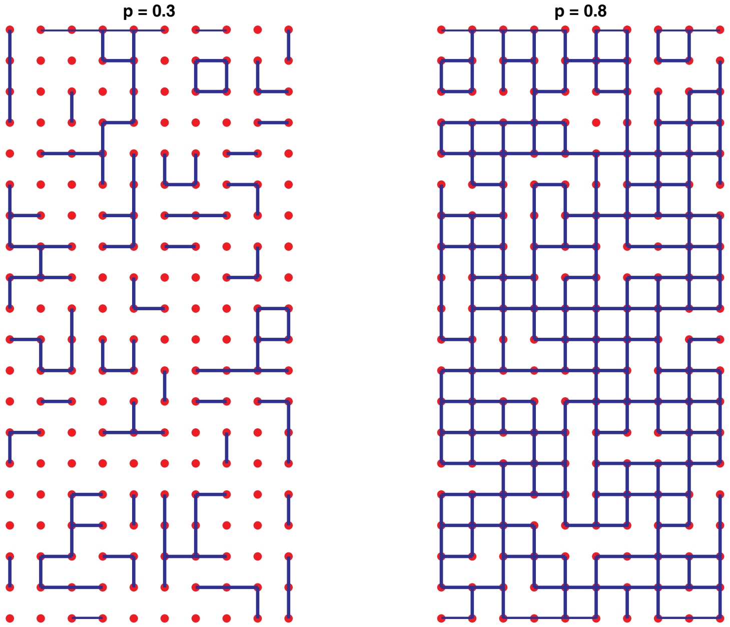


Figure 3.3: A 2-dimensional 10×20 lattice graph with $p = 0.3$ and $p = 0.8$

power-law.

3.1.3.1 Definition and properties

Those networks whose degree distribution follows a power-law of constant parameter are called *scale-free* as the degree distribution is independent of the graph order.

Definition 3.4 (Scale-free graph)

A *scale-free graph* $\mathcal{G}_{sf}(N, \alpha)$ whose degree distribution follows a power-law of parameter α, independent of the graph order, that is:

$$\forall v \in \mathcal{G}_{sf}(N, \alpha), \mathrm{Pr}(d(v) = k) = k^{-\alpha} \tag{3.1}$$

where Pr denote the probability operator

The power-law degree distribution influences the way in which the network operates, including how it responds to catastrophic events. A scale-free graph, where a very small number of network nodes (called *hubs*) are far more connected than other nodes, shows striking resilience against random breakdowns. In scale-free networks, in spite of large sizes of the networks, the distances between most vertices is short because these paths usually go through the hubs.

This explains why most of large international airline companies are using a set of airports as hubs, offering both a service with a minimal number of connections to final destination and resilience to random problems (weather condition, strikes, etc.) impairing airports, while limiting the number of required flight lines (see Figure 3.4).

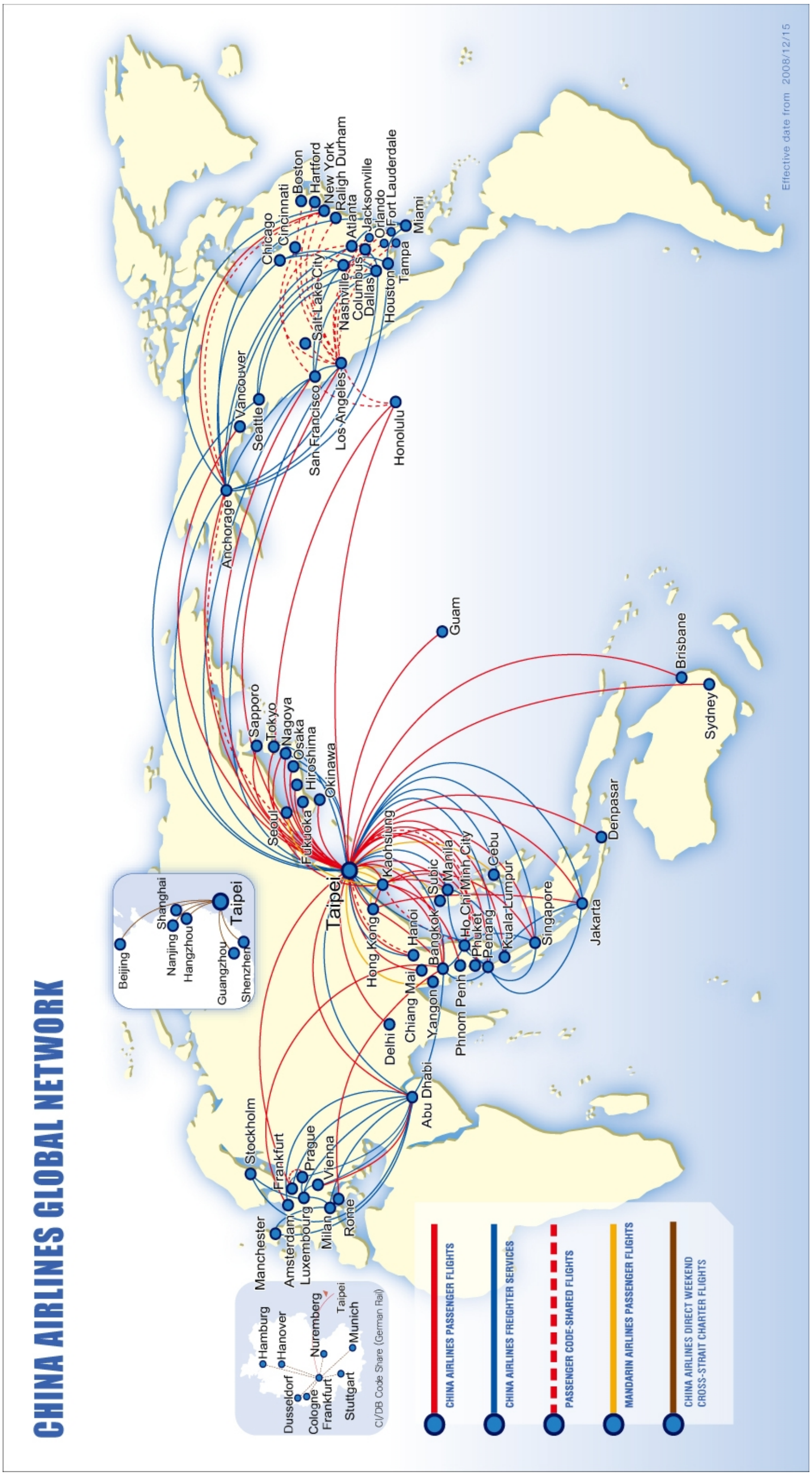


Figure 3.4: China Airlines uses Taipei as main hub; Abu Dhabi, Anchorage, Los Angeles, Singapore, Jakarta as secondary hubs for its international flights

3.1.3.2 Barabasi and Albert model

A specific method for generating a scale-free network is a process in which vertices are added to a graph one at a time and joined to a fixed number of earlier vertices, selected with probabilities proportional to their degrees.

This generation method, called *preferential attachment*, was first introduced by Barabasi and Albert in [BA99]. They start with a fixed number of nodes N and continuously add vertices to the graph one at a time, with edges joining older vertices, that are selected with a probability proportional to their degree. This process clearly favors the generation of hubs with large degree.

3.1.3.3 Evolving networks

The model presented above still present discrepancies with real-world networks: the distribution exponent is fixed, so is the number of links created when a new node is added.

The leverage of these constraints leads to a family of *evolving networks* [Fer02, BXFJ03], where more variables are added, corresponding to growing factors observed in real networks. For example, non-linear functions for preferential attachment can be considered. Alternatively, the addition of an initial attractiveness to each node, reflecting that in real networks each node has some chance to be discovered and linked to even if it has initially no edge.

3.2 Social network analysis

Social network analysis has emerged as an important technique in sociology and abstracts social relationships (friendship, collaboration, acquaintance) using the paradigms of graph and network theory.

Research in a number of academic fields has shown that social networks operate on many levels, from families up to the level of nations, and play a critical role in determining the way problems are solved, organizations are run, and the degree to which individuals succeed in achieving their goals. Its usage has also spread to related fields like anthropology, biology, geography, etc. Lately, with the emergence of web-based social networks applications, it has gained an additional perspective in the investigation of characteristics of online social phenomena and communities.

As in this thesis we envision the application of wireless ad hoc networks to the development of such applications in a ubiquitous form, it appeared as legitimate to us to investigate this field, in search for useful notions in the design of our solution.

3.2.1 Concrete example

As a concrete example, we will consider the most simple social structure: a graph $\mathcal{G}$ representing acquaintances (the edges) among people (the vertices). We will consider two persons as adjacent if they know each other, say on a first-name basis.

3.2.1.1 Observations

The order of our graph is quite large, as $|V(\mathcal{G})| = N \approx 6 \times 10^9$ (the size of the global population), while its average degree Δ is rather small, as we only know a handful (up to several hundreds) of people by their first name. This results in a *sparse* graph.

Definition 3.5 (Sparse Graph)
A graph $\mathcal{G}$ is *sparse* if $\Delta(\mathcal{G}) \ll |V(\mathcal{G})|$.

Due to its sparse nature and its extended size, one might expect to travel a large path of intermediate acquaintances to connect two random people. However, in the 1960s, psychologist Stanley Milgram[Mil67] conducted experimental studies by having letters travel between two complete strangers through as few links as possible. He found that on average, it took only about six intermediaries, a very small number compared to the total number of people in the experiment. This is the experiment we exposed in the subsection 1.2.1 of our introduction.

The question is what model of network allows finding such small paths in such a large sparse graph. Capturing its characteristics may help understanding why and how information could travel fast, and if possible reliably, on large communication networks.

3.2.1.2 Metrics

In order to characterize this social network model, we will introduce first a metric that represents its specificities: the *characteristic path length*, as we have observed it unexpectedly small in social networks:

Definition 3.6 (Characteristic Path Length)
The *characteristic path length* $\bar{l}(\mathcal{G})$ of a graph $\mathcal{G} = (V, E)$, also noted $\bar{l}$ is the average of the length of all shortest paths in $\mathcal{G}$.
That is:

$$\bar{l}(\mathcal{G}) = \underset{(u,v)\in V(\mathcal{G})\times V(\mathcal{G})}{\operatorname{avg}} l(u, v) \tag{3.2}$$

Besides, and as noted in subsection 1.2.1, an important factor can be the number of common relations that two individuals may have. This is reflected by the *clustering coefficient* of a network.

Definition 3.7 (Clustering coefficient)

Let $E_{N(i)} = \{e = (u,v)/(u,v) \in N(i) \times N(i)\}$ the set of edges of $\mathcal{G}$ that have both endpoints in the neighborhood of i.
Then the *clustering coefficient* of node i is computed as:

$$\gamma(i) = \frac{|E_{N(i)}|}{\binom{d(i)}{2}} \tag{3.3}$$

And the *clustering coefficient* of a graph $\mathcal{G}$ is defined as:

$$\overline{\gamma}(\mathcal{G}) = \operatorname*{avg}_{v \in V(\mathcal{G})} \gamma(v) \tag{3.4}$$

The clustering coefficient is thus the ratio between the actual number of links between the neighbors of node i and the maximum possible number of links between these neighbors. In other words, the clustering coefficient is the ratio between the number of triangles that contain i and the number of triangles that would contain i if all neighbors of i were interlinked.

The network presented in Figure 3.5 has a characteristic path length $\overline{l} = 1.2$ and an average clustering coefficient $\overline{\gamma} = 0.67$.

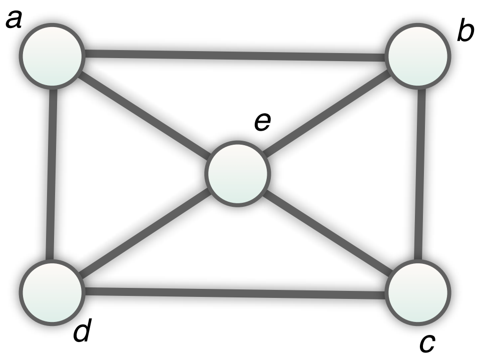


Figure 3.5: A graph with $\overline{l} = 1.2$ and $\overline{\gamma} = 0.67$.

3.2.1.3 Suitability of common graph models

Using the previously defined metrics, we can now assess whether the graph topological models we have introduced in section 3.1 correspond to the expected characteristics of social networks.

Random graphs have a Poisson distribution of nodes degree around an average degree $\Delta = (N-1)p$, where p is the probability for an edge to be created[Hek06].

For sparse graphs, i.e. Δ sufficiently small compared to N, the number of reached nodes after h hops is approximately Δ^h, all nodes being reached when $\Delta^h \approx N$. This lead to a characteristic path length $\overline{l_{rand}} \approx \log(N)/\log(\Delta)$. More precise approximation have been given[AB02], but in any case $\overline{l_{rand}} = O(\log(N))$.

Because clustering coefficient is the percentage of neighbors of a node that are connected to each other, and in a random graph links between nodes are established independently with probability p, we may expect the clustering coefficient in a random graph to be $\overline{\gamma_{rand}} \approx p$.

As a consequence, sparse random graphs have small characteristic path length, but insufficient average clustering coefficient to model social graphs.

Regular graphs are naturally sparse for large values of N, even if the probability p of two grid-adjacent nodes to be connected is high. For $p \approx 1$, we have $\Delta \approx 4$ as the degree of nodes is also constrained by their position. If we allow *diagonal* links, then $\Delta \approx 8$.

Even under those assumptions, we can easily verify that the clustering coefficient $\overline{\gamma_{lat}} = 0$. Allowing *diagonal* links in the lattice would then yield $\overline{\gamma_{lat,diag}} \approx 2/3$.

To compute the characteristic path length, we will consider that the N nodes are disposed on a $r \times c$ lattice. For $c = 1$, all nodes are in a line of length r. In this configuration there are $h - 1$ paths of length h (for $1 \leq h \leq r - 1$) and the characteristic path length is $\overline{l_{r\times 1}} = (r+1)/3$. This result can be extended to two dimensional grids by saying that the hop count can be decomposed on the rows or the columns of the grid, that is $h = h_r + h_c$. It follows, $\overline{l_{r\times c}} \approx (r + c)/3$. For large values of N, $r \approx c \approx \sqrt{N}$ and $\overline{l_{lat}} = O(\sqrt{N})$. We can extend the previous result when *diagonal* links are allowed, by saying that, at best, those links diminish the length of the shortest path between two nodes by 2, so $\overline{l_{lat,diag}} = O(\sqrt{N})$.

So, in any case, the characteristic path length of regular graphs is polynomial with the order of the graph, so this model does not fit with social networks observations.

Scale-free networks despite their large order, have short path length between most vertices because these paths usually go through the *hubs*. It can be shown[CHbA05] that the characteristic path length of scale-free networks $\overline{l_{sf}} = \ln(\ln N)$.

Besides, another important characteristic of scale-free networks is the clustering coefficient distribution, which decreases as the node degree increases. This distribution also follows a power law. This implies that the low-degree nodes belong to very dense sub-graphs and those sub-graphs are connected to each other through hubs.

3.2.1.4 Small world graphs

Random graphs and regular lattice graphs could be considered two extremes in a continuum of possibilities, from the highly structured (with possibly high clustering coefficient but with large characteristic path) to the highly random (that have opposite characteristics).

In 1998, Watts and Strogatz[WS98] proposed another graph topology model, called the WS model, constructed by randomly rewiring the edges of a lattice each with a probability parameter β. With β varying, the model's structural properties can be considered through the two characteristics we are investigating. For $\beta = 0$, the lattice nature of the graph is conserved, while for $\beta = 1$ a purely random graph is constructed. For intermediate values of β, the rewired links, acting as *shortcuts* lower the characteristic path length, while the clustering coefficient is maintained.

So there do exist networks that have the low characteristic path lengths of random graphs but much larger clustering coefficients than a random graph would have. Watts and Strogatz named them *small-world graphs*.

Definition 3.8 (Small World Graph)
A graph $\mathcal{G}_{sw}$ is said to be a *small-world graph* if it has both:

- a small characteristic path length $\bar{l}(\mathcal{G}_{sw})$
- a high clustering coefficient $\overline{\gamma}(\mathcal{G}_{sw})$

Besides, we have seen in section 3.2.1.3 that the small-world property is more strongly present in scale-free networks than in other graph models.

3.2.2 Clusters and hubs

In the previous section, we have seen that the small-world properties, that are found in most social networks, are compatible with the scale-free topology model. This is especially the case when we envision the network as composed of small and tightly interconnected sets of nodes (that will increase the network clustering coefficient) that are linked to other sets through hubs (to keep the characteristic path short).

This intuition of the fragmentation of the society in several clusters of individuals having similar social characteristics (such as ethnic origin, income or interest) has been confirmed in the work realized by sociologist Mark Granovetter[Gra73].

3.2.2.1 Triadic closure

First, we shall focus on a social relationship rule, which is known as the *triadic closure rule*, formalized in [Rap57] by Rapoport:

Definition 3.9 (Triadic Closure Rule - Rapoport(Rap57))
In a social relationship network, if two individuals A and B are connected, i.e. share a social relation, and A is also connected with a third individual C, then it is very likely that B and C are also connected.

This rule, which is simply a sociological transcription of the clustering coefficient presented in definition 3.7, explains by itself one of the small-world characteristic of social networks.

3.2.2.2 Weak ties

But would the triadic closure rule be absolutely respected, all social graphs would be complete (i.e. each vertex would be connected to any other vertex), which is not the case in real world. The originality of the work of Granovetter is to focus especially in the cases where the triadic closure is not respected.

For the second small-world characteristic, small characteristic path length, to be present, it is necessary that the deeply interconnected sets of strongly tied friends are linked to individuals outside of this group. In social networks, this is generally realized through weaker social relations, that we will call here acquaintance by opposition to strong friendship ties.

By not going further in the strong ties, but focusing on the weak ties, Granovetter highlights the importance of acquaintances in social networks. He argues, that the only thing that can connect two social networks with strong ties is a weak tie. His assumption is backed by statistical data showing that people are more likely to find a new job through the help of an acquaintance rather than a close friend[Gra70], because the latter is not able to get in touch with people not already known. Since then, many studies highlighted the importance of weak ties in the spread of trends, informations, diseases[Kra92, Mon92, Mon94].

3.2.2.3 Usage in wireless ad hoc networks

We can now transpose the social network model in the context of wireless ad hoc networks, that are the main focus of this thesis. As we focus on the development of social applications over those networks, and primarily envision its implementation in an urban environment, we can make the following remarks:

- **social relationships** will, to a certain extent, impact our network topology, as friends are likely to be physically close to each other for extended periods of time;
- **the environment** will also impact our network topology, as unrelated people will tend to gather in public places, where they will be connected by wireless links.

In this context, we are in the presence of groups that are tightly interconnected for extended period of time (between friends, inside the same place) as well as shorter-lived weaker links (across buildings for instance).

To establish robust exchanges using the most reliable wireless links, it is then essential to identify the groups where strong links exists and favor communication inside those groups.

3.3 Community detection

As presented in section 3.2, studies on graph theory and analysis of properties of natural, human and social networks have revealed the heterogenous nature of their link density. Those graphs and communication networks like the Internet or wireless ad hoc networks, have structures between order and randomness. They are neither lattices nor purely random graphs. On the contrary, they show global and local inhomogeneities in both vertex degree and edge distribution[For10].

3.3.1 Communities

Particularly, the triadic closure property and the small-world nature of those networks lead to the existence of inhomogeneity in the distribution of degrees across nodes. These discrepancies generate a division of the network into groups within which the connections are dense, but between which they are sparser[GN02].

Those groups of vertices, whose definition is closely related to the network topology, are called *communities*, *clusters* or *modules*.

3.3.1.1 Communities on static graphs

Definition 3.10 (Community - Girvan and Newman[GN02])
A *community* $\mathcal{S}_C$ of the graph $\mathcal{G}$ is a subgraph of $\mathcal{G}$ with $V(\mathcal{S}_C) \subseteq V(\mathcal{G})$ and $E(\mathcal{S}_C) \subseteq V(\mathcal{S}_C) \times V(\mathcal{S}_C)$, defined so that:

$$\Delta(\mathcal{S}_C) \geq \Delta(\mathcal{S}_C \cup \mathcal{S}) \tag{3.5}$$

for any subgraph $\mathcal{S}$ of $\mathcal{G}$.

Community structures have no predefined size, shape or diameter. Finding the community assignment, i.e. placing all network vertices in their respective community, is a NP-hard problem related to graph partitioning[DDDGA05]. More precisely, it is a clustering problem, as the number of classes, or communities, is always unknown to the algorithm.

Despite this complexity, the community assignment of a given network allows to discover sets of deeply interdependent elements, which is helpful while studying intricate phenomena or systems using graph theory.

Considering the underlying concepts the studied graph is abstracting from, communities can also be envisioned as groups of vertices which probably share common properties and play similar roles within the graph[For10].

3.3.1.2 Extension to dynamic graphs

In the context of dynamic graphs, the concept of social structure or *community* shall be extended to integrate this new time-based dimension. The high internal connectivity and the high ratio of closed triads[Gra73] (or common neighbors[HB10]) that characterize the tight interdependence between the vertices are now in balance with the significance of the graphs connections over time.

It remains however an open problem to determine whether more importance shall be given to long-lasting social relations, i.e. stable links of a dynamic graph, or to relations that occur often, leading to links with an important volatility. Far from a definitive answer to this question, we propose to focus on an application-oriented definition for the importance of links. Considering the time-scale of the envisioned interactions (e.g. the time of a shared game or a streamed video), we will give higher importance to links being robust and stable during this time.

Hence, it is legitimate to complement the search for *tight* communities (the one composed of deeply interconnected nodes) with the search for *robust* communities (the ones presenting higher structure stability metrics, such as average structure link stability).

As a consequence, we propose the following extension to the definition of communities for dynamic graphs:

Definition 3.11 (Community - Dynamic Graph)
Let m_S, a stability measure on the elements of the dynamic graph $\mathcal{G}$, and $\overline{m_S}(\mathcal{S})$ the average of the measure over the subgraph $\mathcal{S}$ of $\mathcal{G}$.
A *community* $\mathcal{S}_C$ of the graph $\mathcal{G}$ is a subgraph of $\mathcal{G}$ with $V(\mathcal{S}_C) \subseteq V(\mathcal{G})$ and $E(\mathcal{S}_C) \subseteq V(\mathcal{S}_C) \times V(\mathcal{S}_C)$, defined so that:

$$\Delta(\mathcal{S}_C) \geq \Delta(\mathcal{S}_C \cup \mathcal{S})$$

and

$$\overline{m_S}(\mathcal{S}_C) \geq \overline{m_S}(\mathcal{S}_C \cup \mathcal{S}) \tag{3.6}$$

for any subgraph $\mathcal{S}$ of $\mathcal{G}$.

The stability metric m_S used for community determination can, from instance, be chosen from the dynamic graph metrics presented in 2.2.2.4.

3.3.1.3 Community assignment

From those definitions, we can formalize what *community detection* shall achieve.

Definition 3.12 (Community Assignment)
Community detection aims at finding a partition of the graph $\mathcal{G}$, i.e. a set of communities $\mathcal{C}(\mathcal{G}) = \{\mathcal{S}_{C,1}, ..., \mathcal{S}_{C,n_c}\}$ of the graph $\mathcal{G}$, so that:

- $\forall i, j \in [1..n_c]\ \mathcal{S}_{C,i} \cap \mathcal{S}_{C,j} = \emptyset$
- $\bigcup_{i \in [1..n_c]} \mathcal{S}_{C,i} = \mathcal{G}$
- each $\mathcal{S}_{C,i}$, $i \in [1..n_c]$ verifies definition 3.10 (or definition 3.11 on dynamic graphs)

The partition $\mathcal{C}(\mathcal{G})$ is a *community assignment* on the graph $(\mathcal{G})$.

Note that this definition does not imply that such an assignment $\mathcal{C}(\mathcal{G})$ is unique for any given graph $\mathcal{G}$. However, the quality of all possible assignments might not be the same, as will be explained in 3.3.4.

3.3.2 Comparison with other clustering algorithms

Because any community assignment $\mathcal{C}(\mathcal{G})$ of a graph $(\mathcal{G})$ is a partition of this graph, community detection can be seen as a form of clustering.

Definition 3.13 (Graph clustering)
Graph clustering consists in assigning a set of nodes into groups, called cluster, so that the nodes in the same cluster are more similar, in some agreed sense, to each other than to those in other clusters.

From the previous definition, community detection hence appears as a specialization of graph clustering, where the notion of similarity is tightly linked to local network density: similar nodes being densely linked to each other and hence sharing a lot of closed triads.

Many clustering algorithms can be found in the literature, depending on the the adopted similarity notion. The clusters found by those different algorithms vary significantly in their properties, and understanding these cluster models is key to understanding the differences between the various algorithms.

Clustering algorithms can hence be classified based on the models they use to define clusters:

- **Connectivity or density models**: for example hierarchical clustering building models based on connectivity distance (path length) or mutual connectivity or density, like community detection algorithms ;
- **Centroid models**: for example the k-means algorithm representing each cluster by a single mean vector, not necessarily existing in the cluster, based on the position of the different components of the cluster in an Euclidian space ;
- **Distribution models**: clusters are modeled using statistic distributions, such as multivariate normal distributions used by the Expectation-maximization algorithm.

In the rest of this chapter, we will focus on present existing solutions for connectivity-based clustering, or community detection, algorithms.

3.3.3 Community detection algorithms

As for many complex problems, centralized solutions have been presented first. Then more memory efficient algorithms, based on local information have been introduced before the recent emergence of fully decentralized solutions, the only ones suitable for distributed communication networks. However, all those solutions either rely on assumptions unrealistic in a distributed environment or do not cope with issues related to the dynamic nature of the graphs.

3.3.3.1 Centralized algorithms for static networks

The first algorithms aimed at performing community detection relied on a greedy, centralized approach trying to agglomerate[CNM04] nodes in communities or divide[NG04] the network in such structures. Basically, they iteratively operate until a measure reflecting the assignment is optimized. Most of the time, the *modularity* measure Q (presented in definition 3.14) is used. Nodes to group or split are chosen using similarity, betweenness[NG04] or extremal optimization[DA05]. Alternative approaches are based on spectral properties of the graph, Laplacian matrix[DM04] for instance, or on information theory principles, like in the INFOMAP algorithm presented in [RB08].

A good comparison of those centralized algorithms can be found in [DDDGA05] or in [LF09]. They generally perform very well, achieving high *modularity* values on both unit and weighted graphs. However, their iterative nature, and their requirement for a centralized computation using a global knowledge of the graph makes them hardly applicable to dynamic networks. For the provided solution to be correct, each algorithm shall be run until convergence at each modification of the graph, due to simple or meta-dynamics, making the required complexity soar.

Algorithms relying only on local information have also been proposed, in order to improve the memory efficiency of community detection. In [WCL07], the authors use a cached table representing the network structure and allowing information processing at each vertex to be in $O(1)$ time complexity. In their proposal[WCF08], Wan *et al.* use a variation of the L-shell algorithm introduced by Bagrow and Bollt in [BB05] to iteratively add nodes to a community. They start from the shell origin, until the number of added adjacent vertices goes below a given threshold. In [TCF09], the authors introduce the notions of vertex *intensity* (sum of weights of all its attached edges) and vertex *contribution* to a community C (as the ratio of the sum of weights of all its edges to adjacent vertices in community C over its total *intensity*). They then proceed to a greedy agglomerative algorithm, starting with the edge of highest *contribution*. All those algorithms, however, require at some point node coordination or global network knowledge, which is not suitable for use in ad hoc networks.

3.3.3.2 Epidemic label propagation algorithms

Decentralized algorithms appear amongst most recent contributions to the community detection problem. We distinguish here between *epidemic label propagation algorithms*, where each vertex is responsible for its assignment and *individual or agent-based algorithms*, where distributed agents (possibly distinct from the network vertices) are in charge of mining the social structures.

In [RAK07], Raghavan *et al.* introduced the *epidemic label propagation* principle. In this model, the current community of a vertex v is identified by a *label* (integer, string, etc.) $C(v)$ chosen in the set of communities $\mathcal{C}$. Vertices either synchronously or asynchronously beacon their *label* to their adjacent nodes. Then, each vertex adopts the same label as the majority of its neighbors. This process is illustrated in Figure 3.6, where the propagation of the blue label to vertex d is illustrated.

While directly inheriting from the vertex community definition given in [GN02], this algorithm is proven to generate label oscillations (subsets of vertices alternating between several different community labels) when run synchronously[RAK07] and to form *monster* communities that plague an extended amount of nodes[LHLC09].

In order to address the latter effect, Leung *et al.* refined this algorithm in [LHLC09]. Their version includes a hop attenuation factor δ that fades the infection power of a community and a node preference function (the authors suggest the node degree) weighted by a parameter m. Presented experiments show that the parameter combination $(\delta = 0.1, m = 0.05)$ yields the best results, according to the authors test cases.

The existence of those fixed parameters may not guarantee that this algorithm will perform efficiently whatever the underlying network topology, especially when it is changing, like in dynamic networks with heterogenous mobility. Besides, whether the hop attenuation may limit the creation of *monster* communities, it also limits the natural span of a label around an arbitrary center. Achieved community assignment might therefore be far from optimal.

3.3.3.3 Agent-based algorithms

In a more recent article, Bo *et al.* present a distributed autonomy-oriented algorithm for the community assignment problem[YLL10]. In their approach, a set of autonomous agents is spread over the graph to mine. In their paper, one agent is placed inside each vertex of the graph. Each

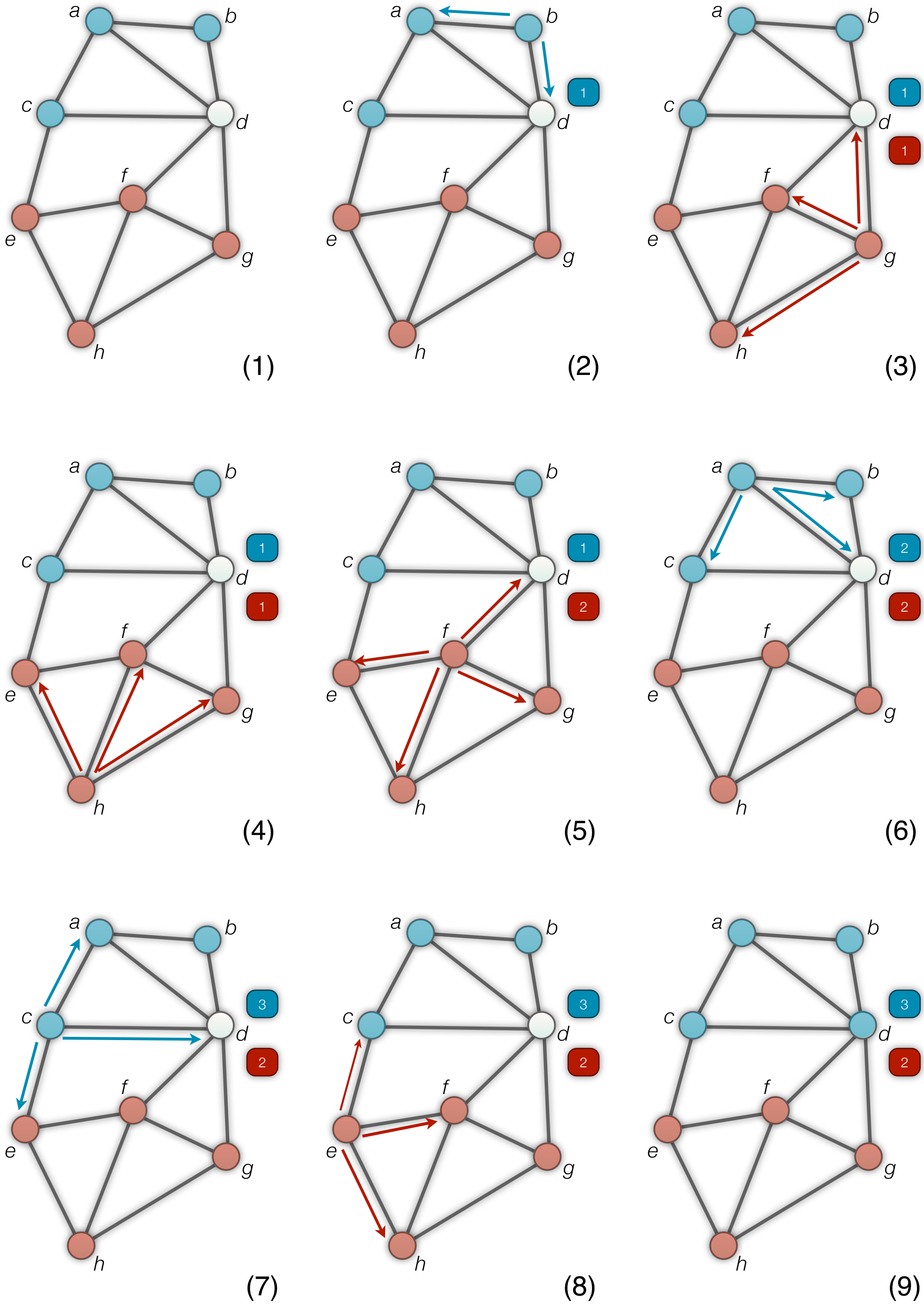


Figure 3.6: Illustration of the epidemic label propagation principle.

agent iteratively builds a *view*, initialized to its set of adjacent vertices, which will converge to the set of vertices of its community. Each agent repeatedly *evaluates* its view (computing a *similarity* measure resembling the one presented in [HB10]), *shrinks* its view (by deleting from the view vertices whose similarity is below a certain threshold), *enlarges* its view (by adding two-hop adjacent vertices to its view) and *balances* (i.e. normalize the similarities) its view. The algorithm converges to the generation of an overlay network composed of all the agent views, where each community is represented by a non-overlapping clique.

Although its operation is fully distributed and might perform correctly over a dynamic graph, no detail nor experiments are provided in the corresponding publication. Furthermore, the presented configuration of the algorithm (one agent residing on each of the graph vertex) is very similar to the approach used by epidemic label propagation algorithms. Besides, as communication with non-adjacent vertices is required when the view is *enlarged*, the algorithm is more subject to the limited path reliability of dynamic networks.

3.3.3.4 Centralized algorithms for dynamic networks

In [BWS06] and [TBWK07], Berger-Wolf *et al.* presented a framework for community identification in dynamic social networks. This work, adopting an optimization-based approach for the problem, is the first major contribution focusing on the impact of dynamic network evolution on social mining.

Assuming that time is discrete, they build an undirected graph $\mathcal{G}$ representing the evolution of the assignment of individuals in *groups* (non-empty and pairwise distinct sets of individuals) over time. A *group* $G_{j,t}$ is defined so that every individual of $G_{j,t}$ is not interacting with any individual of group $G_{j',t}$ at time t, for all $j' \neq j$.

Then, they build a graph where there is a vertex $v_{i,t}$ for every individual i at time t, and a vertex $g_{j,t}$ for each group $G_{j,t}$ existing at time t. Edges are drawn between all $(v_{i,t}, v_{i,t+1})$ pairs, linking two successive representations of the individual, and between each individual and its group at time t, connecting $v_{i,t}$ to $g_{j,t}$ if and only if $v_{i,t} \in G_{j,t}$.

Finally, the optimization problem is to find a valid graph coloring (i.e. where no two group-vertices g and g' share the same color at any time t) minimizing total cost. Costs are introduced to penalize individual or group actions deviating from a virtuous behavior, like an individual changing its color (i.e. community affiliation), or individuals sharing the color of an incorrect group.

The authors reckon this problem as NP-complete and APX-hard[TBWK07] and propose a set of heuristics, such as individual coloring, group coloring or greedy heuristics, to solve it. However, the building of the interpretation graph $\mathcal{G}$ requires a global knowledge of not only the topology but also the complete history of the graph until the last instant of the temporal base $\mathbb{T}$ where the computation happens.

While this approach appears as particularly efficient for *a posteriori* social analysis of dynamic graphs, it is not suitable for distributed communication graphs, where decisions shall be taken *on line* and with a locally limited knowledge in the space and time dimensions.

3.3.3.5 Distributed algorithms for dynamic networks

In [BDGO06], Bertelle, Dutot *et al.* used an individual-based approach to determine communities (referred as *organizations* in the publication). The algorithm simulates several colonies of ants, each having a distinct color, whose members iteratively travel along the edges of the graph. At each pass, each ant marks the traversed edge with a given amount of pheromone of its color. Ants avoid *hostile* edges that are marked with large amount of pheromones from a different colony and can flee to different zones of the graph, if their surrounding environment is judged too hostile. The different groups of ants tend to colonize different areas of the graph that are bounded by drops in the link density. Each ant colony hence dominates a different community, or organization, in the graph.

This approach is really interesting as it inherently manages the dynamics of the underlying graph: changes in the topology will change the hostility level of the environment and allow or prevent ants to explore and maybe colonize new areas of the graph. However, in this algorithm, the number of detected communities is equal to the number of ant colonies introduced in the computation. An estimation or an *a priori* knowledge of the number of communities in the graph is therefore required for results to be significant. This estimation may be hard to obtain or even irrelevant as the graph dynamics may make the expected number of communities vary over time.

3.3.4 Assessing detection quality

As the definition of a *community* can vary depending on the usage context, it is important to define detection quality metrics consistently. Here we will present measures that are only related to the topology of the network on which the assignment is performed. They will be used for performance assessment in section 8.3 and subsection 9.1.3.

3.3.4.1 Modularity

The *modularity* measure Q, introduced by Newman and Girvan in [NG04] is one of the first metric used to evaluate the quality of a community assignment. It measures how the performed assignment matches the community structure exhibited by the network topology.

For a partitioning of the graph $\mathcal{G}$ counting $n_c = |\mathcal{C}|$ communities, we define a squared $n_c \times n_c$ matrix $\mathbf{E}$ whose elements e_{ij} represent the fraction of edges from a vertex in community i to a vertex in community j.

As a consequence, the sum of rows, or columns, of $\mathbf{E}$, $f_i = \sum_j e_{ij}$ corresponds to the fraction of links connected to i. If the community assignment were random, the expected number of intra-community links would be f_i^2. Hence the definition of Q given in definition 3.14:

Definition 3.14 (Modularity - Newman and Girvan (NG04))
With the previously presented notation, the *modularity measure* Q is defined as:

$$Q = \sum_{i \in \mathcal{C}} \left(e_{ii} - f_i^2 \right) \tag{3.7}$$

The modularity can also be expressed as a function of the vertices connectivity (elements a_{uv} of the network adjacency matrix $\mathbf{A}$), the node degree distribution and the current community assignment ($C(v)$ being the current community of node v) as shown in definition 3.15, where the Kronecker function $\delta(x, y)$ is 1 if and only if $x = y$.

Definition 3.15 (Modularity - Alternative definition)
With the previously presented notation, the *modularity* measure Q is also defined as:

$$Q = \frac{1}{2M} \sum_{u,v \in V(\mathcal{G})} \left(a_{uv} - \frac{d(u)d(v)}{2M} \right) \delta(C(u), C(v)) \tag{3.8}$$

As it is closely related to the underlying community structure of a network, the maximum achievable value for Q is not absolute but topology dependent. The advantage of this metric is that it does not require a reference assignment to compare to but, unfortunately, it does not quantify the distance to an optimal assignment and does not measure the robustness of the assignment to small network changes[KLN08].

3.3.4.2 Metrics for weighted graphs

If we suppose that graph elements are weighted, with edge weights reflecting the stability or quality of the link, we need to redefine the modularity, in order to consider those values. We therefore introduce the *weighted modularity* measure Q_w as the weighted extension of the modularity.

Definition 3.16 (Weighted modularity)
With the previously presented notation, the *weighted modularity* measure Q_w is defined as:

$$Q_w = \frac{1}{2W} \sum_{u,v \in V(\mathcal{G})} \left(a_{w,uv} - \frac{d_w(u)d_w(v)}{2W} \right) \delta(C(u), C(v)) \tag{3.9}$$

where W is the total weight of all network edges, $a_{w,uv}$ the elements of the weighted adjacency matrix $\mathbf{A}_w$.

3.3.4.3 Comparative metrics

Whenever a natural or pre-defined assignment exists for a network, many other metrics have been developed to estimate a normalized distance between the given and the computed classification.

In [DDDGA05], Danon *et al.* introduced the *normalized mutual information* (NMI) measure based on information theory. This normalized metric evolves between 0 when computed ($\mathcal{C}$) and pre-determined ($\mathcal{C}'$) assignments are independent, and 1 when they are identical.

Definition 3.17 (Normalized Mutual Information - Danon ([DDDGA05]))
The *normalized mutual information*, or NMI, also noted $I_n(\mathcal{C},\mathcal{C}')$, is defined as such:

$$I_n(\mathcal{C},\mathcal{C}') = \frac{2I(\mathcal{C},\mathcal{C}')}{H(\mathcal{C})+H(\mathcal{C}')} = \frac{2(H(\mathcal{C})-H(\mathcal{C}|\mathcal{C}'))}{H(\mathcal{C})+H(\mathcal{C}')} \tag{3.10}$$

where $H(\mathcal{C})$ is the entropy of the assignment $\mathcal{C}$, $H(\mathcal{C}|\mathcal{C}')$ the conditional entropy of the assignments $\mathcal{C}$ and $\mathcal{C}'$ and $I(\mathcal{C},\mathcal{C}')$ the variation of information (i.e. not normalized) between the two assignments.

In search for a robustness measurement, Karrer *et al.* proposed the *variation of information*[KLN08], based on the difference of the conditional entropies $H(C|C')$ and $H(C'|C)$ of two communities. Those metrics shall be preferred for benchmarks where networks are generated based on pre-determined community assignment, like the tests described in [LF09].

3.4 Highlight

In this chapter:

- We have envisioned several **graph topology models**: random graphs, regular graphs, free-scale graphs;
- We have described the principal characteristics of **social networks**, that represent social interactions between individuals;
- Particularly, we have seen that those networks exhibit **small world** properties;
- Then, we have seen that these properties are due to presence of both **strongly connected groups**, as well as **weaker ties**, and extended this model to urban wireless ad hoc networks;
- We have formalized the definition of those groups as **communities** and investigated algorithms to find them in networks as well as metrics that can evaluate their performance.

We have seen that the existing algorithms to mine communities in network can not be directly applied to wireless ad hoc networks. Such an algorithm will therefore be the key part of our contribution.

II

Wireless ad hoc networks

Contents

Chapter
4

Wireless ad hoc networks

> "It's a platform for pursuing the truth,
> and the decentralized creation and distribution of ideas [...].
> It's a platform, in other words, for reason."
> Albert Arnold Gore Jr., The Assault on Reason.

In this chapter we will present ad hoc networks by putting them in perspective with other wireless communication systems, illustrating their operation by a metaphor. We will insist on the decentralized and dynamic nature of those systems.

Like any other wireless communication system, ad hoc networks are restricted in their capabilities by radio technology limitations on data transmission speeds and range. In order to get a fair idea of these restrictions, we will summarize in this chapter basic characteristic features of some radio technologies commonly used at the physical layer in ad hoc networks.

Next, we will focus on the distributed nature of those networks. After properly defining the notion of a distributed system and illustrating it, we will review the implications on the design and operation of wireless ad hoc networks.

Finally, due to the sporadic nature of those links and because mobility is a challenge in ad hoc networks, we will introduce the notion of dynamic system and study its application as an abstraction of wireless ad hoc networks.

4.1 Introduction to wireless ad hoc networks

In this section, we will present a general definition and key characteristics of wireless ad hoc networks. Due to the possible, and very likely, mobility of the terminals composing those networks, we will indifferently also refer to them as MANET, acronym of Mobile Ad Hoc Network.

4.1.1 A MANET definition

Ad hoc networks are formed in situations where mobile computing devices require networking applications while a fixed network infrastructure is not available or not preferred to be used. In these cases mobile devices could set up a possibly short-lived network for the communication needs of the moment, in other words, an ad hoc network.

Definition 4.1 (Ad hoc network)
An *ad hoc network* is a self-organizing network, capable of forming a communication network without relying on any fixed infrastructure.

A high-level description of ad hoc networks and related research topics can be found in [Per01] and [MM04].

Wireless ad hoc networks are conceptually compared to traditional centralized wireless approaches, such as cellular networks and usual wi-fi deployments. Wireless multi-hop ad hoc networks are formed by a group of mobile users or mobile devices spread over a certain geographical area. We call the users or devices forming the network nodes. The service area of the ad hoc network is the whole geographical area where nodes are distributed.

Each node is equipped with a radio transmitter and receiver which allow it to communicate with the other nodes. As mobile ad hoc networks are self-organized networks, communication in ad hoc networks does not require a central base station. Each node of an ad hoc network can generate data for any other node in the network. All nodes can function, if needed, as relay stations for data packets to be routed to their final destination. A mobile ad hoc network may be connected through dedicated gateways, or nodes functioning as gateways, to other fixed networks or the Internet. In this case, the mobile ad hoc network expands the access to fixed network services.

Although single-hop ad hoc networks are often used in practice, when we refer to ad hoc networks in this thesis we always mean multi-hop ad hoc networks. The multi-hop support in ad hoc networks, which makes communication between nodes out of direct radio range of each other possible, is probably the most distinct difference between mobile ad hoc networks and other wireless communication systems.

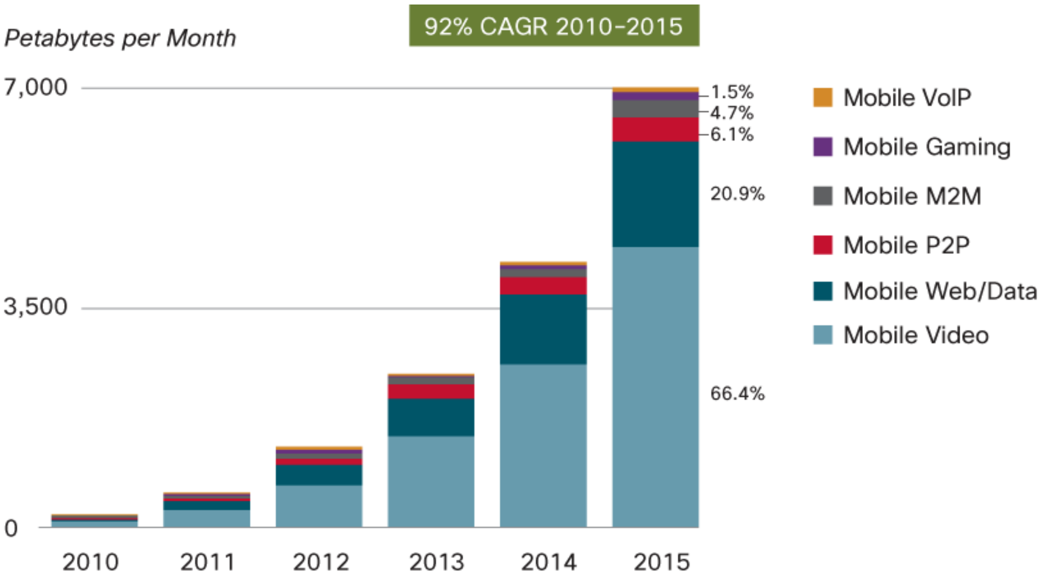
4.1.2 Ad hoc network advantages

Mobile ad hoc networks have certain advantages over the traditional communication networks. Some of these advantages are:

- their inner **flexibility** as communication paths can be created or destroyed in a very short time, with minimal service disruption;

- their lowering of **link budget**, as they eliminate fixed infrastructure costs and reduce power consumption at mobile nodes;
- their higher **robustness** with regards to conventional wireless networks due to their non-hierarchical distributed control and management mechanisms;
- their improved **power management**, as devices can reduce their transmission power and use multi-hop relaying instead of sending a signal over an extended distance in one long hop;
- their more efficient **spectrum management** as short communication links keep radio emission levels low; hence reducing interference levels and increasing spectrum reuse efficiency.

According to network infrastructure providers, mobile data traffic will increase around forty times in the next five years[Cis10], saturating cellular infrastructure. To relieve part of the traffic volume, this calls for the development of peer-to-peer communications between handheld communicating devices in ad hoc mode. When applied to relieve the already saturated bandwidth of 3G cellular networks, this operation is called *3G offloading*. With the soar of mobile applications such as video streaming, relying on decentralized approaches that will diminish the bandwidth usage by caching and mutualizing access to data source servers will become irremediable. Figure 4.1, that presents the latest traffic volume and repartition from network provider Cisco, speaks for this point.



VoIP traffic forecasted to be 0.4% of all mobile data traffic in 2015.
Source: Cisco VNI Mobile, 2011

Figure 4.1: Evolution of mobile traffic volume and repartition according to Cisco

Examples of potential applications of mobile ad hoc networks range from simple ubiquitous data transfer scenarios to very specific uses cases, like disaster relief or battlefield deployments.

We may think of a group of people with laptop computers at a conference that wish to exchange files and data without mediation of an additional infrastructure. We also can think of deploying ad hoc networks in homes for communication between smart household appliances. In such examples, spontaneously creating ad hoc wireless network can fulfill the purpose of reliable transmission of data even in presence of heterogenous devices.

Ad hoc networks are suitable to be used in areas where earthquakes or other natural disasters have destroyed communication infrastructures. They can be useful for search and rescue, crowd control and logistic operations. In situations where the conventional infrastructure-based communications facilities are destroyed, because of natural phenomena -earthquake, flood- or consequence of armed conflicts -massive bombing, jamming-, the deployment of wireless ad hoc networks appears as an efficient solution for organizing rescue activities. In this context, fault-tolerant communication paths and real-time communication capabilities are important since voice communication might predominate, and all exchanged information can be considered of vital importance.

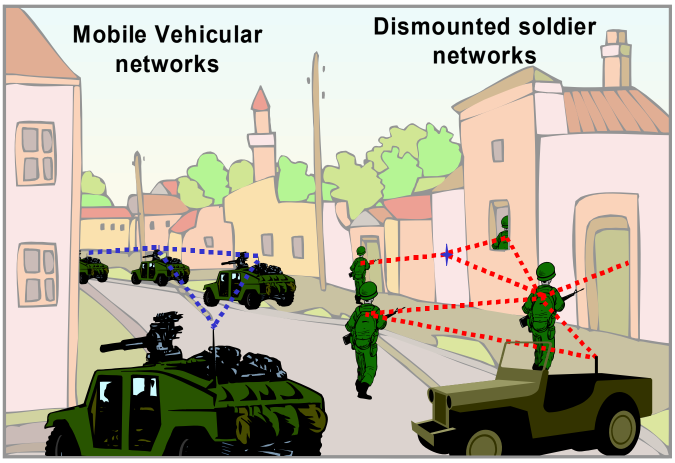


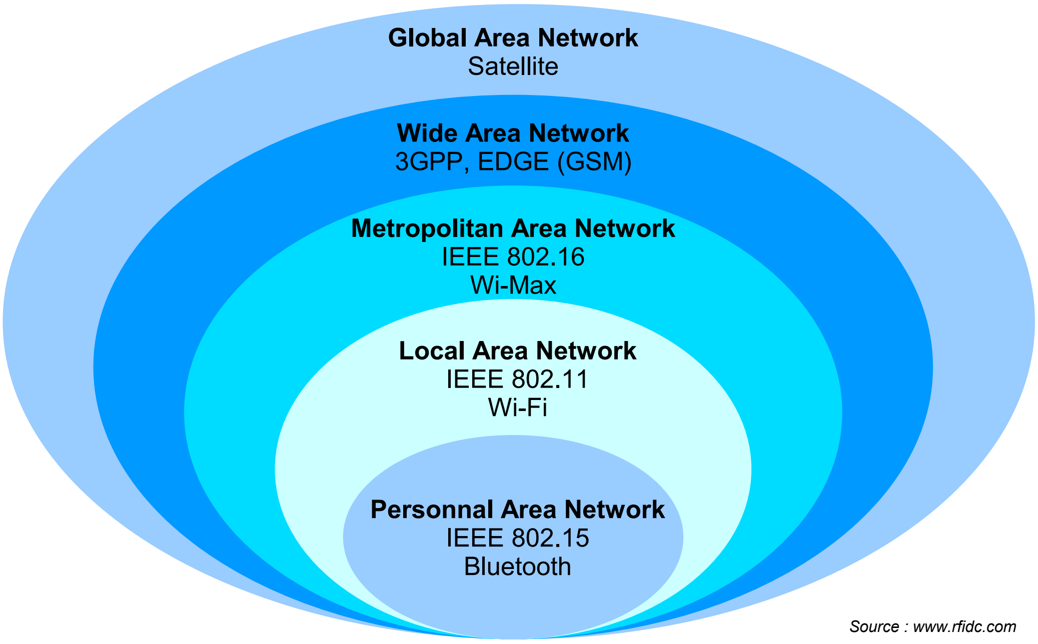
Figure 4.2: Illustration of a MANET use-case scenario for the military [Bau04]

Ad hoc networks perfectly satisfy military needs like battlefield survivability, operation without pre-placed infrastructure and connectivity beyond the line of sight. They can be very useful in establishing communication between battalions involved in tactical operations, or to coordinate military objects moving rapidly, such as airplanes or drones. Besides this requirement for quick reorganization and adaptivity, the military context makes secure communications of prime importance as eavesdropping or other security threats can seriously compromise the success of the operation or the security of the involved personnel.

A specific kind of ad hoc network is the sensor network, where a large number of sensor nodes, very energy-limited communicating devices, are deployed to observe and report a phenomenon. Sensor networks have received much attention in recent years because they have huge potential applications. We will describe more this type of ad hoc network, amongst others in 5.2.3.

4.1.3 Wireless technologies in MANET

In wireless ad hoc networks, radio waves are used to enforce communication between the devices, or nodes composing the network. The particular radio technology used can be chosen amongst a large variety of standards, depending on the desired properties : long transmission range, large bandwidth, available frequencies, etc. Detailing all those technologies is out of the scope of this thesis. However, in order to get an impression regarding possibilities and restrictions imposed by radio communications we provide an overview of basic characteristics of some radio technologies suitable for ad hoc networks.



Source : www.rfidc.com

Figure 4.3: Wireless standards and their respective coverage area

Depending on the service area size, a radio technology developed for Wireless Personal Area Networks (WPAN), Wireless Local Area Networks (WLAN), or Wireless Metropolitan Area Networks (WMAN) may be adopted for ad hoc networks. The coverage radius of a WPAN is roughly in the order of a few meters up to 20 meters. WLAN coverage radius is limited to about 100 meters, while WMAN coverage is in the order of a few kilometers. For each network type various wireless technologies have been proposed. Some examples are:

- Wireless Personal Area Network: Bluetooth, UWB
- Wireless Local Area Network: IEEE 802.11a, IEEE 802.11b, IEEE 802.11g
- Wireless Metropolitan Area Network: IEEE 802.16e

Those standards, along with an illustration of their respective coverage range are given in Figure 4.3

For example, dense low-bit-rate sensor networks may be built based on a WPAN technology, while for communication between moving cars at distances in the order of tens of meters a WLAN technology like IEEE 802.11b may be more suitable. It is also worth mentioning that some frequency bands are license-exempt frequency bands. This makes deployment of ad hoc networks in these frequency bands commercially attractive.

Finally, some radio technologies have not been designed specifically to support mobility or only allow very moderate forms of mobility. However, wireless ad hoc networks can consist in fast moving nodes and some technologies might not be suitable, mainly for limitations of their link layer protocols.

4.2 A distributed system

We have already mentioned the decentralized, or distributed, nature of wireless ad hoc networks, especially when we opposed them to cellular networks or wireless access using a wi-fi access point. In this section, we will introduce the notion of a distributed system and show in which sense wireless ad hoc networks fulfill this definition.

4.2.1 A simple definition from literature

Many definitions for a distributed system are present in the literature, some of them are very elaborated. However, we have chosen to retain the one proposed by Tanenbaum and Van Steen in [TvS02], for its completeness through simplicity:

Definition 4.2 (Distributed system - Tanenbaum and Van Steen (TvS02))
A distributed system is a collection of independent computers that appears to its users as a single coherent system.

This definition was originally designed for computer-based distributed systems and covers both the hardware part of the system (the independent computers) as well as the software part (the single coherent system).

In the light of distributed communication networks, we can simply re-interpret, or rewrite, this definition by considering the independent entities as the communicating devices composing the network, and the coherent system as the network *as a whole*, or the services it is able to offer thanks to nodes collaboration.

This last notion of collaboration is probably the most important of a distributed system that is not captured by the definition of Tanenbaum and Van Steen. Because the entities are considered independent, all interactions between those devices require a cooperative behavior. In later sections (like section 5.1) we will see how this cooperation allows to address some of the challenges of wireless ad hoc networks.

We therefore propose the following definition of a distributed communication system:

Definition 4.3 (Distributed system - application to communication networks)
A distributed communication network is a collection of independent collaborating communicating devices that appears to its users as a single coherent communication system.

This definition, when compared with the one we proposed for ad hoc networks (see definition 4.1), reveals that MANETs can rightfully be considered as an excellent example of a distributed system.

4.2.2 Parallel, Centralized, Decentralized or Distributed?

Once the notion of interaction, or cooperation, of the entities of a distributed system is set, it is legitimate to rise the question of how those interactions are organized and how the components of the system actually achieve cooperation.

This section will present the differences between some denomination that are often thought as equivalent, hence clarifying the remainder of this manuscript.

4.2.2.1 Parallel vs. distributed

Many systems are indifferently denominated as *parallel* and *distributed* as the components of a typical distributed system operate concurrently in parallel. In fact, a parallel system may be seen as a particular tightly-coupled form of a distributed system, and reciprocally a distributed system may be seen as a loosely-coupled form of a parallel system.

However, one criterion allows to easily distinguish between a parallel and a distributed system, based on the nature of the information available to the entities and how it is made available. In a parallel system, all components have access to a shared memory. Shared memory can be used to easily exchange information between all entities that can therefore easily achieve a global knowledge of the system. In a distributed system, each entity has its own private memory (distributed memory) and information is exchanged by passing messages. For scalability and efficiency reason, only a limited set of the information is exchanged and generally only *local* information, i.e. data available in the surroundings of each entity.

Parallel systems are generally found in multiprocessor computers, or in computational clusters or grids, where the calculus is spread across the different components but information (source data, intermediary results) are shared directly. Distributed systems can be found in distributed databases, peer-to-peer networks or communication networks.

4.2.2.2 Centralized vs. decentralized vs. distributed

In a memorandum of the RAND corporation[Bar64] (a U.S. military agency whose aim was to develop tactical communications for defense operation), Paul Baran, considered as the father of packet based and switched networks, investigates the various architectures of communication systems. With tactical applications in mind, he was in search for network resilience and robustness to attacks and system failures.

While the difference between parallel and distributed systems resided in the support for cooperation and information sharing, we can distinguish centralized, decentralized and distributed systems by studying the organization of nodes cooperation. The different approaches can be easily pictured using Figure 4.4, an illustration Baran proposed in his memorandum and that remained famous for its clarity.

Centralized networks exhibit a star topology around a *center* node, which, in the network design, is often used to coordinate the action and the exchange of the other nodes. It generally gathers and processes information sent by the other nodes, forward messages for which it was not intended as destination, and can also generate information (or reports) based on information it has received. However, this center node appears as a point of failure in the network. If the failure of one of the peripheral node would only cause marginal service disruption, affecting the center node would cease all operations as the coordinator and single relay interconnecting all other entities vanished.

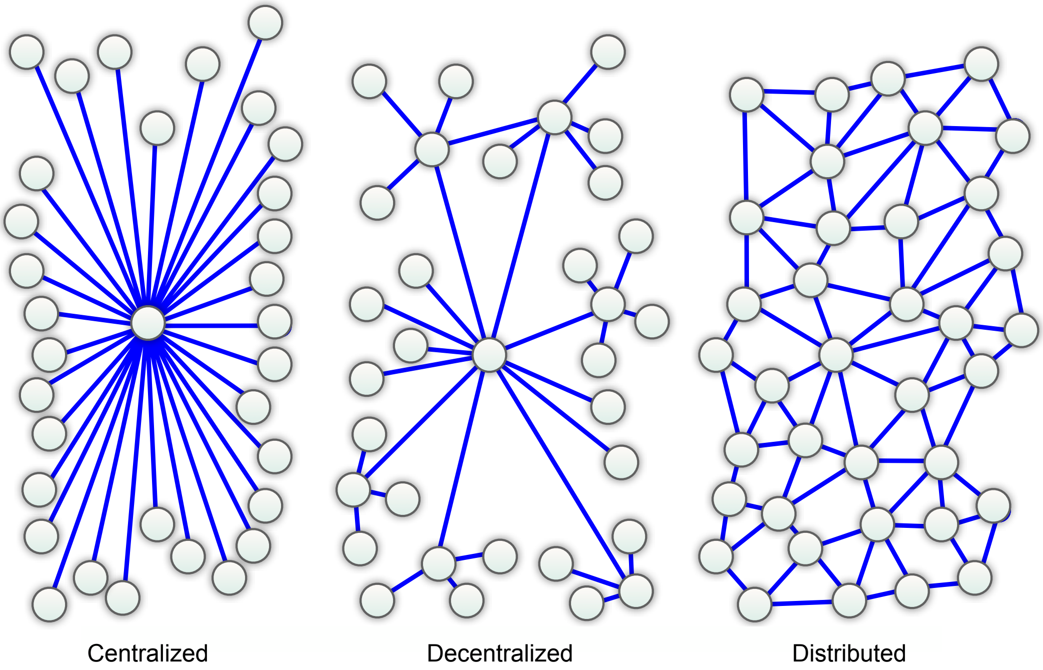


Figure 4.4: Centralized vs. decentralized vs. distributed

Decentralized networks appear in many existing communication networks, like today's Internet or cellular communication systems. They rely on an architecture that mixes smaller star topologies around local *center* nodes and mesh, or interconnected, components. Due to the existence of several coordinator nodes, those networks are often referred to as *decentralized*. Even though it offers more resilience to failure as purely centralized networks, the local *centers* generally act as communication hubs (in the sense of hubs in small-world networks, see sub-subsection 3.2.1.4) and their disruption may dramatically impact not only the performances of a limited area of the network but may also globally disrupt its normal functioning. This is mainly due to the inner hierarchical structure of those networks

Distributed networks appear as the most robust network architecture: no central or local coordination point exist so that the destruction of one or a small number of nodes will only impact locally the availability and performances of communications. As per pure application of definition 4.3, the network is composed of independent and *a priori* identical entities that have to collaborate for the network to operate. It is not forbidden that some form of hierarchy emerges in the network: some node might operate differently than other, but this different behavior is the result of a distributed election process occurring across the nodes.

Finally, from Figure 4.4, we can deduce one of the main advantages of distributed networks: their resilience to failure. This property, also called *robustness* was already noticed by Baran in his report. It it not only due to the strict decentralized nature of all decisions (hence, there is no more coordination point whose failure would impact the complete network), but also to the multiple possible paths between any pair of vertices composing the network (the absence of coordination point theoretically allows to use all the existing links for communication).

4.2.2.3 Robustness assessment

We can further assess the robustness of the decentralized system, by comparison with the centralized and decentralized approach, by using the *three questions of robustness* that were popularized by Siegel[AMSK03]:

Definition 4.4 (Robustness - Siegel (AMSK03))
The *robustness* of a system can be evaluated by answering all of the three following questions:

1. What behavior of the system makes it robust?
2. What uncertainties is the system robust against?
3. Quantitatively, exactly how robust is the system?

Here, we clearly envision the robustness of the communication system against the failure of either a certain number of nodes or vertices (question 2). A robust behavior will be the ability to maintain communication capabilities across the network, despite the failure of a measurable fraction of the graph elements (question 1). The degree of robustness can be quantitatively measured by monitoring, for each fraction of failing elements, how many pairs of nodes (sender, receiver) can still directly or indirectly communicate (question 3).

As we have already seen in section 3.1, this resilience to failure of some elements, is highly dependent on the topology of the graph and on the distribution of failures over the elements. For instance, if failing nodes are chosen at random, there are chances that the coordination point of the centralized network will not be affected for long, hence maintaining a high operation ration despite many failures. However, in many systems, including communication networks, local or global coordination points are more heavily used, appear as bottlenecks, and are more prone to failure. A more realistic measurement method would hence target those entities for failure first.

Besides, when link failure are considered, distributed system offer many more redundant, or alternative, paths from one node to another so that, whatever the failure distribution model used, they appear as more robust than other approaches.

4.3 A dynamic system

We have already emphasized on the dynamic nature of wireless ad hoc networks. Composed by moving communicating entities that use a communication channel whose performance is closely related to the relative distance between sender and receiver, those networks are prone to failure of both communication links and nodes. This lead us to chose dynamic graphs as mathematical formalism for their study (see subsection 2.2.2).

While previous paragraph underlined the strength, or *robustness*, of distributed systems, we will see how the dynamic nature of wireless networks somewhat mitigates this advantage.

4.3.1 Information propagation

Due to their distributed and dynamic nature, algorithms designed for wireless ad hoc networks require devices to constantly exchange up-to-date information. This is particularly the case of routing protocols (presented in subsection 5.1.2) that use a series of *broadcast* messages to find a valid path in the reactive approaches (see subsubsection 5.1.2.2) or to maintain valid routing tables in proactive approaches (see subsubsection 5.1.2.3).

4.3.1.1 Broadcast storm

Although it is required for the nodes to quickly exchange information and hence have an up-to-date representation of the network, the broadcast messages have the following drawbacks :

- **broadcast is spontaneous**: mobile nodes can start broadcasting at any time, without synchronization and broadcast messages generally bypass channel access mechanisms (see subsection 5.1.1) as the information they convey is generally used to make the latter operate;
- **broadcast is unreliable**: as it bypasses channel access mechanisms and is intended to all nodes in the transmission range; broadcast messages are generally redundantly sent to ensure each envisioned destination receives at least one copy of it.

Due to those constraints, the broadcast traffic volume can go incredibly high, in a phenomenon called *broadcast storm*.

Definition 4.5 (Broadcast storm)

A *broadcast storm* occurs when a large fraction of the nodes composing the network are willing to transmit at the same time.

Generally, this expression is used when the information to be transmitted is aimed for the complete network (*broadcast* information packets), although it can also occur when many one-to-one communications (*unicast* information packets) are generated concurrently, as the wireless transmission medium in inherently broadcast.

During a storm, many nodes will contend for access to the transmission medium, leading to an increased delay in packet delivery. Besides, none of the existing medium access control mechanism for wireless ad hoc networks is able to guarantee a collision-free access (see subsection 5.1.1).

4.3.1.2 Effects on distributed algorithms

The broadcast storm could lead to extended delays in data delivery or to packet loss in case of collisions. Under those circumstances, the information available at the node for the routing operation is very likely to be outdated and partial. Due to the network dynamics this is likely to lead to:

- **unreachable nodes**: if, for instance, too many signaling messages emitting from the considered device are lost or arrive to late to be considered then this node will not appear in any valid route;

- **routing loops**: due to the partial and outdated nature of the information, erroneous routes may make packets bounce between hosts or be indefinitely routed across the network (or until they expire).

Broadcast storms indeed affect all distributed algorithms that rely on one or two-hop neighborhood knowledge, as this information is generally acquired through periodic local broadcast packets. This is the case for most of the algorithms we present in this document, including the SHARC algorithm (see [HB10]) and its associated set of algorithms.

Under extreme conditions (either a very dense network, in which situation many nodes of the network will be contending for transmission of those packets; or when the network dynamics are really high, requiring a rapid update of the information to keep it relevant; or a combination of both cases), the broadcast storm problem is even likely to prevent the sole convergence of the algorithm. This is hence an important problem that needs to be addressed properly.

4.3.2 Link and path stability

Wireless ad hoc networks cumulate the disadvantages of using a naturally unreliable channel for the transmission of information and of allowing communicating entities to move. As the ability for those entities to communicate is tightly correlated to their relative distance, the result is that communication links are likely to experience instability.

As ad hoc networks rely on multi-hop relay of the transmitted information, it is wise to examine the achievable stability of those communication paths, especially if they have to be long, necessitating the use of many wireless links.

4.3.2.1 Path existence

The probability of existence $p(u, v)$ of an edge (u, v) is highly dependent on the location function (or mobility model) L and the transmission function T. Many formal expressions have been presented in the literature for some special cases, for instance [MZ99] and [YLG03] consider random movements and euclidean transmission functions.

Naturally, the probability of existence of a path can be expressed as depending on the existence of all the links composing the path.

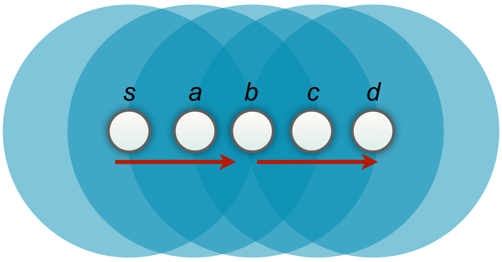
Definition 4.6 (Path Probability)

Given $p(u, v)$, the probability that the edge between vertices u and v of a graph $\mathcal{G}$ is established, the *probability that the path* $\mathcal{P} = \{v_0, ..., v_{n-1}\}$ *is established* is:

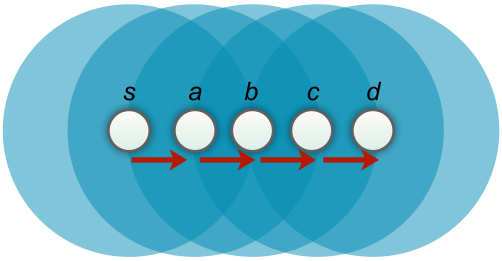
$$p(\mathcal{P}) = \prod_{i=0}^{n-2} p(v_i, v_{i+1}) \tag{4.1}$$

This naive expression would favor the choice of shortest path to relay information, as this would maintain the existence probability of the path to an acceptable level.

However, this expression does not take into account the fact that in wireless ad hoc networks link existence are not completely independent, mostly because links are space-correlated. Particularly, Lim *et al.*[Lim02] exhibited what they call the *edge effect*. The shortest path (in terms of intermediate hops) generally involves nodes which are at the edge of their respective transmission range. As illustrated on Figure 4.5, those paths are not stable as intermediate links are very likely to fail due to relay node mobility (e.g. node b moving just a bit closer to node c in Figure 4.5(a)) or sporadic interferences.



a) shortest route from node s to d



b) route through most stable links

Figure 4.5: Illustration of the edge effect: the shortest path is not the most reliable path.

Note that it might also not be wise to rely on extremely long path for communication, as this would induce an increased latency in information delivery from one path end to another.

In order to achieve reliable transmissions, it is therefore important to focus not only on path lengths, but also on **path stability**, in the search of a balance of the two.

4.3.2.2 Path stability

Once a link stability, or link failure, model is established, it is natural to derive a path stability model from it. One simple assumption is that a path stability metric is cumulative[GdWFM02], meaning that the cost of an $n - hop$ path can be calculated from the cost of an $(n - 1)hop$ subpath and the cost of the n^{th} hop.

Definition 4.7 (Path Stability)

Given $s(u, v)$, a stability metric on the edge between vertices u and v of a graph $\mathcal{G}$, the *stability of the path* $\mathcal{P} = \{v_0, ..., v_{n-1}\}$ is

$$s(\mathcal{P}) = \min_{i=0..n-2} s(v_i, v_{i+1}) \tag{4.2}$$

So, based on this observation, the search for a reliable path for communication requires to both favor stable links and avoid unstable ones, but also to limit the number of links in the path, so the global stability is less likely to be impaired by one noticeably unreliable link and latency is kept at an acceptable value.

4.4 Highlight

This chapter allowed us to:

- Introduce the notion of **mobile ad hoc network** (MANET), their advantages and potential fields of applications;
- Focus on the **distributed nature** of MANETs and highlight their **robustness** against random failures of communicating entities or links;
- Be aware of their **dynamic nature** and the associated constraints:
  - the need to **disseminate information** so that distributed decision can be taken with up-to-date information about the network state;
  - the limited **stability** of communication links, due to the mobility of nodes and the nature of the communication channel.

Particularly, we have concluded that **reliable communications** can only be achieved on **size-limited paths**, including preferentially the **most robust links**.

This matches the social **community** structures interlinked by **weaker ties** we have exhibited in chapter 3.

Contents

Chapter 5

Challenges and applications

> "The only thing doomed to fail is the one you don't attempt."
> Paul-Emile Victor, Dialogues à une voix (personal translation).

We will now focus on more practical challenges of wireless ad hoc networks. Due to both their distributed and dynamic nature, we will see how simple operations like sharing the information medium between entities, or routing information across the network requires elaborated algorithms that, unfortunately, are not able to guarantee error-free operation. Those limitations are even further effective when the network is either large, densely connected or highly dynamic, diminishing the stability of communications links.

Despite those challenges, we will illustrate actual systems where wireless ad hoc networks can play a very interesting and important role. This list is far from exhaustive, rather it introduces applications that we had in mind while realizing the work presented in this thesis.

5.1 Wireless ad hoc networks challenges

To better understand the challenges in wireless ad hoc network and the involved technologies, we will consider the following metaphor. Suppose you want to propagate a message from one side of a crowded room to the other during a speech or a presentation. Of course, you are not able to move from your chair, or to talk too loudly, in order not to disturb the spokesman. An efficient way to achieve your goal is to find a set of person that will propagate the gossip from yourself, the message initiator, to the final recipient.

For this purpose, you first need a **physical medium** to support your message. You can write your message, but suppose you are all computer scientists, so nobody has neither a pen, nor a sheet of paper. You will then use your voice which presents a lot of similarities with the radio waves used by mobile ad hoc networks. Particularly this medium is broadcast, so everybody in a given range is very likely to hear what you say or repeat.

5.1.1 Medium access control

The first element to ensure is that what you say has a chance to be correctly understood by somebody near your. At least, this means you have to ensure that nobody else is trying to communicate at the same time.

This problem of medium access control can be easily solved in a centralized manner, for instance if a moderator was here to allow some people to whisper without disturbing the main speaker. Even in this favorable situation, achieving fairness in allocation, and maintain a minimal access delay is complex to achieve.

5.1.1.1 The hidden terminal problem

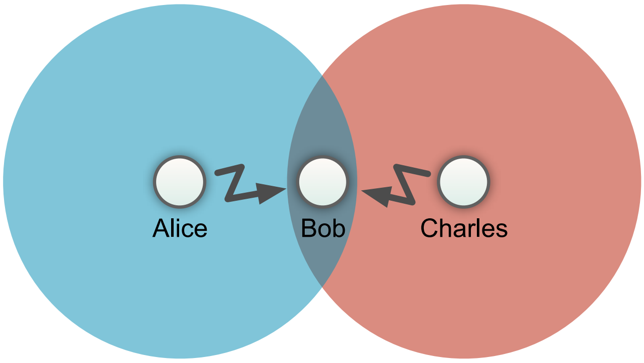


Figure 5.1: An illustration of the hidden terminal problem

Besides, due to their distributed nature, medium access control has to operate without central coordination. Distributed access methods, based on local knowledge only may be subject to the hidden terminal problem: nodes hidden from the potential sender (i.e. for which the sender is not reachable) but able to reach the intended receiver can create collision, as no coordination can be easily established between the potential sender and those hidden nodes. This case is illustrated in the Figure 5.1, where Alice (that hears from Bob but not from Charles) is not able to coordinate transmission with Charles (that can reach Bob but not Alice). This can create collision when both are likely to transmit to Bob.

5.1.1.2 Probabilistic approach vs. scheduling

While some medium access control mechanism rely on a probabilistic approach based on contention (e.g. 802.11[BFO96]) and negotiate access to the channel for each transmission opportunity, others (see [You96, You99] for instance) rely on dynamic resource reservation mechanism.

While the latter allows to either support a fair allocation between the nodes or an adaptive channel allocation, based on quality of service requirements, it is more computationally costly and required a tighter synchronization between the nodes, as it is generally based on some Time Division Multiple Access (TDMA) scheme. Besides, the scheduling is generally operated for the length of a frame. Due to network dynamics, this scheduling may not be valid anymore (two nodes, separated by a large distance, can be scheduled to transmit at the same moment and then move to be in contention with each other during the frame time).

Under dynamic network conditions, none of the presented approaches hence guarantees that all transmissions will happen collision-free.

5.1.1.3 Conclusion

The quality of the implemented medium access control mechanism is particularly important in wireless ad hoc networks, especially when a large number of nodes wants to communicate (either to generate or relay information) at the same moment. This problem is often referred to as the *network broadcast problem* (see definition 4.5). Extreme amounts of broadcast traffic constitute a broadcast storm. A broadcast storm can consume sufficient network resources so as to render the network unable to transport normal traffic.

5.1.2 Routing

Continuing with our metaphor about gossiping, it also appears important that what we want to say reaches its intended receiver, and the sooner the better. We, and every intermediate relay, therefore need to wisely chose who will repeat our message.

A routing protocol is responsible for gathering and disseminating information about the network topology, so that each node is able to determine, at least partially, the optimal path (based on hop count, reliability, energy consumption or any other relevant metric). In wireless ad hoc networks, routing shall be realized in a distributed manner. It is most of the time performed on a per-hop basis (the notable exception are source routing protocols such as [JM96]). Sufficient knowledge is achieved by broadcasting control packets, containing up-to-date information about the partial image of the network topology that each node is able to construct.

In wireless ad hoc network the task of routing protocols is further complicated by the mobility of nodes that induces frequent link breaks, collisions between control packets, and the possible creation of routing loops if the routing tables are not updated. As a consequence, control packets with updated information shall be exchanged with a sufficient frequency so that the routing process can cope with the dynamicity of the network.

This calls for another tight constraint: the overhead generated by the routing protocol (i.e. the total network bandwidth used by the propagation of the control packets) must be kept as low as possible. The throughput available in wireless ad hoc networks is already scarce, due to the shared nature of the medium and the limited bandwidth of wireless technologies, and it shall not be eaten up by management messaging in order to offer the end-users the best quality of service. Particularly, this means that ad hoc routing protocols have to be scalable, and nicely adapt the control packets volume when the network size increases.

5.1.2.1 Neighborhood discovery

Generally, all routing protocols for wireless ad hoc network rely on a link sensing, also called *neighborhood discovery* mechanism: all nodes periodically send a *beacon message* containing their identifier and potentially other relevant information. Due to the broadcast nature of the medium, all nodes within the transmission range of the sender receive and decode this message, making them aware of the existence of the sender. The sender node is then considered as an (asymmetric) neighbor. Symmetric state of links is often achieved by including the list of known neighbors in the next beacon of each node.

5.1.2.2 Reactive routing protocols

Taxonomies of ad hoc routing protocol generally distinguish between reactive and proactive protocols. The former keep the control messages overhead low in absence of traffic and generate additional messages when an original path has to be found for applicative packets. A good example for this kind of protocols is the Ad hoc On demand Distance Vector (AODV) algorithm (see [PR99]), where a network flooding of *RouteRequest* messages is initiated by the message source. Relayed by all nodes in the network a copy of the *RouteRequest* eventually reaches the intended destination. While forwarding a *RouteRequest*, each intermediate node stores the identifier from which it has received this message.

The intended destination, or any intermediate node that has a valid path to the destination in its cache, generates a *RouteReply* message aimed at the source. Nodes that relayed the corresponding *RouteRequest*, forward back the reply, using the stored identifiers. While forwarding a *RouteReply*, each intermediate node stores the identifier from which it has received this message, this allows to construct a distance vector table toward the destination, and help building a cached path to this destination that can be used for a further path request while it is valid.

Rather than finding a local replacement when a link breaks (which could be though as a distance-limited detour in the routing), AODV generates a *RouteError* message towards the source and a new path discovery process shall be executed.

5.1.2.3 Proactive routing protocols

Contrary to on-demand routing protocols, proactive routing protocols aim at constructing paths to all possible destinations at each node of the network. As a consequence, when a message is generated the latency of requesting a valid route to its destination is avoided. However, this advantages comes at the cost of a stable background level of overhead, as control messages are periodically exchanged, even in the absence of traffic.

The arch-example of a proactive routing protocol for wireless ad hoc networks is the Optimized Link State Routing (OLSR) protocol, defined in [JMC$^+$01, CJ03]. In a link state protocol, every node constructs a map of the connectivity to the network, in the form of a graph, showing which nodes are connected to which other nodes. In a dynamic and distributed system like MANETs, achieving such a global knowledge would be hard and would require an unbearable overhead. Therefore, OLSR optimizes the pure link state approach by reducing the number of links used for forwarding, which diminishes the number and the size of control packets.

Based on information from the neighborhood discovery phase, each nodes elects a subsets of its symmetric neighbors as *Multipoint Relays* (MPR). Those nodes take the responsibility for forwarding packets from the nodes that have elected them as MPR (i.e. the *MPR Selectors*). Hence, each node only has to announce links to its set of *MPR Selectors*, and those messages, called *Topology Control* (TC) messages are relayed across the whole network, using the virtual routing backbone formed by all the MPRs. And receiving the TC messages is sufficient to construct a table for routing over this backbone.

However, TC messages still need to be relayed through the complete network, resulting in large delays in delivery. This and the absence of dedicated link breakage discovery leads to inconsistencies in the routing table for highly dynamic networks, where links are not stable.

5.1.2.4 Other approaches

There are also several different or hybrid approaches to tackle the routing problem on wireless ad hoc networks and detailing them all would be beyond the scope of this thesis. However, some widely known approaches are worth mentioning.

Dynamic Source Routing (DSR) is an on-demand routing protocol designed to restrict the overhead generated by control packets as it does not require neighborhood discovery. Instead, the path resulting from a *RouteRequest/RouteReply* similar to AODV is included in the message header. Each relay node simply addresses the message to the next intermediate destination in the path. With mobility, this path may not be valid long enough to avoid routing problems.

Location-Aided Routing protocol (LAR) and other protocols such as the Zone-based Hierarchical Link State (ZHLS) routing protocol take advantage of geographical location information that can be provided, for instance by a Global Positioning System (GPS) terminal, to address some limitations of traditional routing protocols. LAR tries to determine the *expected zone* where the destination of a message may be located (based on its past coordinates and its mobility) and restrains the path request control packets (similar to *RouteRequest*) to a *request zone* accordingly. ZHLS uses static non-overlapping zones to improve its routing operations by performing as a proactive protocol for intra-zone routing and as a reactive protocol for inter-zone routing.

Stability-based routing protocols try to favor stable links for routing. Associativity-based routing (ABR) classifies links as stable or unstable, based on their temporal stability, by counting the number of successive beacons received from the link other endpoint. This definition of stability is similar to the notion of link age mentioned in 2.24. This reactive protocol also uses a *RouteRequest* mechanism, but the destination replies by selecting the path with the maximum proportion of stable links. The Signal Stability-based Adaptive (SSA) routing protocol behaves equivalently but relies on a cross-layer design, as the signal strength of the neighborhood discovery beacons is used to estimate the stability of links.

5.1.3 Security

Securing wireless ad hoc communication may be vital, especially in the context of military applications. However the decentralized design of those networks and the shared nature of the radio medium makes them more vulnerable than wired networks, may it be to passive (eavesdropping without disturbing the network operations) or active (denial of service, etc.) attacks.

Denials of service are particularly easy to achieve by simply jamming the communication at the radio level or by saturating the communication channel with malicious packets. This attack can be countered by using fast frequency hopping with a large variety of frequencies. It is hence hard to jam a important fraction of the frequencies used for transmission. Even if some of them are jammed, this will only impact limited parts of the message and this one may still be recoverable, by instance by also transmitting redundant parts.

Malicious messages, if interpreted as legitimate by the attacked devices, are also able to generate an overflow of the nodes packet buffer, leading to the drop of legitimate information. Finally, malicious node can easily impersonate other devices or concentrate a lot of traffic to eavesdrop by sending erroneous information in the control packet of routing protocols. There are some work to define and formalize the rules of a correct routing behavior and to implement some control checking in order to prevent attacks based on routing [HL04, TBK$^+$03, HTR$^+$04, HHF05].

5.1.4 Energy management

As wireless ad hoc network are primarily designed for operating with mobile devices, it induces that, in most situations, those terminals will hardly have access to a continuous source of power and will rely on a quantity-limited source of energy. This is not true for some kinds of MANETS (like vehicular ad hoc networks), but it is vital for other variants, such as sensor networks, where the power source of devices is very limited. Efficient energy management can be performed by optimizing the following components.

5.1.4.1 Transmission power management

The power consumed by the radio module depends on the state of operation (sleeping, idle, receiving, sending from least to most energy consuming), the transmission power and the technology used for transmission. As packet transmission is the most energy consuming function, it is important to limit the number of packets to be sent, particularly the control packets used by ad hoc algorithms, like routing for instance.

Furthermore, adjusting the transmission power dramatically reduces the consumed energy while sending. Many energy-aware approaches have been designed, especially for network broadcast[LNM03, GC07, RB10b] as, in this case, limiting the transmission power also reduces interferences and diminishes the contention areas. However, this comes at the cost of more relay of messages which induces more end-to-end delay.

5.1.4.2 Processor power management

The other major factor of energy consumption in mobile devices comes from the computational operations. If those operations at the various communication layers (encoding, channel access, routing, application layer) are kept simple, then less operations are needed for real-time processing and the clock of the device processor can be slowed down. Finally, if there exists some computational idle periods, the duty cycle of the processor can be adjusted, to the extent of being completely shut down for time.

5.1.5 Scalability

Wireless ad hoc network are not designed to contain as many nodes as today's Internet and in some application, like military operations, the maximum number of entities in a given network is often set by design. However, the exploding number of communicating devices in our everyday life opens possibilities of wide commercial deployments of ad hoc networks that will help with the offloading of traditional centralized networks (see subsection 4.1.2).

In this context, wireless ad hoc networks shall be able to deliver a quality of service at least equivalent to what user can experience with centralized wireless solutions like Wi-Fi or cellular networks, even with a large number, say a thousand or more, nodes composing the network.

We have already seen that the stability of long path is hard to achieve and solutions involving a large number of relays may not be suitable for demanding applications like video streaming, voice over ip or interactive online gaming. Besides, the latency of path finding in an on-demand routing protocol may be unacceptably high with long paths. On the contrary, the periodic routing overhead, of table-driven routing protocols, generating a lot of control messages, may be particularly inefficient in large and dense networks. In this case, even for constructing short paths, many information shall be exchanged, and the node may spend a lot of resources contending just to exchange this internal information. Finally, large ad hoc networks are likely to contain heterogeneous nodes, with varying resources in power, throughput, computational capability, etc.

5.2 Applications of wireless ad hoc networks

5.2.1 Vehicular Ad hoc Networks (VANets)

Wireless Vehicular Ad hoc Networks (VANets) consist in vehicles traveling on urban streets, capable to communicate with each other with or without the aid of a fixed infrastructure.

We therefore distinguish between vehicle-to-vehicle communications (V2V) when cars communicate between themselves, generally in an ad hoc fashion, and vehicle-to-infrastructure communications (V2I) for data transfer between cars and some fixed access points, generally located in the vicinity of a road. Those roadside relays are generally deployed to increase coverage or the robustness of communications. They can also be used as a gateway to the Internet and, thus, data and context information can be collected, stored, processed, and its results disseminated back to the interested vehicles using multi-hop V2V communications.

5.2.1.1 VANets applications

One envisioned application for VANets relates to cooperative driving systems, particularly path prediction of other cars on the road. This basic component is the building block of cooperative vehicular safety applications who relies on the ability of cars to directly exchange data, a particular advantage of vehicular ad hoc networks which is unavailable through other sensing technologies.

Another important use cases for VANets are convenience and efficiency applications. They comprise Internet access, service announcements, infotainment, payment services, and most notably collaborative traffic information services. The latter, with the help of a centralized traffic management service, can more efficiently route car flows along available roads, leading to diminishing traffic jams and offering users the most time-efficient path to their intended destination. Examples of such systems can be found, for instance in [PDB10]. Figure 5.2 illustrates a VANet system hybrid architecture, using both V2I and V2V, that would allow easy traffic management and planning.

5.2.1.2 VANets challenges

While being conceptually straightforward, design and deployment of VANET is a technically and economically challenging endeavor. Key technical challenges include the following issues:

- **Unfavorable environment for radio propagation:** Urban environment contains many reflecting objects that are likely to degrade the quality of signals. Besides, fading and Doppler shift effect, see subsubsection 6.2.1.3, may also have an important impact on signal quality, due to the possible high relative mobility between sender and receiver;

- **High relative mobility of network components:** This challenges distributed algorithms by causing low link and path stability and very aggressive topology changes;

- **Security and privacy of data:** VANET safety and traffic management application are likely to require access to the current of the user, either to provide relevant data or to validate reported alerts or road status as coming from a legitimate. This might contradict with some privacy requirements for the user;

5.2.2 Mobile Social Networks (MoSoNets)

Wireless ad hoc networks are heavily constrained by the capacity of the terminals and the lack of reliability of the communication channel, especially on long paths[GdWFM02, MZ99]. It is therefore relevant to focus on applications that suit those constraints: social-oriented applications that take benefit of the proximity of users, like enhanced interactivity, gaming or local streaming and relay of media exhibiting a shared interest between co-located individuals.

In this context, ad hoc networks become "tools that support interaction among networked mobile users", which is the characterization of *mobile social services* given by Lugano in [LKS06]. Deployment of social applications over those dynamic networks can be envisioned as the next step further of integration of online social networking in our everyday lives.

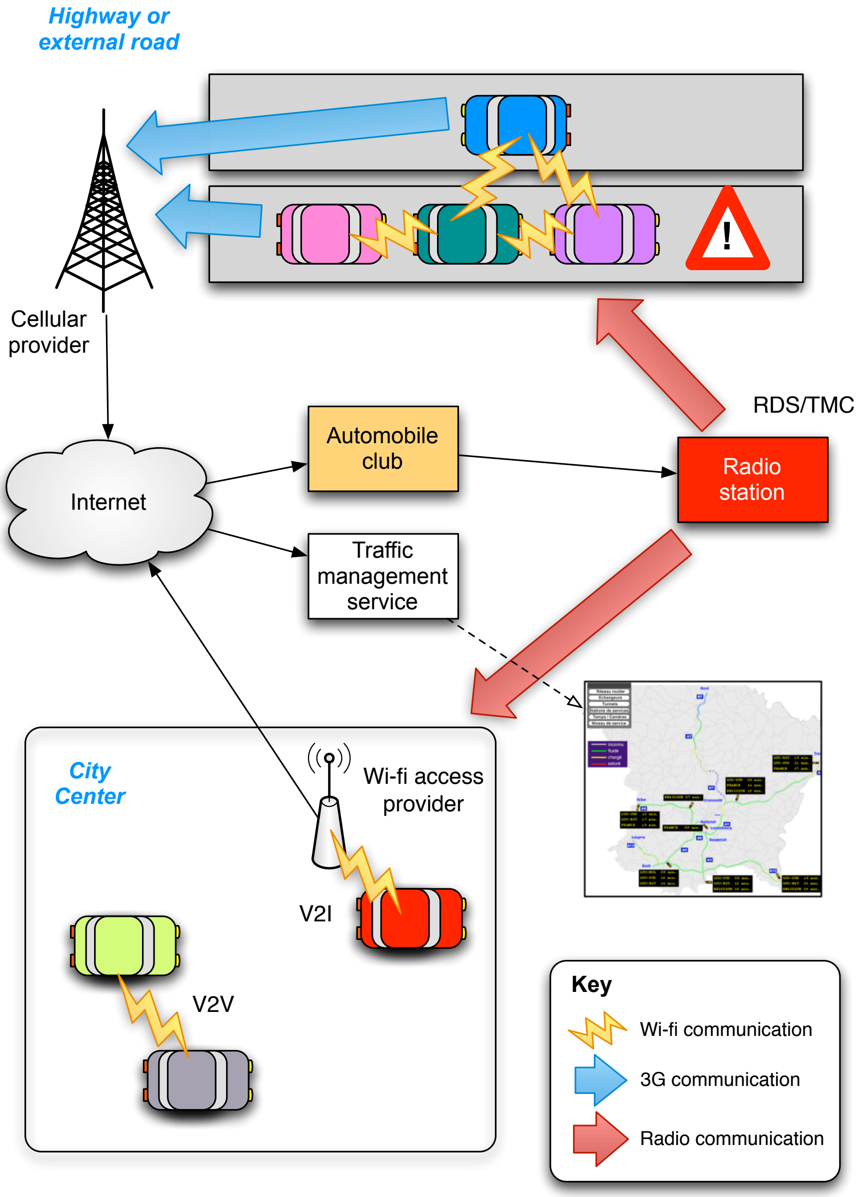


Figure 5.2: A VANet architecture for traffic management and planning

Those dynamic mobile communication networks, when used to facilitate and enhance users social interactions, are referred to as Mobile Social Networks (or MoSoNets). Their main application case considers urban users that will take benefit from their terminal for unprompted communications and data sharing either while on the move (by walking or using private or public transportation means) or when standing in some points of interest, like offices, plants, malls or restaurants.

5.2.2.1 Examples of MoSoNets

CenceMe is motivated by a twofold goal[MLF$^+$08, CEL$^+$08]: first to provide information to individuals about their life patterns and second to provide more texture to interpersonal communication (both direct and indirect) using information derived from hardware and software sensors on user devices. This includes the storage and history of previous locations and activities to determine the *life pattern* of the user, presence indication and report to web-based social networks (such as Facebook) and the possibility to subscribe to the reports of friends. All this should empower advanced social interactions, like detecting where and when friends are meeting. The author also introduces the notion of *significant places*, derived through a continuously evolving clustering, classification, and labeling of users locations so that they can benefit to their friend in search for a interesting place, like a good restaurant.

WozThat is motivated by the idea of bringing rich contextual information from online social networks into the real world of local human interactions, in order to substantially enrich local social interactions. It is presented in [BGA$^+$08]. By being informed via mobile technology of the identity of the person with whom you are seeking to interact and consulting information obtained from that person's public social networking profile, you could more easily initiate a conversation. WhozThat achieves this vision of social interaction through MoSoNet technology by implementing a basic two-step protocol that first shares identities between any two near-by cellular smartphones (e.g., via Bluetooth or WiFi) and then consults an online social network with the identity to import the relevant social context into the local context to enrich local human interaction. Examples of application are, for instance, a context-aware music player in a bar that could adapt its song playlist based on the tastes of the people in the bar, that is, on the social profiles obtained from the identities advertised by the smartphones of people in that bar. The principle of this protocol is detailed in Figure 5.3.

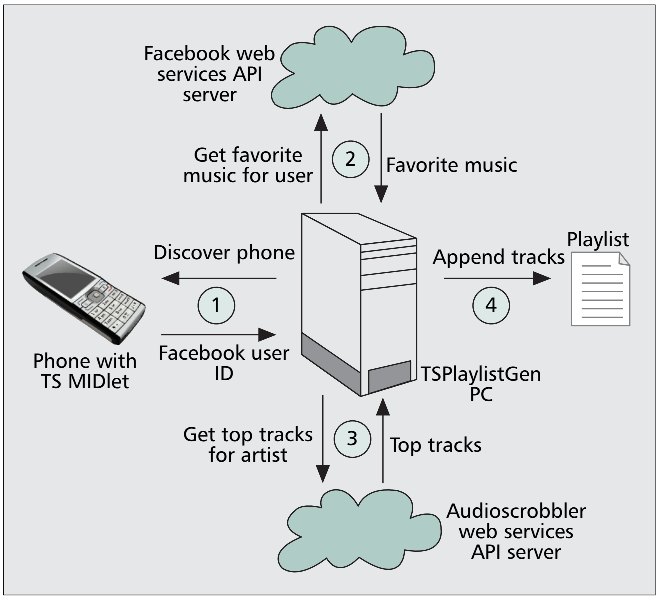

Facebook web services API server
Get favorite music for user
2
Favorite music
Discover phone
1
Facebook user ID
Phone with TS MIDlet
Append tracks
4
Playlist
TSPlaylistGen PC
Get top tracks for artist
3
Top tracks
Audioscrobbler web services API server

Figure 5.3: WozThat architecture for context-aware music playlist generation[BGA$^+$08]

5.2.3 Wireless Sensor Networks

A sensor network is composed of a large number of sensor nodes, which are densely deployed either inside the phenomenon to be observed or very close to it. The position of sensor nodes does not need to be engineered or pre-determined. This allows random deployment in inaccessible terrains or in disaster relief operations.

5.2.3.1 Devices and operation

Each such sensor network node is typically composed of a radio transceiver with an internal antenna or connection to an external antenna, a micro-controller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery or an embedded form of energy harvesting.

Wireless sensor networks generally operate using the following setup: sensor nodes monitor one or several physical or environmental conditions and collaborate so that a report of the observed conditions is available at a *sink* node. This node is, at least logically, an entity where the retrieved information is made human-observable.

The physical dimensions of sensor nodes, which can be in the order of a few cubic millimeters, along with their low costs due to mass production, makes them suitable for many applications: weather and seismological monitoring, inventory control, chemical and biological monitoring, are just a few examples. Wireless sensor networks, can also be used in military applications such as monitoring a conflict area: the generated data by sensors is transmitted to the commandment station to be able to follow the ammunitions, equipment and movements of soldiers in the battlefields. Also, we can find sensors in environmental applications like fire detection: when a node detects a fire in its zone, it reports the data to the sink node.

5.2.3.2 Energy vs. real-time constraints

As the available battery is highly limited, and that some deployment requires the nodes to operate without human intervention (for instance to replace batteries) for an extended period, energy-awareness in communication is an important field of optimization in wireless sensor networks. It is hence desirable to look for a trade-off between network lifetime and fault tolerance or accuracy of reported results.

This energy-efficiency governs the design of all communication layers of wireless sensor networks :

- at **the physical layer** data is generally transferred at a low bit-rate and low power, resulting in reduced transmission ranges;
- **the channel access layer** is generally designed so that nodes remain in sleep mode as long as possible, in order to save energy. Many protocols avoid relying on channel sensing for an extended period of time;
- **routing** generally takes the remaining energy of nodes into account and tries to spread energy consumption homogeneously across the network;
- the **application** is generally designed so that nodes do not report events if nothing is noteworthy. On the contrary, reactive behavior to events is favored. In addition, nodes can also interpret reports from other nodes and aggregate information to avoid false positive reports.

5.3 Highlight

- Wireless mobile ad hoc networks are a **challenged environment** due to their distributed and dynamic nature;
- Medium access control **fails to provide reliable collision-free transmissions** in a distributed and dynamic network;
- Many approaches for routing exist, however they all face some **scalability issues**;
- MANET also have to deal with **security and energy constraints**;
- Despite those drawbacks, **actual applications using wireless ad hoc networks emerge**, including vehicular social and sensor networks.

All those previously announced constraints speak for an **efficient hierarchical topology-based organization** of the network where not only communication-related operations, like channel access or routing, but also the services offered to the end-users would **be tailored to operate locally, using short and robust paths only**.

Besides, the presence of an additional infrastructure network, to create an hybrid architecture may improve scalability.

III

Characteristics of urban wireless ad hoc networks

Contents

Chapter 6

Modeling connectivity

> "The economic transmission of power without wires
> is of all-surpassing importance to man."
> Nikola Tesla, The Transmission of Electrical Energy without wires [...].

The envisioned applications of our contribution all concern users in an urban environment. It is therefore important to evaluate our solution proposals in close-to-reality scenarios, the best solution being concrete experiments in a controlled environment such as a testbed.

However, it is still complicated to perform extensive experimentation on large networks under a controlled environment. Therefore, ad hoc network research heavily relies on network simulation in order to design, assess and improve communication protocols in this challenging context.

Network simulation requires a preliminary modeling effort, in order to capture the significant parameters of the environment that will directly impact the network characteristics and the performances of the applied protocols. Some of the main factors conditioning the network are the propagation model (which impacts the static topology of the network), the node mobility model (which impacts the dynamic topology of the network).

This task is complex to achieve as the captured environment should provide a trade-off between accuracy and complexity so as to limit the simulation computational complexity . Therefore, a proper characterization of what aspects to capture in a model is deemed helpful.

In our modeling process, we will pay a particular attention to capture the elements that constraint the communication capabilities of the involved communicating devices. We will also remind and take into account the human and social factors characterizing the users behind the terminals and their behavior.

6.1 A network model for simulation

Before any modeling effort, it is perhaps important to precisely define what is commonly understood by the expression *network model for simulation*. Therefore, let us consider what are the parameters impacting the network characteristics (mainly its dynamic topology) and performances.

Those factors can be separated into two parts: on one side the *environmental factors*, on the other side the *user-dependent factors* :

- **Environmental factors** account for whatever is possible doing in the model. This includes the propagation model which, based on the distance between users and other refinements, defines whether or not direct communication is possible. Another important feature is the set of constraints applied to users mobility, like restraining movements to streets, limiting the maximal speed or the motion evolution to respect some physical constraints.
- **User-dependent factors** describe whatever is actually done in the simulation span. That is for instance the destination and paths that the users are actually taking, respecting the environmental constraints. This is also the data exchange pattern which depends on the user requests. Those user-dependent factors, to be accurately described, should include the social aspects that may determine the behavior and interactions.

Current research showed that the social behavior of users has an important impact on network characteristics[CHC+07] and made this aspect a new open field for ad hoc protocols improvements, while considering the nature of human interactions in design.

However, most current network models for simulation take into account environmental factors only. Only few approaches include social factors to study network behavior in simulation and those contributions are either based on a custom simulator or is a simple proof of concept[MHM04][MM06].

6.2 Data propagation at the physical layer

Performance of network applications is mostly depending on the volume and pattern of traffic but also on the network topology (i.e. the way users are connected to each other)[GK00, LBDC+01].

The topology of a network is being conditioned by two main factors:

- **the mobility of users**, which depends on geographical (presence of halls, streets, roads) and sociological (points of interest, social relation between users) factors, which models will be presented in the next chapter;
- **the propagation of radio waves** affecting the power at which transmissions between users are received, which depends on the environment where the users are located (presence of buildings and obstacles, etc.)

This section will describe the basics of physical data propagation, as the principles governing the mobility of users are the object of the next chapter. This will help us in choosing an appropriate transmission function T for the simulation of our networks.

6.2.1 Radio waves propagation essentials

In wireless communication systems, information is transmitted using radio signals which propagate using *reflection*, *diffraction* and *scattering* with various relative importance depending on the propagation scenario and environment. Those phenomena are illustrated by the Figure 6.1 below. Besides, a fourth phenomenon, the *Doppler shift* appears when there is a relative mobility between sender and receiver.

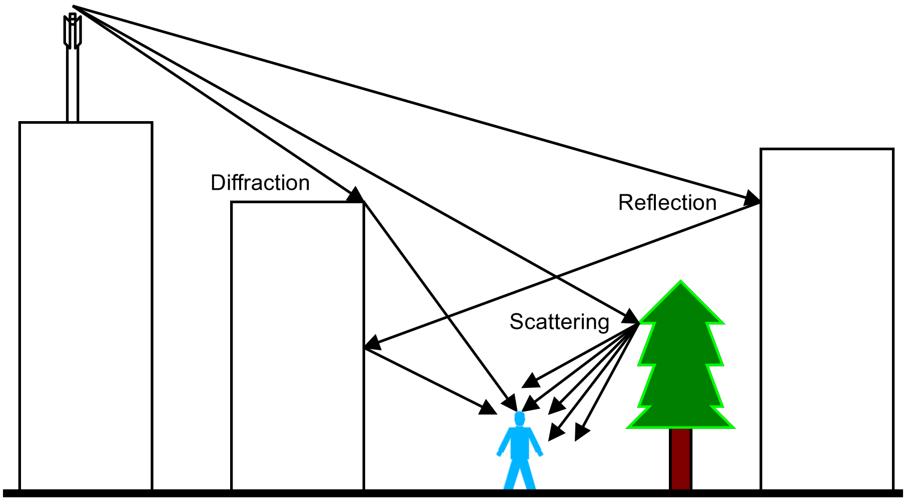


Figure 6.1: Reflection, diffraction and scattering in an urban environment

The combination of those phenomena causes the received signal to suffer from four distinct and independent effects during propagation[St1]:

- a large scale **path-loss variation**, reflecting the attenuation of received signal with distance, generally modeled by following a power law;
- a medium scale **shadowing** or **slow fading**, caused by the *diffraction* of radio waves on impenetrable objects. It is usually captured following a log-normal distribution around the area mean received power;
- small-scale signal fluctuation or **multipath-fading**, due to the reception of out-of-phase signals that have traveled different paths because of the *reflection* of radio waves on large plane surfaces and of the *scattering* by objects of a dimension similar to the used wavelength.
- the **Doppler shift** is caused when a signal transmitter and receiver are moving relative to one another. Under those conditions, the signal is received at a slightly different frequency.

The consequences on the received signal power are illustrated in the Figure 6.2 below.

6.2.1.1 Path-loss

Path-loss is the propagation attenuation effect when a free space environment (without any obstacle nor reflection, refraction and diffraction) is considered. It represents the decay in intensity

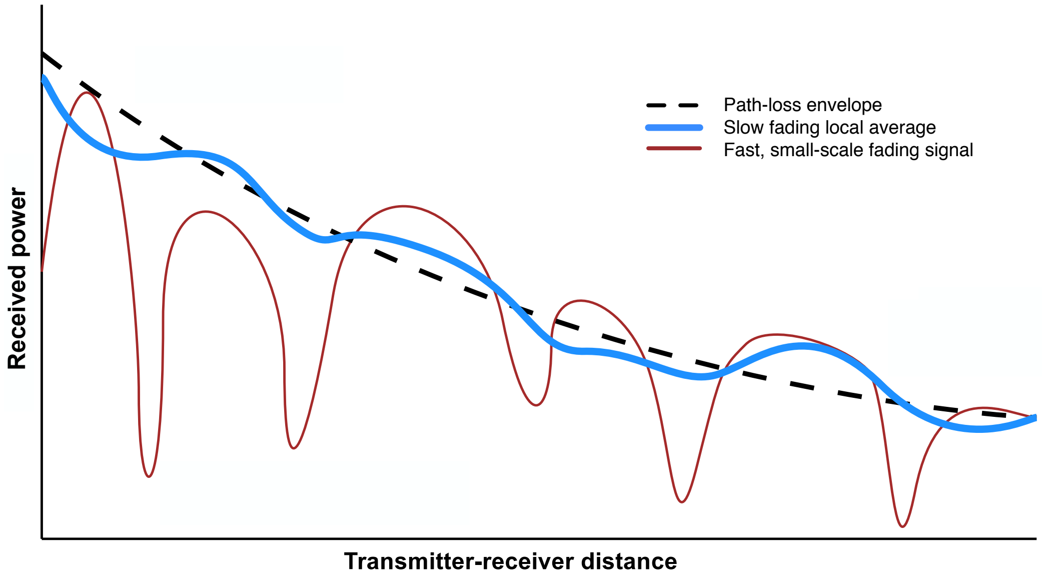


Figure 6.2: Variation of signal level according to transmitter-receiver distance

of the electromagnetic wave with distance between sender and receiver, such that the received power is given by:

$$\Omega_{p,avg}(d) = \Omega_t k \left(\frac{\lambda}{4\pi d}\right)^{\beta} = c\left(\frac{d}{d_0}\right)^{-\beta} \tag{6.1}$$

where Ω_t is the transmitted power, λ is the wavelength, k is a constant of proportionality and β is the path-loss exponent which determines the strength of the attenuation. In the second part of the expression, d_0 is a reference distance where the average received power is known.

Path-loss is most of the time the single considered propagation effect, resulting in each node transmission range being represented as a circle, centered at the node and with a radius depending on the transmission power only.

In more complex propagation models, the received power predicted by the path-loss model $\Omega_{p,avg}(d)$ can be seen as the area mean received power at distance d from the transmitter.

6.2.1.2 Log-normal shadowing

In the log-normal shadowing model, the received power expressed in dBm is given by the following expression:

$$\begin{aligned}\Omega_{p(dBm)}(d) &= \Omega_{p,avg(dBm)}(d) + \epsilon_{(dB)}(dBm)\\ &= \Omega_{p,avg(dBm)}(d_0) - 10\beta\log_{10}(d/d_0) + \epsilon_{(dB)}(dBm)\end{aligned} \tag{6.2}$$

where $\epsilon_{(dB)}$ is a zero-mean normal random variable (in dB) representing the statistical variation in the received power due to shadowing.

The standard deviation σ of $\epsilon_{(dB)}$ is called shadow standard deviation or *power fluctuation* and typically ranges from 5 to 6 dB[Rap02].

6.2.1.3 Multipath-fading

Small case signal fluctuations are caused by the reception of different out-of-phase signals traveling along slightly different paths.

When no line-of-sight component exists, multipath fading is called Rayleigh fading as the instantaneous magnitude of received signal $\alpha(t)$ follows a Rayleigh distribution, named after Lord Rayleigh, that is:

$$p_\alpha(x) = \frac{2x}{\Omega_p} \exp\left(-\frac{x^2}{\Omega_p}\right) \tag{6.3}$$

On the contrary, when a specular or line-of-sight component exists, multipath fading is such that $\alpha(t)$ follows a Ricean distribution. Other multipath fading models exist, such as the Nakagami fading, and can be used to match particular use cases of radio propagation.

6.2.1.4 Doppler shift

Relative sending/receiving station mobility generates the Doppler shift effect. A typical example of this phenomenon is the change in the sound of an ambulance passing by. In such a situation the frequency of the received signal will not be the same as that of the source.

When they are moving towards each other the frequency of the received signal is higher than that of the source, and when they are moving away from each other the frequency decreases.

For a mobile receiver moving along an axis at velocity v and receiving a radio wave of length λ_c from a fixed sender with an angle of incidence θ_n, the Doppler shift is given by:

$$\begin{aligned} f_{D,n} &= f_m \cos\theta_n (Hz) \\ &= \frac{v}{\lambda_c} \cos\theta_n (Hz) \end{aligned} \tag{6.4}$$

This phenomenon becomes important when developing mobile radio systems as it need to be actively combated to ensure proper signal reception in an acceptable range of relative speed between sender and receiver.

Besides, in environments where signal is received through multiple paths, due to diffraction, reflection or scattering, it arrives at the receiver with multiples angles of incidences θ_n, $n \in [1..k]$, causing reception on k slightly shifted frequencies.

6.2.2 Impact on network characteristics

To investigate the sole impact of the chosen propagation model on the network characteristics, we will consider the *lognormal geometric random graph* model, introduced by Hekmat *et al.* in [Hek06]. This model is an improvement of the *geometric random graph model* presented in subsubsection 3.1.1.3.

In this representation, link probability is not only based on path-loss (like in geometric random graphs) but also on the medium scale shadowing component. A fundamental parameter is ξ, defined as the ratio of the signal power fluctuations σ and the path-loss exponent β, that represents the relative importance of the two phenomena.

By introducing the notion of normalized power and normalized distance, the authors show that the expression of the link probability in the lognormal geometric random graph model can take the same form as in the standard geometric random graph model.

Results from simulated topologies using the lognormal geometric random graph model show that coverage fluctuations have limited effect and that the node degree distribution of ad hoc networks follows a binomial law when the border effect can be neglected (i.e. either the simulated area is way larger than the coverage area of a single node or the node density is low, which is equivalent). Topology changes are illustrated in Figure 6.3.

Behavior model and hop count distribution of ad hoc networks depend on the value of the ξ parameter:

- for $\xi = 0$, the network behaves like a regular lattice graph;
- when $\xi \approx 1$, the network tends to a random graph;
- for intermediate values of ξ, the network shows stronger small-world properties.

Therefore, in the proposed model, the hop count in ad hoc networks is a function of the parameters N (number of nodes in the network), ξ and the network region $\mathcal{R}$ size:

- hop count is independent of N when the node density increases over a given service area;
- behavior is dictated by the ξ parameter;
- the hop count increases when the network region dimension increases.

6.3 Data propagation in urban environments

We have seen the impact of the choice of a correct transmission function T on the network characteristics. But if we have integrated physical refinements in our propagation model, we also need to take into account specificities of the urban environment, the main one being the high density of buildings, of variable height, shape, and nature.

Besides confining the vehicular movements to streets, those buildings are physical obstacles that also highly affect radio signal propagation through attenuation, reflection, diffraction, and refraction. This is in addition to the free-space attenuation of radio signals with distance. Since a receiver needs a minimum signal-to-noise ratio to receive data, accounting for the obstacles is important when evaluating urban network through simulations.

6.3.1 Propagation models

As propagation models are used extensively in network planning, particularly for conducting feasibility studies and during initial deployment, they have been widely developed, especially in the urban context.

These models can be broadly categorized into three types: deterministic, empirical or stochastic.

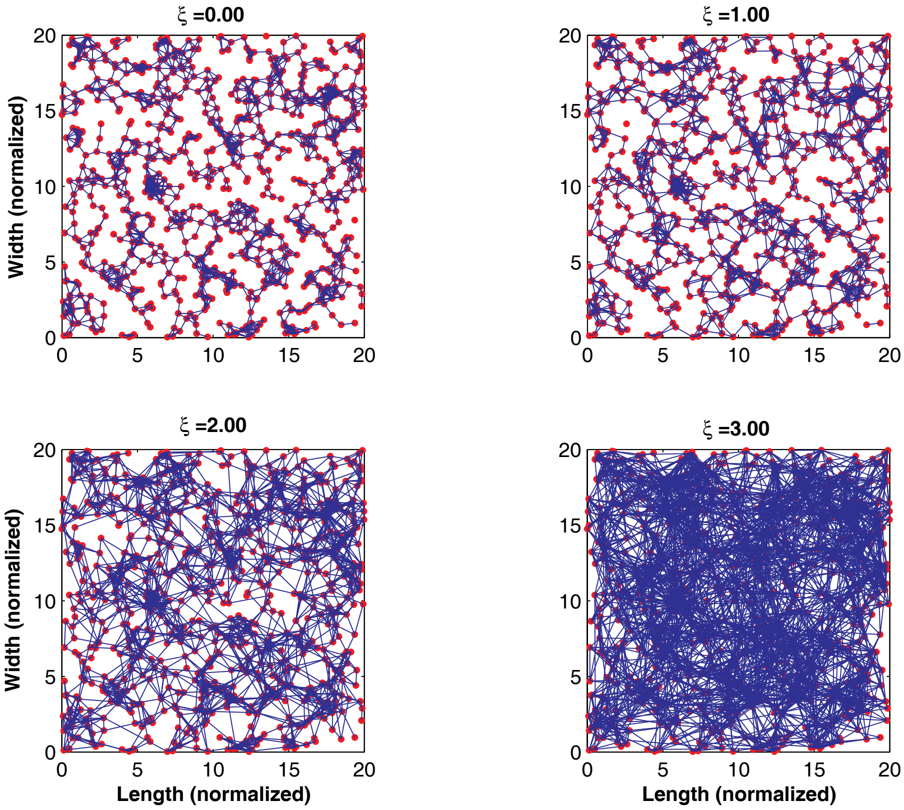


Figure 6.3: Variation of the network topology for different values of ξ for $N = 1000$ nodes and a region of dimension 20×20

6.3.1.1 Deterministic models

Deterministic models make a direct use of the electromagnetic wave propagation laws to determine the received signal power.

They can be very accurate but often require a lot of information about the environment. For instance, they may require a complete tridimensional map of the environment, as well as the reflection index of the materials used at the surfaces of all the elements (building and ground). Obtaining this information requires a large and costly preliminary effort as, for instance, the blueprints of the buildings need to be inspected and the reflection index of each construction material needs to be experimentally studied or otherwise assessed.

Even when the availability of this large amount of data is assumed, deterministic models remain computationally costly. To formally compute the received signal power by each devices means that the Maxwell equations for electromagnetic need to be solved for each receiver, based on all potential senders. Unfortunately, this is not easily achieved, neither analytically nor with numerical methods. In general, approximations are made regarding the nature of the propagation, and *ray-tracing* is one such popular approximation.

Ray-tracing method models electromagnetic waves as rays that propagate according to the laws of geometrical optics. Those rays can undergo numerous reflections, diffractions and transmissions based on detailed information about the positions, shapes and surface characteristics of the urban structures. More details about ray-trancing techniques can be found, for instance in [RWG97].

Despite this simplification, the number of computation remains high, especially in large dynamic ad hoc networks, where each node is likely to transmit (as a source or relay), necessitating the computation of the transmission function for each pair of vertices, at each instant of the network evolution.

6.3.1.2 Empirical models

Empirical models are based on observations and measurements alone. An extensive campaign is realized in different environments with, for instance, varying density of buildings, or different weather conditions. Based on the results of this measurement campaign, a propagation model is determined. These models are mainly used to predict the path loss, but models that predict shadowing and multipath fading have also been proposed[Cra80].

Empirical models can be split into two subcategories, namely time dispersive and non-time dispersive[And03]. The former type is designed to provide information related to the time dispersive characteristics of the channel, i.e. the multipath delay spread of the channel. An example of this type are the Stanford University Interim (SUI) channel models developed under the Institute of Electrical and Electronic Engineers (IEEE) 802.16 working group[EH06]. Examples of non-time- dispersive empirical models are ITU-R[P.101], Hata[Hat81] and the COST-231 model[EH06]. All these models predict mean path loss as a function of various parameters, for example distance, antenna heights etc.

Although empirical propagation models for cellular communication systems have been comprehensively validated, their appropriateness for Wi-Fi based systems has not been fully established. As their conclusions depend highly on some communication-related parameters (like the used frequency, the channel access mechanism, etc.), they are not directly transposable to Wi-Fi based systems.

6.3.1.3 Stochastic models

Stochastic models, on the other hand, model the environment as a series of random variables. These models are the least accurate but require the least information about the environment and use much less processing power to generate predictions.

The degree of accuracy of stochastic models highly depends on the realism of the assumptions made in the model definition. For instance, a stochastic model could completely ignore the presence of buildings (this is the case for the geometric random graph model) or misinterpret their impact on the propagation of radio waves.

Assuming meaningful heuristics are chosen, leading to reasonably sound and trustworthy results, those models are the most computationally efficient. They reduce a complicated computation to a set of conditions : are the sender and receiver close enough for transmission, are they obstructed by a building, does this obstruction prohibit transmission. Those conditions can be efficiently assessed with simple operations related to the environment geometry, completed with stochastic functions for increased second-order accuracy.

6.3.2 Stochastic models for the urban environment

Stochastic models therefore appear as the most suitable for the modeling of data transmission using radio waves in the urban deployment of wireless ad hoc networks. We will analyze some of them, as they represent attempts to model accurately an urban environment.

6.3.2.1 Obstacle-based propagation model

Jardosh, Belding-Royer *et al.* proposed, in [JBRAS03] one of the first models integrating the presence of obstacles in the computation of propagation. In their model, geometric obstacles are placed on the network region $\mathcal{R}$ of dimension $\mathrm{d} = 2$ and are identified by their corners. They can have elaborated shapes, including concavities. All nodes transmit at the same power using omnidirectional antennas.

The obstacles placed within the network have the effect of obstructing node transmissions. Nodes can be either interior of a single building i, or exterior to all buildings (outside). The authors assume the walls of the obstacles prevent the passage of signals between exterior and interior nodes, and between nodes that are interior to two different buildings.

Besides, they model the propagation of signals around obstacles through the use of *obstruction cones*. Because there may be more than one obstacle in the omni-directional transmission range of a node, the node can have multiple obstruction cones.

Definition 6.1 (Obstruction set)

The *obstruction set* $\mathcal{O}(v)$ is then the set of all nodes located in the obstruction cones of a node v. They are formally defined as follows:

$$\mathcal{O}(v) = \{u \in V(\mathcal{G}) / u \text{ is not in line of sight of } v\} \tag{6.5}$$

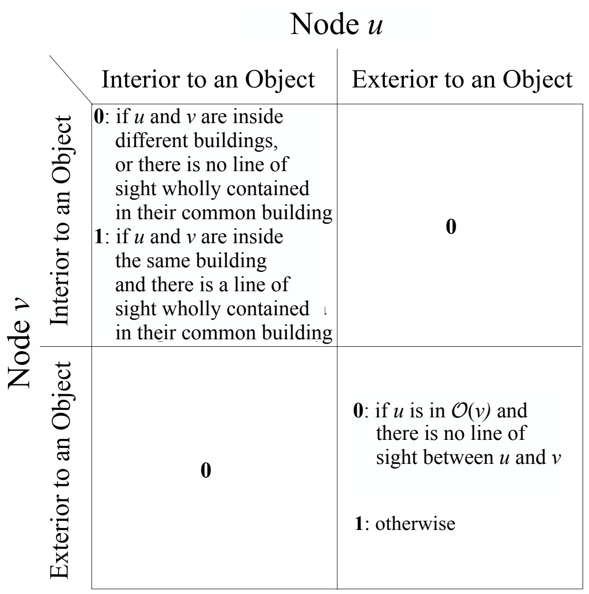
The notion of *line of sight* means that there exists a straight line from sender to receiver, which is not obstructed by any obstacle. Note that in the authors model, the obstruction sets are symmetric, as $u \in \mathcal{O}(v) \iff v \in \mathcal{O}(u)$.

Based on this set definition, the authors derive the *reachability matrix* that governs the connectivity of the nodes, presented in Figure 6.4.

The reachability matrix can be complemented by a distance-based propagation model to figure path-loss and other sources of received power degradation.

Using this model, the authors run a set of simulation and compared results in terms of node density, path length, data packet reception and end-to-end latency with Random Waypoint mobility model a modified version of their model where buildings don't impact wireless transmission.

From those experiments, they concluded that introduction of obstacles slightly diminishes the node density but has a large impact on the number of broken routes and failed connections due to non-existing routes to the destination. They note that the probability of formation of long routes is very low and that only the short routes are successfully discovered.



		Node u	
		Interior to an Object	Exterior to an Object;
Node v	Interior to an Object	**0**: if u and v are inside different buildings, or there is no line of sight wholly contained in their common building **1**: if u and v are inside the same building and there is a line of sight wholly contained in their common building	**0**
	Exterior to an Object	**0**	**0**: if u is in $\mathcal{O}(v)$ and there is no line of sight between u and v **1**: otherwise

Figure 6.4: Obstacle-based propagation model reachability matrix

6.3.2.2 CORNER

In [GFPG10], Giordano *et al.* present CORNER, an urban propagation model, more precisely tailored for VANETs, that implements a propagation model whose formula is presented in [STT05].

The formulae require the knowledge of the position of nodes relative to the underlying road network, or building position in order to classify the attenuation scenario in which the considered pair of nodes are in.

In this model, nodes can be :

- in **line of sight**, with same meaning as explained in subsubsection 6.3.2.1;
- in **non-line of sight** with **one** street corner separating the two nodes;
- in **non-line of sight** with **two** street corners separating the two nodes;

CORNER provides this classification and the geometric computations and information needed to apply the formulae. It assumes that streets are organized in an almost-regular grid, which is often the case in northern american cities, and has been validated experimentally in this context only.

Despite the interest of this approach, it is not easily transposable to scenarios mixing vehicles and pedestrians or in cities where streets are not organized in grids, like in Europe for instance.

6.3.3 Madhoc: Metropolitan Ad Hoc Network simulation

Madhoc is a metropolitan ad hoc network simulator targeting the investigation of ad hoc grid-computing. It has been presented by Hogie *et al.* in [Hog05]. It features the components required for both realistic and large-scale simulations, as well as the tools essential to an effective monitoring of the simulated applications. Madhoc models the physical layer in terms of available bandwidth, signal power and packet size.

Madhoc introduces the concept of *network environment*. The network environment aggregates the mobility model for the stations and a radio wave propagation model. Both the mobility and the propagation of the radio waves are constrained by the environment: obstacles to mobility (like walls) are also obstacles to the propagation of the radio waves. The simulator proposes a set of pre-defined environments, designed with metropolitan applications in mind, that can be adjusted with a set of parameters.

For instance, in the metropolitan model, within a given street, there is no special obstruction to radio waves. Stations on different sidewalks can perfectly communicate with each others (if sidewalks are not too distant). But two stations on two different streets cannot communicate because of buildings that constitute an obstruction to the radio signal.

Besides, it is possible to define concentration areas, or *spots*, that will also impact the mobility of users (see subsection 7.3.2). Those spots are generally buildings or rooms. The model defines that the radio signal that cross the wall of a concentration area keeps only a given percentage of its initial strength. As a consequence, two stations however very close to each other but separated by a wall can seldom communicate.

6.4 Highlight

In the search for a **transmission function** suitable for wireless ad hoc networks deployed in urban environments, we have seen that:

- Radio waves propagation is a **complex phenomenon**, especially in urban context, involving path-loss, shadowing, multipath fading, frequency shift (Doppler effet);
- The choice of the transmission function has a **direct impact** on the network characteristics, including its small-world properties;
- Formally computing radio wave propagation for a dynamic ad hoc network in a city is **computationally complex**;
- To achieve this task **different models** have been established: deterministic, empirical and stochastic;
- **Stochastic models** can achieve **accurate results** with limited computational complexity.

Amongst stochastic models, some of them are **dedicated to metropolitan simulation**. We will rely on them to assess our results through simulation.

Contents

Chapter 7

Modeling mobility

> "The least movement is of importance to all nature.
> The entire ocean is affected by a pebble."
> Blaise Pascal.

For results from a network simulation to be trustworthy, it is necessary for the simulation environment to be as close as possible to the reality. In particular, it is important for the model of users mobility to be carefully designed and to be meaningful (i.e. representing a plausible mobility pattern for the users in the considered use case).

The types of mobility models can be classified as follows[CBD02], based on how they are constructed:

- **synthetic models** that try to reproduce realistic behavior of mobility without the use of traces. Synthetic models use most of the time a random or probability-based behavior.
- **traces models** that replay previously recorded traces in the simulation environment.

Due to the lack of available data from the real-world, most research works rely on synthetic models, despite the risk of inaccurate conclusions on the assessed performances[YLN03]. We can distinguish two modeling approaches for synthetic models:

- **entity models** where each node movement is independent;
- **group models** where nodes movements are correlated. Group models are often composed of a group component and a random additional individual motion.

Some recent efforts tried to tackle this issue by proposing more advanced models by integrating the presence of buildings in mobility and propagation modeling[JBRAS03, HBG06], by using available data from real world experiments to generate a mobility model[KPL08, KK06] or by using city maps to condition user movements[RDK+08].

After reviewing the most common mobility models by trying to construct a taxonomy, this chapter presents specificities and requirements for a proper modeling in the context of urban social networks and vehicular ad hoc networks, two use cases that will be envisioned for application of our algorithm.

7.1 Simple mobility models

Those models are the simplest attempts to capture user mobility in wireless networks. Despite their limitation described below, they remain widely used in simulation and serve as basis for more elaborated models.

7.1.1 Synthetic mobility models

Synthetic mobility models rely on a probabilistic choice of one or several of the mobility parameters (coordinates of the destination, speed, turning angle, pause time, etc.). They allow to easily put in motion a large number of nodes and, by varying the size of the simulation field, the range of available speeds or the pause times, to simulate networks having a wide range of nervousness, from very stable networks to highly dynamic scenarios.

Below are some examples of such models.

Random Walk is a memoryless mobility pattern similar to the notion of Brownian Motion first introduced by Einstein. In this mobility model, each mobile node starts traveling by choosing a random speed and direction. Then, if the entity reaches a specific constant time interval t or a travelled distance d, it may change its speed and direction. These parameters are respectively in the following ranges: $[V_{min}, V_{max}]$ and $[0, 2\pi]$. The new speed is generally chosen using a uniform or Gaussian distribution, the new angle using uniform distribution. Nodes bounce back when they reach the simulation zone boundary.

The Random Walk model was proposed to mimic the movement behavior of the nodes whose movement is believed to be unexpected. It can be considered as the specific Random Waypoint model with zero pause time and bouncing effect.

However, this model generates unrealistic movements with sudden stops and sharp turns, and the movement is limited to a small portion of the simulation area. Besides, it has been shown that random walk makes nodes certainly return to their origin, which may not be realistic.

Random Waypoint resembles Random Walk, but instead of choosing an angle, an intermediate destination within the simulation zone is chosen and reached with a random speed chosen within a specific range. Besides, a random pause time is observed once the destination is reached. Random Waypoint causes a high variability in neighborhood at simulation start and the relation between speed and pause time is complex as long pause time causes stable networks even if nodes move at high speed. Also, Random Waypoint is known to cause density waves with nodes clustering in and out at the simulation zone center. Finally, if speed range includes null or small values, it has been shown that average speed of nodes tends to decay[YLN03].

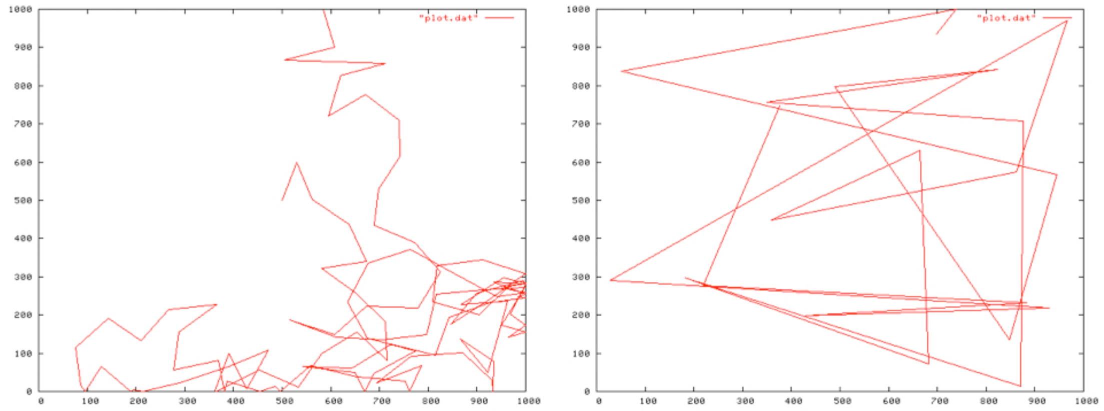


Figure 7.1: Random walk (left) and random waypoint (right) mobility models

Random Direction was proposed in order to avoid the density wave problem which appears in the Random Waypoint model. In the Random Direction model, the moving entity travels along a selected direction until it reaches the edge of the simulation field. The next direction is uniformly chosen along the available angles after a pause time. This way, the entities are uniformly distributed within the simulation field.

7.1.2 Spatial-temporal dependent mobility models

In the previously presented entity models, velocity at a given time is independent of the velocity a the previous epoch, nor related to the position, change in direction or other parameter of the mobile node. This modeling fails to capture realistic behavior of moving entities as, in real life, instantaneous acceleration or stop is impossible, and speed is generally adjusted when direction is changed. Spatial-temporal mobility models try to address those limitations by introducing a coupling or correlation between the different mobility parameters, using their current and sometimes previous values.

7.1.2.1 Entity mobility models

In our description of spatial-temporal dependent mobility models, we will distinguish between entity and group mobility models. While in the former, each mobility decision is taken independently by each node of the network, the latter presents a form of interdependence between the current (or future) position of a node and at least some of the other entities in the network. This section will first focus on entity models.

Gauss-Markov Mobility Model uses the memory of the $(n-1)$ first velocities and directions to avoid sudden stops and sharp turns. Besides, a tuning parameter α allows to manage the degree of randomness in movements (from pure brownian to linear). Finally, by tweaking the final direction parameter, it is possible to avoid that nodes stick to the edge of the simulation area.

Probabilistic Random Walk uses a probability matrix (or an equivalent flowchart) to model a probabilistic rather than purely random movement evolution. Typical value forbids moving from previous to next position without passing by the current position. However, appropriate values for the probability matrix are deemed hard to find.

Smooth Mobility Model has emerged in the attempt to cope with some of synthetic models caveats. More precisely, Bettsteter et al. [Bet01a, Bet01b] developed an approach which prevents sharp direction and speed change, two unrealistic properties of random-based mobility models.

They introduce the *smooth mobility model* in which nodes have a set of preferred velocities (corresponding, for instance, to various speed limits in different types of roads) with higher probability of occurrence. Node velocity changes are decided using a Poisson process and a acceleration/deceleration threshold limits the speed change per unit of time.

Node direction is managed in a similar way. A stochastic process decides for the direction change. When this one occurs, direction is progressively modified during a randomly set *curve time*, until the new direction is reached.

Finally, speed and direction changes are correlated, to model the node slow down during a turn, making this model much closer to real user behavior than simpler synthetic models.

7.1.2.2 Group mobility models

Besides a space-time dependency on the parameters of each moving entities, group models account for the possible interdependence between individuals. This behavior reflects some social interactions (like a group of friends going together to the same place), presence of *hot spots*, or points of interest in the simulation field (like an objective to secure in a battlefield scenario).

Exponential Correlated Random presented is one of the first model introducing the notion of group mobility. Given a position of a mobile node or group, the next position is given by the following relation, called motion function, and Equation 7.1.

$$b(t+1) = b(t) * \exp\left(\frac{-1}{\tau}\right) + \left(\sqrt{\left(1 - \exp\left(\frac{-1}{\tau}\right)^2\right)}\right) \times r \qquad (7.1)$$

where τ adjusts the location change rate, i.e. the smaller τ, the larger the change. Different values of τ are attributed to different groups of nodes, which bind their movement together. Generally r is a random Gaussian variable used to give different positions to nodes of the same group. This model makes however hard to define a realistic motion pattern, based solely on the values of the tuple (τ, r).

Reference Point Group Mobility (RPGM) model uses the notion of *center* or *leader* to derive the next position of all members of a group based on the movement of the leader. This allows to easily model a group mobility, particularly adapted to the movements of soldiers.

Once the *motion vector* of the group leader is decided (by using an entity mobility model), the new position of all members of the group is computed by adding a *random motion vector*, different for each member, to the leader motion vector. Length and direction of this *random motion vector* are chosen uniformly, respectively in a given $[0, L_{max}]$ interval (L_{max} being the maximum length of the vector allowed during the simulation) and in $[0, 2\pi]$.

Column Mobility Model can be considered as variation of the RPGM model as it still relies on the idea of adding a random component, different for each moving entity, to the deterministic group-related movement pattern.

However, in the column mobility model, the reference point is not the position of some member of the group. Each entity has its own reference point, aligned on a line, or *column* hence the name, for each group. This line of reference points moves using an entity mobility model. This model was designed to mimic behavior of soldiers charging or clearing a zone of mines, or rescue teams scanning an area for survivors.

Pursue Mobility Model is another variation of the RPGM model. Here, the members of a group move toward a common moving objective point. All entities have a limited speed and a random direction variation. Speed variation is related to the distance between the considered node and the target, so entities slow down when they are closer to the objective. Finally, mobility parameters of the objective are also periodically updated. This model is appropriated to simulate cops chasing a criminal or similar scenarios.

Nomadic Community Mobility Model is close to nature-inspired group mobility models, such as boids[Rey87]. In this model, group coherence is achieved by having the entities all moving towards the same direction while trying to maintain a distance between each other in order not to collide. This generates relatively random relative movements inside a group although all the members globally move along to a given destination.

7.2 Environment-aware mobility models

Environment-aware mobility models do not assume that the network region is a full free space. They consider the presence of buildings and obstacles that will impact, not only the radio waves propagation as illustrated in subsection 6.3.2, but also the mobility of users.

7.2.1 Heuristic-based

As the models presented in subsection 7.1.1, those models rely on heuristics and stochastic methods to model the mobility of users in a more complex environment.

7.2.1.1 Pathway mobility model

In their article[THB+02], Tian *et al.* use a connected graph to model the mobility constraints of the nodes, the graph vertices being the set of intermediate destination and the edges streets or train connections between those positions. Nodes travel between randomly-chosen destinations using the shortest available path and pause each time the chosen vertex is reached. This work was developed in order to assess performance of different routing protocols under more realistic scenarios. However, this model, contrary to obstacle-based mobility models like the one presented below, does not propose any associated propagation model.

7.2.1.2 Obstacle Mobility Model

In [JBRAS03], the authors present the Obstacle Mobility Model which allows to insert buildings inside a simulation area. Those buildings will act as potential destinations for the mobile nodes, determine their pathways and have an impact on wireless transmission. We have presented the corresponding transmission model in subsubsection 6.3.2.1.

Using the description of the building size and place, defined as an ordered list of corners coordinates, the model defines pathways using the Voronoi Graph technique[dBvKOS00]: all the paths are equidistant to the surrounding buildings. Intersections of those paths with building serve as entering doors. Nodes randomly select a destination building, and walk along the pathways on the shortest path to their destination at a random speed within walking speed range, then pause for a random time at their destination.

This work is however more about a realistic propagation model, rather than a real mobility model. Even though the authors say that the Voronoi diagram quite matches the real pathways of their simulation case (a subpart of the University of California at Santa-Barbara campus), the mobility model seems quite limited as it in fact only offers a modified version of Random Waypoint along the paths defined by the Voronoi graph. No special attention has been ported neither to the fact that destination building choice is not purely random, nor to define some group mobility mimicking behavior of students and faculty around a campus.

7.2.2 Trace-based

While still integrating heuristics to some extent, those model also rely on information recorded in the real world, most of the time traces of the mobility of a set of users for a given period of time. While this may increase the accuracy of the model in the chosen scenario, any important variation in the context requires the collection of the corresponding data to maintain validity of the results.

7.2.2.1 Graph-Based Mobility Model

In [KPL08], Koberstein *et al.* introduce a mobility and propagation model for simulation in an urban environment. Based on GPS data or maps from the OpenStreetMap[OSM], they extract a graph $\mathcal{G}$ representing the city section in which the nodes will evolve: the vertices V being the intersections and road bends, the edges E being the road sections. This model is built on the OMNeT++ network simulator.

Parameters stored in each graph components allow to store waiting probability, average waiting time and associated standard deviation on vertices and average speed and associated standard deviation on each edge. In this model, nodes motion is determined by a succession of random choices: choice of the next edge, of the speed, of the waiting time at a vertex.

The corresponding propagation model uses the direction created by the edges to generate two cones of preferred propagation (parallel to the edge) and two cones of reduced propagation (orthogonal to the edges) to account for the building obstructing nodes in different streets.

These cones refine a more generic randomized unit disc propagation model and parameters of the model have been matched to real-world traces using an iterative process trying to minimize the differences between real and simulated results on two applicative performance indicators.

While taking into account the environment for propagation and mobility modeling, this model still relies on a probabilistic approach to govern the nodes movements. Besides, nothing is said about the inner behavior of the nodes when they have to change speed and/or direction. Finally, this model doesn't capture any social aspect (like points of interest inside the city to which nodes are more likely to converge). This aspect could be used to generate a group component to the mobility model.

7.2.2.2 Vehicular mobility model based on traffic counting

In [PDB11], Pigné *et al.* propose a new vehicular mobility model based on real traffic counting data. It relies on both traffic data collected by the local road network authority and geographical information about the nature of different areas. Their contribution uses both sources to generate a novel realistic vehicular macro-mobility model which allows the simulation of vehicles flows at a city or region scale.

The information about traffic data, collected for instance by induction loops on roads, gives precise information about flows of vehicles with a high precision in terms of time distribution and geographical location. However, no information about the actual origin and destination of vehicles is recorded and a path planning heuristic is developed to interpolate those locations, based on the notion of time-dependent attractiveness of geographical zones. As an example, destinations at commuting hours are certainly more concentrated on work places than on residential places. Moreover limiting the source of traffic to the counting loops positions would not be realistic either. As a solution, this mobility model includes a comprehensive and tunable model for the selection of destinations based on geographical attraction points (work places, home places, etc.) extracted from real data maps.

The geographical zones characteristics (location, shape, size, type) are extracted from data provided by the OpenStreetMaps (OSM) project (see [OSM]). From this information, the probability for a car to chose a given zone as destination is determined using three main parameters:

- the weight of the zone type, which depends on the current time of the day (for instance workplaces and malls are less attractive during the night);
- the attractiveness of the zone, which depends on the proximity with a city-center or a dense population area;
- the surface of the zone, as it is assumed that bigger zones are likely to attract and host more users.

Based on this model, the choice of a type, attractiveness area and zone is computed as three non-independent events. This allows to determine the macro-mobility model. Micro-mobility is simulated using the SUMO[SUM] traffic simulator.

7.3 Social-aware mobility models

The previously presented models consider, with a coarser to finer grain, the importance of the laws of mechanics, of the environment and the possible interaction between the member of a group. However, none of them really takes into account the user, in his social aspects, behind the moving entity. This requires an analysis of the rationale of mobility, for instance in an urban environment, like in our envisioned application.

In order to develop a social mobility model, it is important to have a complete idea about the social context of the users in movement. So, it is necessary to consider the social relationships that may exist between the entities as they will tend to make them share a common location, at least for some period of time (we can think of co-worker, or friends meeting occasionally at a bar or a restaurant). We also have to consider that, especially in an urban environment, the location of those meeting places is not random, and that their attractiveness is likely to change over time.

7.3.1 Social Mobility Model

٢

This mobility model, presented in [Her03] uses the notion of *anchor*, also often referred to as *point of interest* or *spot* in the literature. Very common in social-oriented mobility models, they correspond to potential places, like restaurants, malls, stations, bars, etc., where mobile entities are likely to join.

In the Social Mobility Model the simulation time is split in rounds of a fixed number T of units of time. During each round, a mobile node has to visit all the anchors of a given list in the same order. This list is set at the beginning of the simulation and does not vary. Therefore a round can be assimilated to a day, where all users will commute between home and work, go to the same place for lunch, go to another fixed location after work, then go back home.

The original input of the social mobility model is a randomly generated relationship graph between the moving entities. For all cliques of this network an anchor is created and added to the schedule of all members of the clique, i.e. a clique in the relationship network represents the fact that all nodes will meet at a given time of the round. The schedule is determined so that there is no conflicting time intervals (no entity has to be at two anchors at the same time).

While this mobility model introduces an interesting incentive for representing real-life social interaction between users, it lacks a realistic modeling of low-scale mobility. Anchors are just placed on a grid layout, without any geographical relevance and the way entities move between anchors is not described at all in the publication presenting the mobility model.

7.3.2 Human Mobility Model

The Human Mobility Model tries to mimic the behavior of users moving from a point of interest to another and stopping for a certain time at each point of interest before moving to the next one. It was introduced by Hogie in [HGB04].

The Human Mobility Model defines that the simulation area is a bounded area. This area contains a set $\mathcal{HS}$ of hotspots which act as intermediate destinations for the nodes. A hotspot is located by its coordinates. It is a round area defined by the value of its radius. Spots should not overlap. Nodes can be located either in or out of spots.

The mobility model used by the node which moves out of a spot is a variant of the random walk mobility model which respects the mass mobility constraints.

On the first hand, when a node n is located out of any spot, it must choose a destination spot $D_S(n)$. To do this, it maintains a set $\mathcal{VS}(n)$ of the spots it visited in the past. A node cannot visit twice the same spot. $D_S(n)$ is chosen so as it is the closest spot which has not yet been visited (which does not belong to $\mathcal{VS}(n)$). The node then moves towards its destination spot, with a certain degree of random variation in its direction. The angle range a node can choose from is bounded by the two tangent lines to the spot passing by its current position and the direction is adjusted at every move step. Those requirements ensure that the node will eventually reach its destination spot at one moment.

On the other hand, when the node n is within a spot $D_S(n)$, it moves according to a fixed direction mobility model. The current spot $D_S(n)$ is added to $\mathcal{VS}(n)$ when entering, the node pauses for a random bounded period and a new destination spot is defined. After the pause time, the node starts moving in straight line to the center of the next spot.

If all spots have been visited ($\mathcal{VS}(n) = \mathcal{HS}$) then $\mathcal{VS}(n)$ is emptied and the search for a new destination is triggered again. After each pause time, the node starts moving from speed 0, and increases its speed with a bounded variation at each movement step.

While it presents a detailed large and low scale mobility model, and abstracts the social aspect of human mobility by using the spots, the Human Mobility Model does not completely capture the complexity of human movements. This is especially due to the simple heuristic governing the choice of the next spot to visit (nearest non-visited spot). Besides, as the sequence of visit is set independently for each entity, it might not reflect any inner social relation (for instance office mates, or gym club buddies) that can exist between the modeled users.

7.3.3 Working Day Movement Model

The main idea behind the development of such a model, developed in [EKKO08], is to reproduce people daily activities in an urban environment, inspired from real users schedules: people can be either at home, or at work, or having activities in a given spot with friends.

At the beginning of the simulation, nodes start from their home location at a wake-up time assigned from a normal distribution with a zero mean. At this time, nodes depart from their homes. They utilize diverse means of transport to go to their workplace such as cars or buses. All transportation means are simulated using dedicated relatively accurate low-scale mobility models.

Once the nodes reach their workplace, they enter their office from a particular entry point called door. All the nodes inside the square are located by their desks coordinates The employees movements and pause times inside the office are supposed to follow a Pareto distribution. When the work is finished, each entity is supposed to join a preferred spot, mimicking evening social activities like going to shopping centers, bars, walking, etc.

If a meeting spot is full, a new one is formed with a arbitrarily chosen and uniformly distributed dimension with minimum and maximum values defined in the simulation settings. Nodes could reach the spot using the same transportation model or each node could chose his own model and join the rest of the group members at the gathering spot. After that, the group finally divides and each member goes back to its home residence.

Presenting a fairly elaborated model for both large and low-scale mobility, the working day movement appears as one of the most realistic solutions for simulating behavior of users in an urban environment. However, if the mobility of each day phase (being home, working, socializing after work) is thoroughly described, the description of the interaction, and the model of the social behavior of nodes (with just three identical activities for all entities) is rather simplistic.

7.4 Urban mobility

So as to overcome the shortcomings of the previously described models, we designed UrbiSim[HB09] to simulate the behavior of human *users* with communicating devices in an urban environment, represented by a set of *pathways* (straight portions streets, roads, etc.) and *spots* (buildings, parks, meeting points, etc.). For sake of realism in both the physical and sociological aspects, the mobility of users is modeled at two different levels: at small-scale for local speed and direction control, at a larger scale in order to assign meaningful intermediate destinations based on an user profile. We will now present in further details the components of our model: the urban environment, the user profile and the mobility management at large and small scales.

7.4.1 Urban environment

The urban environment used for this model consists in a set of spots linked together by different interconnected pathways. It can be generated stochastically using the Voronoi graph technique or might rely on an external source (like OpenStreetMap[OSM] which uses a similar way to describe its maps using XML).

The set of pathways is stored as a weighted directed graph, where the weight of an arc is the expected time that may be required to traverse this arc. When a node wants to reach a distant spot, it simply runs a shortest-path algorithm on the stored graph from the spot at which it is currently located. For this computation, two modes are proposed:

- In the first mode, the weight of each graph is simply the product of its length by the maximum speed allowed. This behavior is similar to a Global Positioning System (GPS) device planning for the shortest trip in time.

- In the second mode, the traversal time is not fixed during the whole simulation. It is now also dependent on dynamic parameters like the current congestion of the pathway on this direction: the more cars are currently on this arc, the lower the maximum reachable speed will be. As those value may evolve, the path chosen by a user is recomputed at each intersection. This behavior is similar to what can be observed with a GPS device aware of traffic information.

7.4.2 Mobility

The major contribution of this framework is also to introduce a way to model the behavior of human users in their everyday life using the concept of *social groups*.

7.4.2.1 User profile

As a configuration of the model, a set of social groups might be defined, which will set at which spot and at which time members of the groups should go and how long they should stay inside this spot. Then, each simulated user is assigned a *profile*, composed of the list of groups it belongs to. This allows to easily model the fact that some users belong to the employees of a given company, meet a group of friends at given time and place, or registered to the same gym class after work.

When users are found in a period of time when no destination spot is imposed by their profile, or if no profile is defined, a more naive random weighted choice of preferred spots is used to simulate a human spontaneously going to some of its favorite places.

This concept of user profile allows to study the impact of an heterogenous mobility pattern, alternating between transfers from a spot to another and more static stays within a spot, with different time-scales of interaction with users met more or less frequently.

7.4.2.2 Large-scale mobility

Large-scale mobility governs the way a user will globally move during the simulation time, using a finite-state machine (FSM) paradigm. In the current implementation, large-scale mobility is ruled by the FSM presented in Fig. 7.2 and described below.

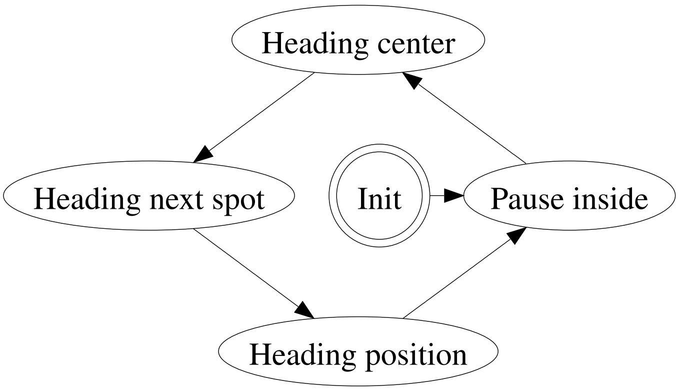


Figure 7.2: Simple large-scale mobility finite state machine

At simulation startup, after initialization, each user starts with a random pause time inside its original spot. After this waiting period (or if its profile requests to go to another spot), the user aims for the spot *center*, which is actually the doorway to exit the current spot. Once reached the exit of the current spot and joined to closest pathway, the user heads for its destination spot. During this phase, its mobility is managed by the *short-scale mobility* finite-state machine presented in section 7.4.2.3. When the destination spot is attained, the user enters and goes to a random place inside the building and starts another waiting period (set either by its profile or by the weight of this spot regarding this user). All movements within a spot are made under walking speed and currently follow the Random Walk mobility model.

7.4.2.3 Small-scale mobility

The model we present uses a separate finite-state machine that is responsible for preventing the user movements from being unrealistic at a small-scale, by preventing sharp turns and instantaneous speed changes. But in the current implementation, this automaton is also used to mimic more closely the behavior of motorized vehicles in city streets.

As presented in 7.4.1, the urban environment is composed of a set of straight directed street sections, called pathways, each having a maximum reachable speed, either set statically (depending on traffic laws) or dynamically depending on the current congestion status of this section.

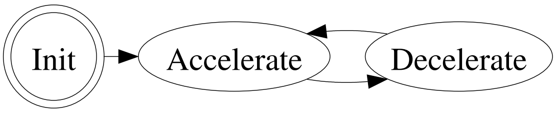


Figure 7.3: Simple Small-scale mobility finite state machine

While entering a pathway, a user will progressively accelerate, trying to reach the current maximum achievable speed on the current street section, but it also computes the speed at which it should reach the intersection at the end of this pathway. This end-of-pathway or *target speed* depends on the nature of the intersection. If it is a simple road turn (only two pathways intersect) then the target speed is correlated with the angle the user will have to turn (bigger turning angle resulting in smaller target speed). On the contrary, if the intersection is a road crossing (more than two pathways intersect), then the user shall mark a stop and the target speed is set to zero.

Based on its breaking capacities, the current and the target speed, and its distance to the end-of-pathway intersection, the user might start to break. Depending on the pathway length, its current maximum speed, and the user characteristics, the breaking phase can occur before the maximum pathway speed is reached.

7.5 Highlight

According to this chapter, we can reckon that

- Mobility of users is **as complex as transmission** of radio waves to model;
- Many models for mobility have been presented in the literature, but they all have caveats, and lead to results with only a **limited validity**, especially in urban environments;
- Mobility has also a **direct impact on the density, quality and stability** of links;
- More recent models include **social impact** on the mobility of users, and consider more specifically the urban context.

As a consequence, for simulation results to be trustworthy, it is necessary to avoid simple synthetic models and rely on more sophisticated models, like those presented in section 7.3 and section 7.4.

IV

New techniques for distributed community detection on dynamic graphs

Contents

Chapter
8

Distributed community detection on static graphs

> "Community can be defined simply as a group in which free conversation can take place. Community is where I can share my innermost thoughts, bring out the depths of my own feelings, and know they will be understood."
>
> Rollo May, Power and Innocence.

In chapter 6 and chapter 7, we have seen how the particularities of the urban environment and the specificities of urban mobility, especially when the corresponding social incentives are considered, are likely to generate dynamic ad hoc communication networks with inhomogeneities in both degree distribution and connection duration.

In subsection 3.3.3, we have seen that none of the existing techniques and algorithms used so far in social network analysis suited the specificities and design requirements of dynamic networks. Particularly, we have seen that no decentralized solution presented in the literature, although being the only applicable approaches suitable for such networks, was correctly integrating local information about the network history and topology.

In this chapter, we will present the first part of our major contribution, consisting of novel approaches that improve the distributed detection of communities in static graphs and introduce an incentive to create communities based on edge weights, which can represent, for instance, the stability of the quality of the associated communication channel.

8.1 Improving distributed community detection

An efficient social analysis technique for dynamic networks shall prevent the creation of *monster* communities, especially when the inner and inter-community link densities are close. Besides, it shall be resilient to the underlying changes in the topology (while keeping the same overall characteristics) or the initialization parameters; that is make the assignment more robust, in the sense of the robustness given by Siegel *et al.* in definition 4.4 and studied for community detection in [KLN08].

In this section, we will introduce SHARC (Sharper Heuristic for Assignment of Robust Communities), a novel technique for distributed community detection, based on epidemic label propagation, that composes an improved constructive mechanism of community structures.

8.1.1 Neighborhood similarity

The limited performance of traditional epidemic label propagation algorithms can be explained by their limited use of simple information about the adjacent vertices only. However, it has been reckoned in several fields[CME05, Col88] that the tendency to join a community depends not only on the number of connected peers already in this community, but also crucially on how these peers are connected to one another[BHKL06].

8.1.1.1 Motivation

To express this dependency, we will use the concept of common adjacent vertices set, or *neighborhood similarity*, as heuristic in the community assignment process. This requires each vertex to know not only its set of adjacent vertices, but also the relations (or connections) of those vertices. This can be achieved with a knowledge of the two-hop neighborhood (see definition 2.5). Such heuristic will allow to make the mining sharper and more robust to network changes.

Besides, the assumption of two-hop neighborhood knowledge is realistic, even in dynamic ad hoc networks, where it should be updated in a distributed manner. For instance, it could be acquired through the periodical exchange of beacons containing the identifier and the set of neighborhood of the sender. Many algorithms rely on this knowledge, either for routing (like the Optimized Link State Routing protocol presented in subsubsection 5.1.2.3) or for creation of communication backbones (like the Blackbone2 algorithm we will present in chapter 13). Such use of two-hop neighborhood knowledge is also present in algorithms controlling broadcast of information across communication networks, like DFCN (Delayed Flooding with Cumulative Neighborhood), presented by Hogie *et al.* in [HGB04].

8.1.1.2 Definition

In definition 3.9, we have presented the triadic closure rule, first introduced by Rapoport[Rap57]: if A is a friend of both B and C, then B and C are also very likely to be friends. Triads not respecting this closure rule explain the existence of links between communities, as exhibited by Granovetter[Gra73].

With our standard graph theory notations, as a consequence of the triadic closure rule, two neighbor vertices u and v of a network $\mathcal{N}$, belonging to the same community $c = C(u) = C(v)$, both have a high link density with other vertices of c and should hence have a more similar set of adjacent vertices than with a given vertex n, not belonging to c.

The size of this common adjacent vertices set, can be used to assess the likelihood of both u and v belonging to the same community. However it needs to consider the ratio of common neighbors over the total node degree, in order not to favor high degree nodes.

Therefore, we propose the following definition of the *neighborhood similarity measure*:

Definition 8.1 (Neighborhood similarity)

The *neighborhood similarity measure* n_{sim} is defined as, for any vertices $u, v \in V(\mathcal{N})$ with either $|N(u)| > 0$ or $|N(v)| > 0$:

$$n_{sim}(u,v) = 1 - \frac{|(N(u) \setminus N(v)) \cup (N(v) \setminus N(u))|}{|N(u)| + |N(v)|}\ . \tag{8.1}$$

The neighborhood similarity of a node u and a node v both of degree 0 can, by extension, be defined as:

$$n_{sim}(u,v) = \delta(u,v) \tag{8.2}$$

The computation of the neighborhood similarity measure is based on simple set operations and can be implemented in linear time (using sorted lists for instance). Therefore, this method is running in $O(\Delta)$, where Δ is the network average degree. An implementation of the neighborhood similarity measure is presented in Algorithm 1.

Algorithm 1 Neighborhood Similarity (NS)

```
Data: $u$: vertex for which the neighborhood similarity is computed
Data: $N(u)$: one-hop neighborhood of node $u$
Data: $N^2(u)$: two-hop neighborhood of node $u$
Output: $C(u)$: Current community of node $u$
Output: max_score: Score of current community of node $u$

$C(u) \leftarrow u$
community_score ← []
max_score ← 0
for all $v \in N(u)$ do
    c ← $C(v)$
    community_score[c] ← community_score[c] + $n_{sim}(u,v)$
    if community_score[c] > max_score then
        $C(u)$ ← c
        max_score ← community_score[c]
    end if
end for
return $C(u)$, max_score
```

The neighborhood similarity linearly varies between 0 when the assessed nodes have no neighbor vertex in common and 1 when the neighborhoods are identical. It is a normalized value which does not favor high or low degree vertices. This measure is also symmetric for any pair of adjacent vertices of the network $\mathcal{N}$.

8.1.1.3 Using neighborhood similarity in community assignment

As in other epidemic algorithms, SHARC accounts for each neighbor vertex community label $c \in \mathcal{C}$ and assigns any vertex u to the community $C(u)$ with the highest count. However, in our algorithm, neighbors do not contribute equally, as the Equation 8.3 shall be verified:

$$C(u) = \arg\max_{c \in \mathcal{C}} \sum_{\substack{v \in N(u) \\ C(v)=c}} n_{sim}(u, v) \tag{8.3}$$

Each neighboring vertex v contributes to the count of its community label $C(v)$ at node u to the extent of its neighborhood similarity with the vertex u. This heuristic therefore strengthens the gap between community counts as it considers both the number of neighboring vertices with this label and their value in terms of neighborhood similarity, which is high for tightly interconnected nodes. This is the principle of the original SHARC algorithm introduced in [HB10] and its implementation is presented in Algorithm 2.

Algorithm 2 Sharper Heuristic for Assignment of Robust Communities (SHARC)

Data: $\mathcal{N}$: a network

Output: $\mathcal{C}$: set of communities of the current network partition

```
$\mathcal{C} \leftarrow V(\mathcal{N})$
loop
  for all $u \in$ Shuffle($V(\mathcal{N})$) do
    $C(u) \leftarrow NS(u)$
    if $C(u) \notin \mathcal{C}$ then
      $\mathcal{C} \leftarrow \mathcal{C} \cup C(u)$
    end if
  end for
end loop
return $\mathcal{C}$
```

The main loop of the algorithm contains one iteration of the algorithm. It is made to end after a certain number of iterations on static networks, for instance, or to be executed periodically until the network ceases operation on dynamic networks.

Each iteration of SHARC is simply the execution of the neighborhood similarity computation, presented in Algorithm 1, for all the neighbors of the considered node. Therefore, it is running in $O(\Delta^2)$, where Δ is the network average degree.

8.1.1.4 Mathematical model

From the previous definitions, we can derive a mathematical model of the community assignment using the SHARC algorithm.

Let $\mathbf{A} = [a_{ij}]$ be the adjacency matrix associated to a graph $\mathcal{G} = (V, E)$. This graph can, for instance, be a snapshot of the evolutive graph representation of a network $\mathcal{N}$ (see definition 2.22).

Each element a_{ij} of $\mathbf{A}$ is defined as follows, $\forall (i,j) \in V \times V$:

$$a_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are adjacent in } \mathcal{G} \\ 0 & \text{otherwise} \end{cases}$$

Let $\mathcal{C}$ the current community assignment of $\mathcal{G}$. We have $\mathcal{C} \subset V$. Hence, we can define the *assignment matrix* $\mathbf{C} = [c_{ik}]$ as follows, $\forall (i,k) \in V \times \mathcal{C}$:

$$c_{ik} = \begin{cases} 1 & \text{if } i \text{ is in community } k \\ 0 & \text{otherwise} \end{cases}$$

Further, we can define the *strength matrix* $\mathbf{S} = [s_{ik}]$, representing the strength with which vertex i is associated to community k.

Under those definitions, we have, $\forall (i,k) \in V \times \mathcal{C}$:

$$n_{sim}(i,j) = 1 - \frac{\sum_{p \in V} a_{ip}(1 - a_{jp}) + \sum_{q \in V} a_{jq}(1 - a_{iq})}{\sum_{n \in V} a_{in} + \sum_{m \in V} a_{jm}} \tag{8.4}$$

$$\begin{aligned} s_{ik} &= \sum_{j \in V} a_{ij} \times c_{jk} \times n_{sim}(i,j) && \text{(8.5a)} \\ &= \sum_{j \in V} a_{ij} \times c_{jk} \times \left(1 - \frac{\sum_{p \in V} a_{ip}(1 - a_{jp}) + \sum_{q \in V} a_{jq}(1 - a_{iq})}{\sum_{n \in V} a_{in} + \sum_{m \in V} a_{jm}}\right) && \text{(8.5b)} \end{aligned}$$

$$c_{ik} = \begin{cases} 1 & \text{if } s_{ik} = \max_{c \in \mathcal{C}}(s_{ic}) \\ 0 & \text{otherwise} \end{cases} \tag{8.6}$$

With this model it is easy to iteratively compute, for each instant $t \in \mathbb{T}$, the assignment matrix $\mathbf{C}(t)$, based on $\mathbf{S}(t)$. The latter is computed using $\mathbf{A}(t)$ and $\mathbf{C}(t-1)$. As at network initialization, all nodes are considered being their own community, we have $\mathbf{C}(0) = [a_{ik}(0)] = [\delta(i,k)]$.

8.1.2 Extension to stable communities

Besides the constructive mechanism based on neighborhood similarity, the relative importance of links shall be introduced to modulate the contribution of each node, based on the stability and lifetime of relations with the other entities of the social structure, reflecting the passed history of the network.

As the algorithm is designed to operate in a heavily-constrained dynamic system, it is required that the value of links is estimated independently by each entity, with minimum node coordination, using local information and minimizing computational and memory requirements.

In order to consider the importance of links in the community assignment process, we propose to substitute the *neighborhood similarity measure* (defined in definition 8.1) by the *weighted similarity measure*, defined as follows.

Definition 8.2 (Weighted similarity measure)

The *weighted similarity measure* $n_{w,sim}$ is defined as, for any vertices $u, v \in V(\mathcal{N})$:

$$n_{w,sim}(u, v) = n_{sim}(u, v) * \hat{s}_u(v) \tag{8.7}$$

where $n_{sim}(u, v)$ is the *neighborhood similarity measure* as defined in definition 8.1 and $\hat{s}_u(v)$ is a *stability estimator* of the edge between u and v at vertex u.

The use of this estimator (instead of referring directly to the stability measure value of the edge) allows to keep the different *weighted similarity measures* consistent to what is the current stability state of a vertex neighborhood: a given absolute stability value s might, at different instants, be considered as corresponding to a really stable or really unstable edge, depending on the value of the other incoming edges.

We therefore propose the following computation formula for the *stability estimator*:

$$\hat{s}_u(v) = \frac{w_{vu}}{\sum_{n \in N(u)} w_{nu}} \tag{8.8}$$

where w_{vu} is the weight (reflecting the stability value) of the incoming edge from v to u (which is equal to w_{uv} if the graph is not directed). As w_{vu} is supposed to be a *positive* value, in the sense that the greater w_{vu} the better the considered link, w_{uv} can be safely set to 0 if vertices u and v are not adjacent. A direct consequence is that $\hat{s}_u(v)$ yields 0 for every pair of non-adjacent vertices (u, v).

Note that this value is not symmetric as $\hat{s}_u(v) \neq \hat{s}_v(u)$ in the general case. We therefore have a *node-centric stability estimation* at each vertex of the graph.

As our algorithm is implemented in real devices using beacon messages for the nodes to announce their identifier, current community label and adjacent vertices set, the weight value of each edge can easily be set using one of the stability metrics presented in subsubsection 2.2.2.4. For instance, the age of a link can be estimated by counting the number of consecutive beacons received from the corresponding neighbor node.

To account for some requirements in stability (e.g. the link shall live long enough to establish a communication), a *stability estimation threshold* can be set, under which value of all estimators $\hat{s}_u(v)$ are set to 0.

Finally, the community assignment rule presented in Equation 8.3 and the community membership strength computation formula given in Equation 9.1 can be rewritten as follows:

$$C(u) = \arg\max_{c \in \mathcal{C}} \sum_{\substack{v \in N(u) \\ C(v)=c}} n_{w,sim}(u, v) \tag{8.9}$$

$$m_{w,str}(u, C(u)) = \sum_{\substack{v \in N(u) \\ C(v)=C(u)}} n_{w,sim}(u, v) \tag{8.10}$$

This extension of the SHARC algorithm favors the creation of communities with high-weighted internal links. As these weights are deemed to reflect the stability of the associated links, we will refer to this extension as SAw-SHARC (Stability Aware SHARC).

8.2 Implementation

SHARC has been designed to be fast and computationally light and to operate without node synchronization, in order to be easily implemented on handheld devices forming wireless ad hoc networks. Also, SHARC relies on simple decentralized operations using local information only. To perform its assignment, each vertex only requires knowledge of the neighborhood of its adjacent vertices (referred as *two-hop neighborhood*) and their current community label.

In this section we will describe two possible implementations for SHARC: one used for benchmarking the algorithm (as presented in section 8.3) and the other suitable for deployment on real devices.

8.2.1 Implementation on real devices

When implemented on real devices, each wireless network node periodically updates its community before sending a beacon message containing the node identifier, the updated community label and the list of identifiers of its neighbors. Processing of all received beacons during a period allows to compute all relevant information for the next update. Typical period values are in the range of a couple of seconds.

This beacon can be sent as a one-hop broadcast applicative packet, or piggybacked to an existing beacon message (for instance if the devices use 802.11 in ad hoc mode). The payload of this packet is given for reference in Figure 8.1. The overhead generated by the SHARC protocol may be higher than for other distributed protocols, especially because beacon messages shall include the list of neighbor identifiers. However, networks exhibiting community properties are usually sparse graphs, which should limit this extra overhead.

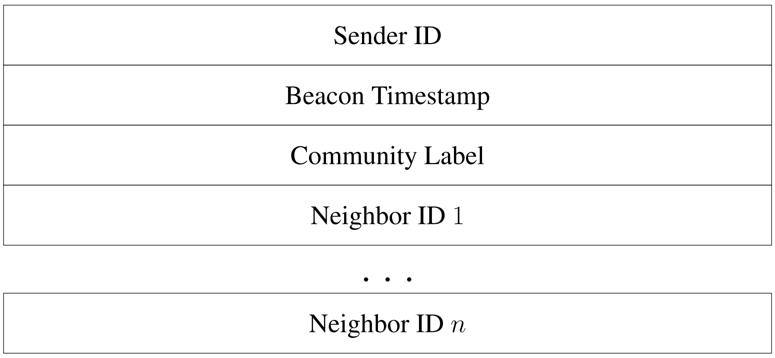


Figure 8.1: Example of beacon message to be used with SHARC

8.2.2 Implementation for simulation

For benchmarking on static or dynamic networks purposes, SHARC has been implemented as a simple single-threaded algorithm, where nodes iteratively proceed to their assignment in a random order until a termination condition is met. Access to neighborhood information was granted using the locally shared memory model of communication.

8.2.2.1 GraphStream

GraphStream[Gra] is a graph handling Java library that focuses on the dynamics aspects of graphs. It allows the modeling of dynamic interaction networks of various sizes and provides a way to handle the graph evolution in time by handling the way nodes and edges are added and removed, and the way data attributes may appear, evolve and disappear.

Therefore, the library defines in addition to graph structures the notion of stream of graph events that can be generated, read and forwarded by different components of the library (graph objects, algorithms, file readers and writers, etc.)

The GraphStream library is also perfectly suitable for the simulation of static graphs and proposes a large set of state-of-the-art algorithms in various domains, including random network generation and social network analysis.

We have chosen this library as, thanks to the notion of stream, it allowed an easy implementation of our algorithms and to test them on both static and dynamic networks with the same code. It also allowed to replay connectivity and mobility traces from finer-grain simulators, hence focusing only on the algorithmic part of our development, once the former were generated.

8.2.2.2 Implementation and class hierarchy

For the sake of performance evaluation of our proposed solution, as well as benchmarking with other comparable state-of-the-art community detection, we have implemented the following classes, composing the `org.graphstream.algorithm.community` package in GraphStream:

- `Community`: a class representing a community structure with all the required information (label, graphical representation, etc.);
- `DecentralizedCommunityAlgorithm` implements the `DynamicAlgorithm` (to react to graph changes) and the `Sink` (to listen to streams) interfaces of GraphStream, this base class for all distributed community detection algorithm provides the base loop that iterates over a shuffled list of the vertices of the graph;
- `AutonomyOrientedCommunityDetection` implements the autonomy-oriented algorithm for the community assignment problem presented in subsubsection 3.3.3.3;
- `EpidemicCommunityAlgorithm` introduces the core elements of epidemic label propagation algorithms, including community assignment update at each iteration based on information obtained from the neighbor nodes;
- `SyncEpidemicCommunityAlgorithm` is the synchronous version of the epidemic label propagation algorithm of Raghavan *et al.*[RAK07] presented in subsubsection 3.3.3.2;
- `Leung` implements the variation of the epidemic label propagation algorithm proposed by Leung *et al.*[LHLC09] and presented in subsubsection 3.3.3.2;

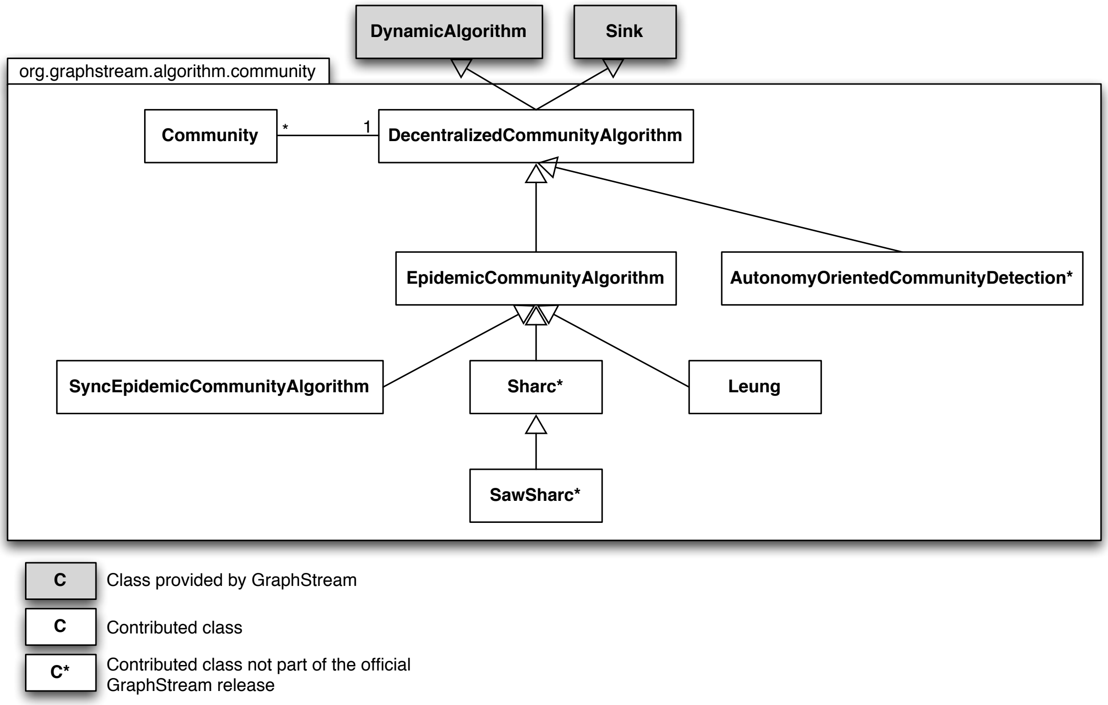

DynamicAlgorithm
Sink
org.graphstream.algorithm.community
Community
*
1
DecentralizedCommunityAlgorithm
EpidemicCommunityAlgorithm
AutonomyOrientedCommunityDetection*
SyncEpidemicCommunityAlgorithm
Sharc*
Leung
SawSharc*
C
Class provided by GraphStream
C
Contributed class
C*
Contributed class not part of the official GraphStream release

Figure 8.2: UML class diagram for the implementation of community detection algorithms in GraphStream

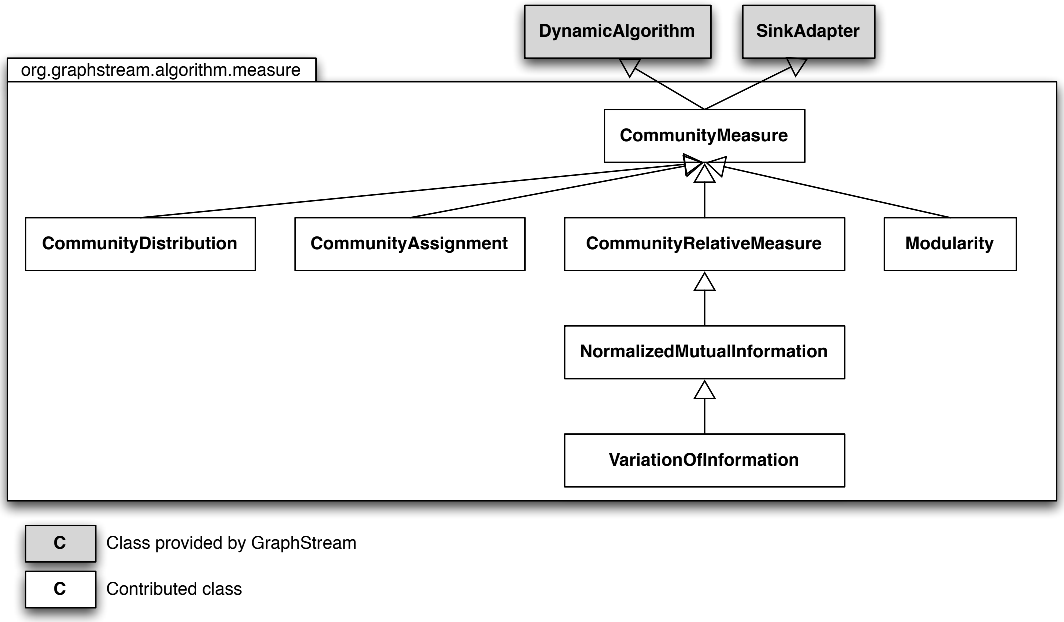


Figure 8.3: UML class diagram for the implementation of community detection metrics in GraphStream

- `Sharc` is the core implementation of the SHARC algorithm, using the neighborhood similarity and SHARC algorithms presented respectively in Algorithm 1 and Algorithm 2;

- `SawSharc` is the extension of the SHARC algorithm to account for weighted graphs, as explained in subsection 8.1.2.

Besides, we also completed the `org.graphstream.algorithm.measure` package with a set of community detection related metrics:

- `CommunityMeasure` is the base class for all community assignment metrics, providing easy access to the current assignment on the graph;
- `CommunityDistribution` provides statistics on the current assignment, including number of communities, minimum, maximum, average and standard deviation in the community sizes;
- `CommunityAssignment` allows an easy access to the different communities and the list of vertices that are currently assigned to them;
- `Modularity` is the implementation of the *modularity* measure we presented in subsubsection 3.3.4.1 in either the unweighted or weighted variants;
- `CommunityRelativeMeasure` is the base class for all metrics comparing the current assignment to reference assignments, which includes the `NormalizedMutualInformation` (see definition 3.17) and the `VariationOfInformation`.

8.3 Performance evaluation

After detailing the principles governing novel techniques for social structure mining in section 8.1, we will now present a thorough analysis of the performances of those solutions. We have chosen to consider both *classical* or *standard* graphs from sociological studies and random graphs for benchmarking.

In this section, we will first put our results in perspective with local and centralized algorithms, hardly applicable to the ad hoc network use case, in order to assess the gap with our decentralized approach. Then we will compare our contribution with other similar algorithms also relying on epidemic label propagation. All those algorithms were introduced in subsection 3.3.3.

8.3.1 Standard social network analysis graphs

Although they are not directly applicable to ad hoc networks, we want to compare the performance of our contribution with centralized and local algorithms used for community detection. This allows to put our work in perspective with more generic solutions recognized as valuable in the literature for this problem.

For this purpose, we will use the maximum modularity Q achieved on the *Zachary karate club*[Zac77] network. This metric and this network are the most used for comparison between algorithms. The corresponding results are shown in Table 8.1.

It appears that SHARC achieves results comparable to, or better than, other distributed algorithms. Besides, this algorithm outperforms early centralized community detection algorithms, like those introduced by Newman[NG04] and Donetti *et al.*[DM04].

Compared to local algorithms, i.e. greedy algorithms that mostly use local information to perform their community assignment, the performance gap lies between 1% and 33%, when compared to the most efficient algorithm. However, as explained in subsubsection 3.3.3.1, those

Algorithm	**Year**	**max. Q**	**Type**
Newman[NG04]	2004	0.381	Centralized
Donetti *et al.*[DM04]	2004	0.412	Centralized
Wang *et al.*[WCL07]	2007	0.4188	Local
Wan *et al.*[WCF08]	2008	0.488	Local
Tian *et al.*[TCF09]	2009	0.564	Local
Raghavan *et al.*[RAK07]	2007	0.4151	Distributed
Leung *et al.*[LHLC09]	2008	0.3718	Distributed
SHARC[HB10]	2009	0.4151	Distributed

Table 8.1: Comparison of maximum achieved modularity Q by various algorithms on the "karate" network

local algorithms assume, at some point of their execution, a global knowledge of the network. This assumption makes them incompatible for a use with networks where no global knowledge nor node coordination exist, like ad hoc networks. This result corroborates the general observation that decentralized algorithms, based on a given heuristic like SHARC, offer a sub-optimal solution while comparing with results from centralized algorithms.

Examples of community assignments achieved by SHARC on the "karate" network, but also on the "football" network (see [GN02]) are given in Figure 8.4 and Figure 8.5 respectively.

8.3.2 Random graphs

In this section, we will analyze the behavior of our contribution and compare it to other distributed algorithms on a set of randomly-generated graphs, with varying characteristics.

8.3.2.1 Girvan-Newman experiment

In this experiment, first presented in [NG04], we generated random networks composed of $N = 128$ nodes, pre-assigned in four different communities, each composed of 32 nodes. The average network degree is set to $\Delta = 16$. The node degrees follow a power law distribution of exponent $\tau_1 = -2$. Edges are created so that each vertex v of degree $d(v)$, has $(1-\mu_t)d(v)$ links with other nodes of its community and $\mu_t d(v)$ links outside its community. Sharpness of the assignment is assessed by increasing the value of the mixing factor μ_t until the community structures are no longer properly defined.

In some experiments, the mixing factor μ_t is replaced by the parameter z_{out}, which was originally used in [NG04], that represents the expected number of inter-community links for each node, that is $z_{out} = \mu_t \times \Delta$. This notation is not degree normalized and requires knowledge of the network density for proper interpretation.

In order to ensure the statistical confidence of our results, for each unique simulation configuration (unique graph instance, unique algorithm, unique set of configuration parameters), we run 200 simulations using different seeds for the initialization of the random number generator governing stochastic aspects of the algorithms or events management. We provide statistical estimators (average, standard deviation, minimum, maximum) for our results based on those different runs.

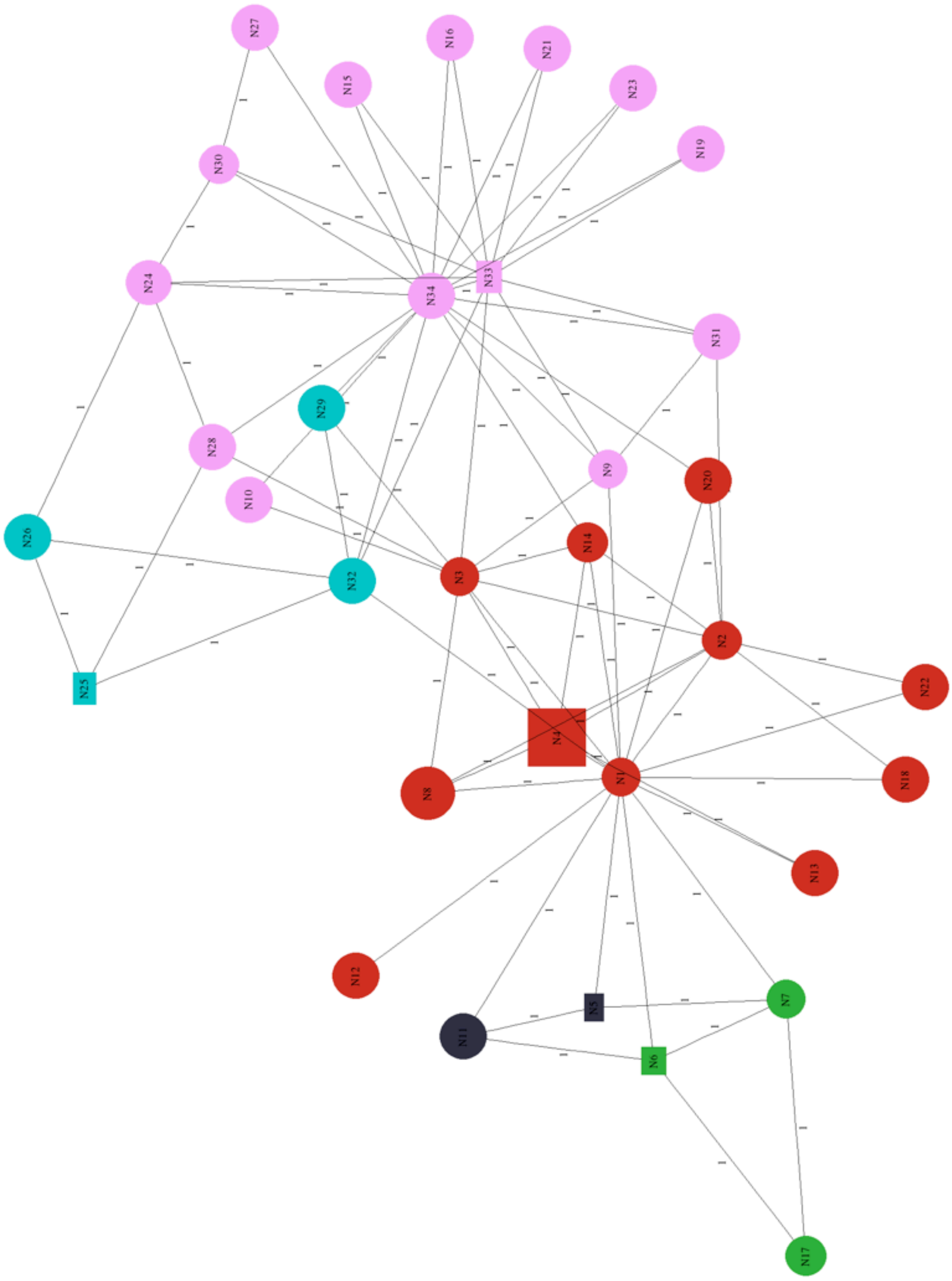


Figure 8.4: Example of SHARC community assignment on the "karate" network

This experiment will allow us to test the performances of the neighborhood similarity of the SHARC algorithm, introduced in subsection 8.1.1.

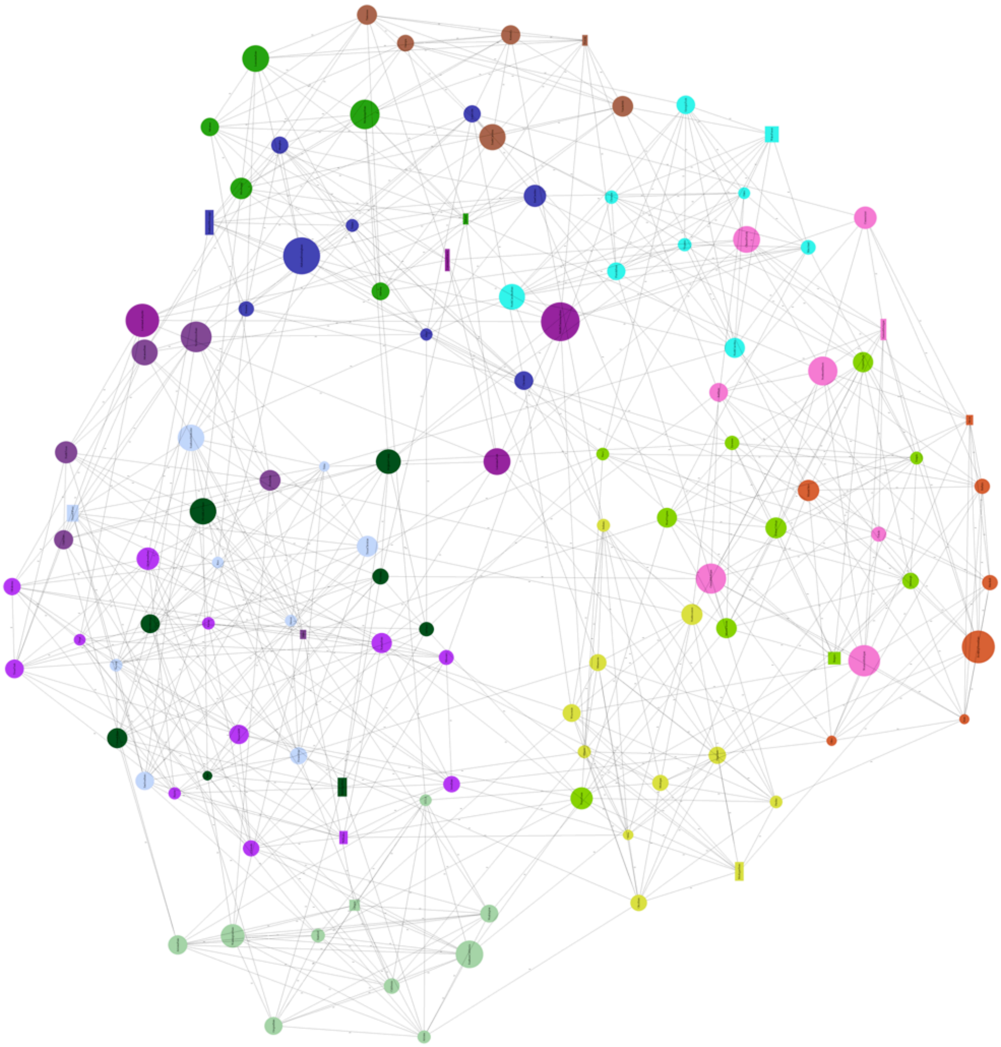


Figure 8.5: Example of SHARC community assignment on the "football" network

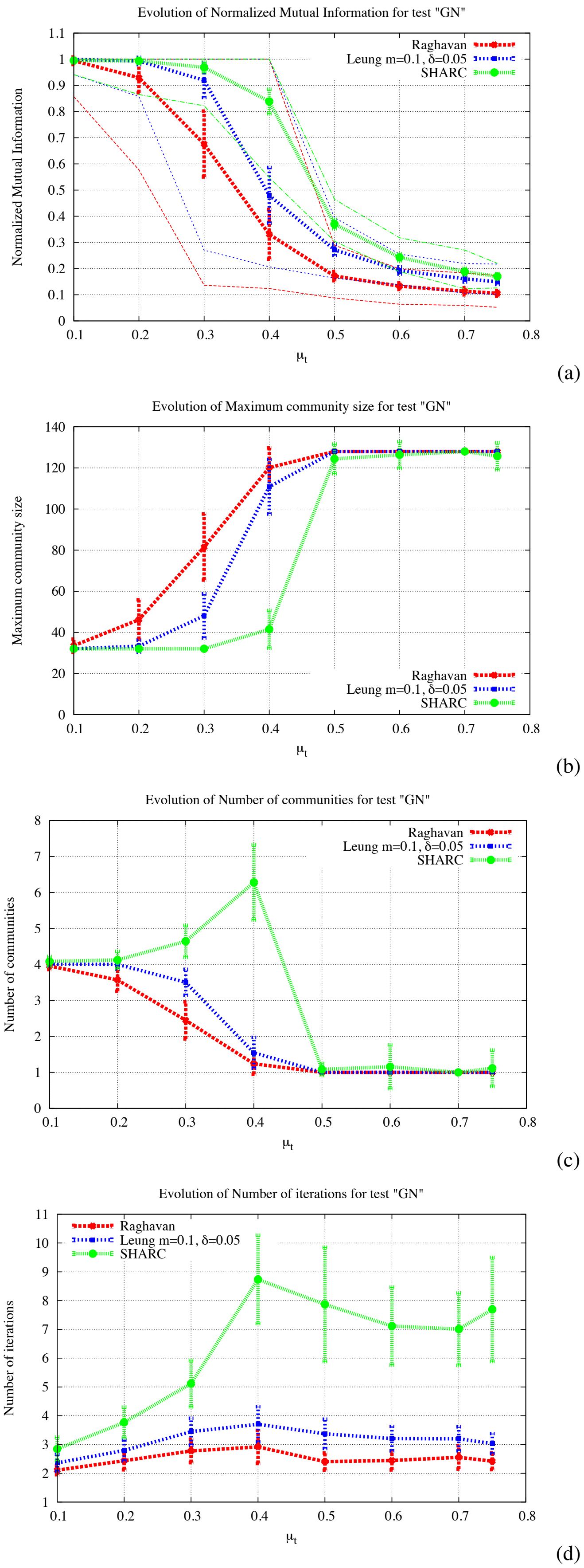


Figure 8.6: Evolution of NMI (a), Maximum community size (b), Number of communities (c) and Number of Iterations (d) values for the GN benchmark

The Figure 8.6(a) presents the average (with standard deviation as error bars) normalized mutual information (see Danon *et al.* in [DDDGA05]) achieved by the different distributed algorithms presented in subsubsection 3.3.3.2 for increasing values of μ_t. Dotted lines present the maximum and minimum values.

Those results show SHARC is able to maintain higher value for average NMI, even when the community structures become less clear cut ($\mu_t > 0.4$). The asynchronous epidemic label propagation algorithm[RAK07] performs the poorest and presents the widest variation in its results, making it unreliable for correctly detecting communities in any case. The algorithm presented by Leung *et al.*[LHLC09] performs in between the two other algorihtms.

When communities are not well-defined any more ($\mu_t > 0.7$), SHARC results are comparable with those of other algorithms.

A look at the evolution of the maximum community size, presented in Figure 8.6(b) shows that SHARC good performances are explained by its ability to longer bound the size of the communities to meaningful values. Thanks to the neighborhood similarity metric, the assignment performed by SHARC is sharper. Particularly when communities become fuzzy ($0.3 < \mu_t < 0.5$), while other algorithms generate too few communities, SHARC finds an alternative assignment, with more smaller communities (see Figure 8.6(c)), which yields to better results in terms of NMI.

It it also interesting to look at the spread of metrics on the different runs. It appears that SHARC presents less standard deviation in its results. In particular, it does not produce very low results in some configurations. SHARC is hence robust against particular network configurations and initialization parameters.

As shown in Figure 8.6(d), SHARC good performances come at the price of an increased number of runs before the assignment is stabilized, which was the termination condition in this experiment. However this increase is limited to a factor two or three. It would be interesting to analyze the performances of our contribution in dynamic network, when all algorithms will execute at the same pace.

8.3.2.2 Lancichinetti-Fortunato-Radicchi experiment

The LFR experiment, as presented by Lancichinetti, Fortunato and Radicchi in [LF09], can be seen as a generalization of the Girvan-Newman experiment presented above.

In this model, nodes degree still follows a power law distribution of exponent $\tau_1 = -2$ but the size of the communities also varies according to a power law distribution of exponent $\tau_2 = -1$. In our experiments, average node degree was set to $\Delta = 20$, maximum degree to 50.

This experiment uses a larger network size ($N = 1000$ nodes) and two different community configurations: S (*small* communities of size between 10 and 50 nodes) and B (*big* communities composed of between 20 and 100 nodes). The mixing parameter μ_t is still used to make the communities more or less clear-cut. This test can be considered as more demanding for the algorithms, due to the increased network size and the wider distribution in the community sizes.

As seen on Figure 8.7 and Figure 8.8, on both tests, SHARC maintains a higher NMI level whatever the current value of the community mixing factor μ_t. This means that SHARC is able to perform an assignment closest to expected assignment than all main other distributed algorithms. This is especially noticeable for very high values of μ_t and is justified by the ability of SHARC to maintain a larger number of size-bounded communities, even when they are not clear-cut.

In this demanding range ($\mu_t \geq 0.6$), SHARC minimum achieved NMI values are most of the time higher that the other distributed algorithms best results. Besides, the variance in SHARC results is very low, thanks to the effect of the neighborhood similarity metric. Those results are obtained with comparable number of iterations.

Results on both Girvan-Newman and LFR benchmarks tend to prove that SHARC is well suited for sharp and robust community detection, even in large networks and when the communities have no pre-defined size or when they are not clear-cut.

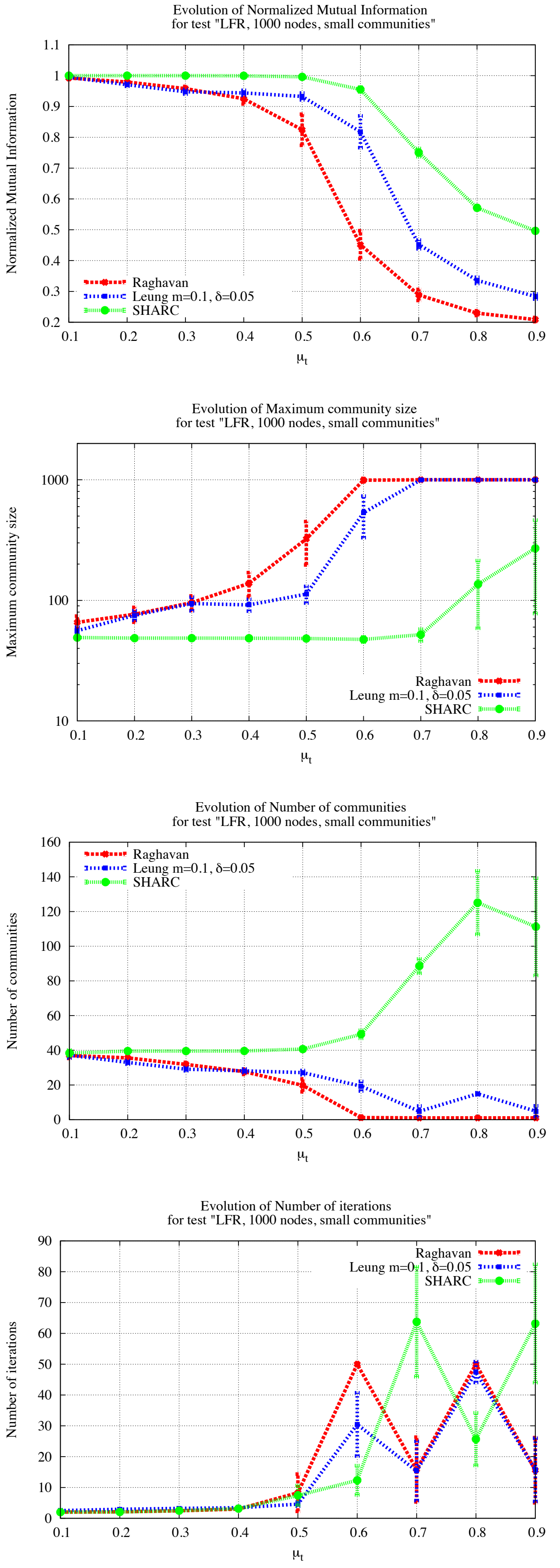
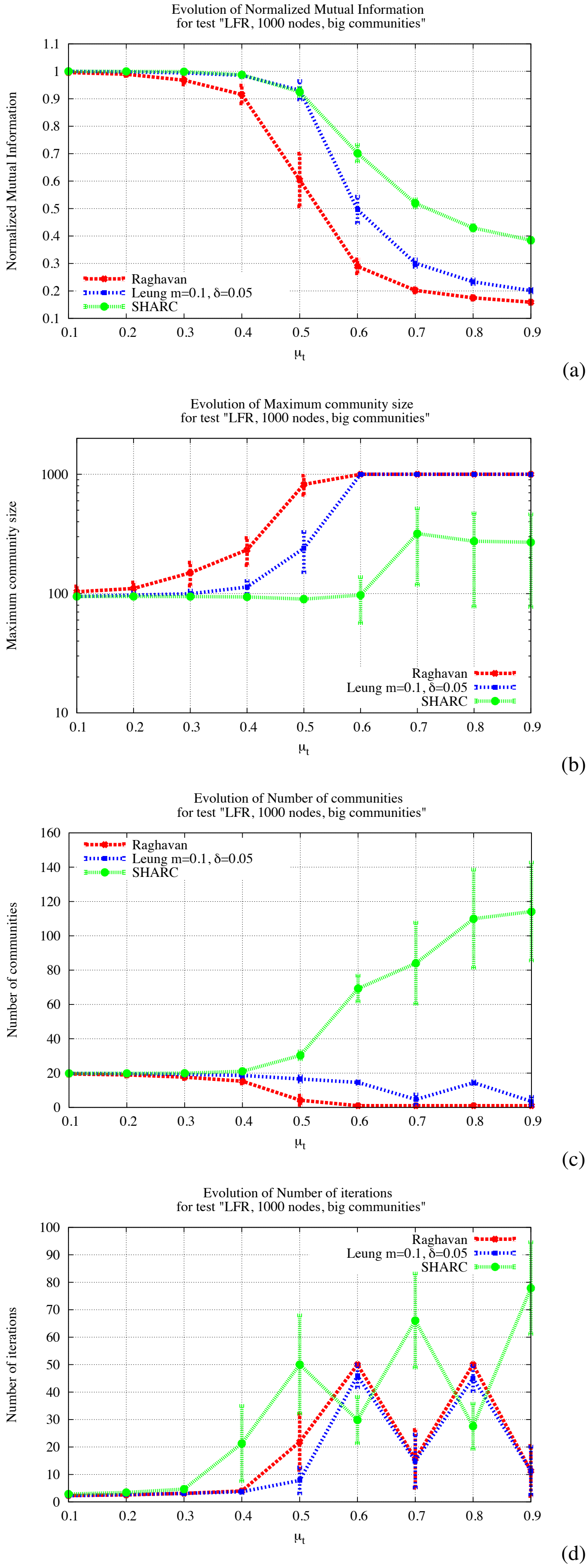


Figure 8.7: Evolution of NMI (a), Maximum community size (b), Number of communities (c) and Number of Iterations (d) values for the LFR benchmark, 1000 nodes, small communities



(a)

(b)

(c)

(d)

Figure 8.8: Evolution of NMI (a), Maximum community size (b), Number of communities (c) and Number of Iterations (d) values for the LFR benchmark, 1000 nodes, big communities

8.3.2.3 Weighted LFR experiment

In order to test algorithms that integrate link-reliability aspects in the community assignment process, we will rely on the weighted version of the Lancichinetti-Fortunato-Radicchi (LFR) benchmark[LF09]. We will assume that the weight assigned to each link reflects its stability level, as explained in subsection 8.1.2.

In this test, networks of $N = 1000$ nodes are generated with the tool provided by the LFR benchmark authors[LF09]. The nodes degree distribution follows a power law of exponent $\tau_1 = -2$ and sizes of the communities also vary according to a power law distribution of exponent $\tau_2 = -1$. In our experiments, average node degree was set to $\Delta = 20$, maximum degree to 50.

We also used two different community configurations: S (*small* communities of size between 10 and 50 nodes) and B (*big* communities composed of between 20 and 100 nodes). The links weights were distributed with a power law of exponent $\beta = 1.5$.

Nodes are also characterized by the *mixing parameter*, μ_t, which defines the ratio between links they have outside and inside their community. We also used the weighted counterpart of the mixing parameter, μ_w, comparing the total weight of links outside their community to those inside the pre-defined community. We set μ_t to 0.5 and made μ_w vary between 0.1 and 0.8 to assess the drop of performance when inter-community links become stronger.

For each unique parameter configuration, we have generated 10 different networks, and ran each algorithm 20 times on each network, leading to a total of 200 unique simulations for each algorithm.

In Figure 8.9 and Figure 8.10, we present results for the small and large communities configurations. For assessment, we used one topological-only metric: the *Normalized Mutual Information*[DDDGA05] and one metric considering the stability of the generated structures: the *structure stability*[Pig08]. We provide the average, standard deviation, minimum and maximum achieved values by each algorithm.

On both configurations (small and large communities), we observe that SHARC performs well in terms of normalized mutual information, showing its ability to correctly retrieve the *reference* assignment that was used to distribute link weights on the graph, whether or not the distribution is clear-cut amongst communities. Note that the epidemic algorithm of Raghavan *et al.*, as it is unweighted, performs equally poorly whatever the μ_w value. The algorithm of Leung *et al.*, while efficient for low values of μ_w, performs the worst for high values of the weighted mixing parameter. This underlines the limitation of parameter-based approaches that may not yield good results in different graph configurations.

By considering the stability of the generated communities, it appears that SHARC is able to construct the most reliable structures (i.e. grouping the most-weighted edges *inside* communities). Besides, our algorithm does not suffer any performance drop for intermediate values of μ_w, showing its ability to automatically adapt to changing network conditions.

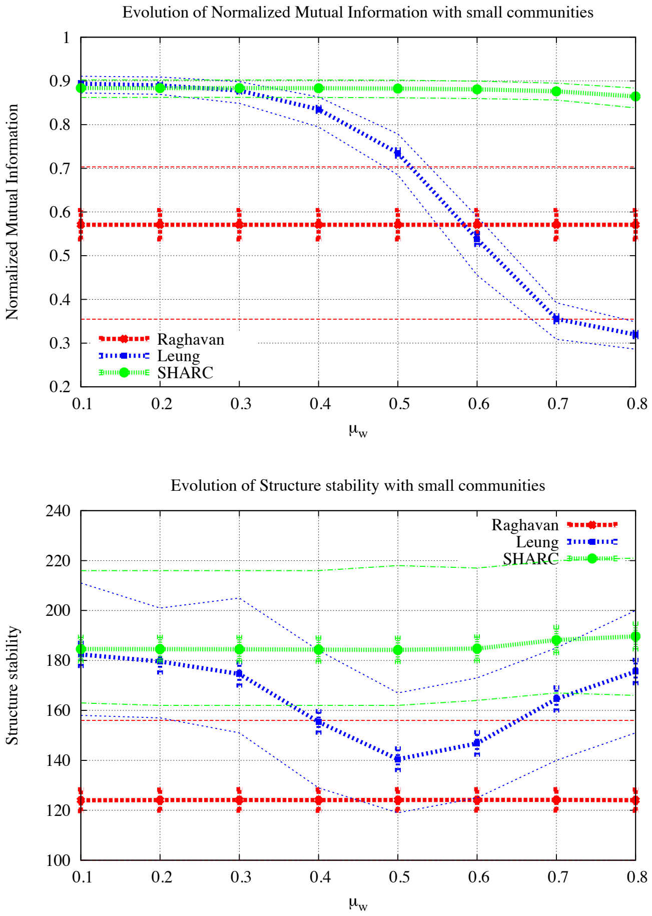


Figure 8.9: Comparison of distributed community detection algorithms on weighed LFR with small communities

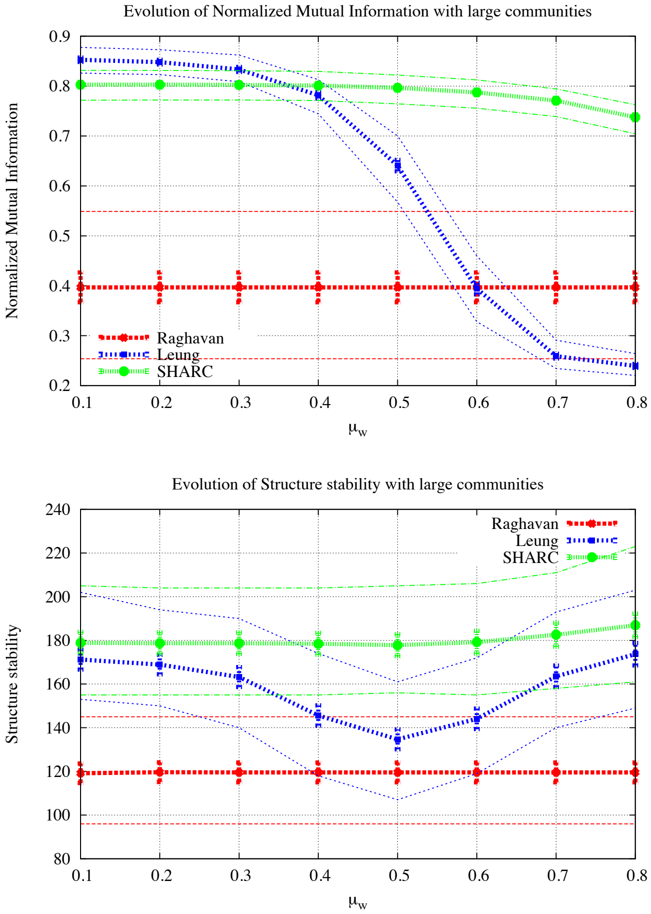


Figure 8.10: Comparison of distributed community detection algorithms on weighed LFR with large communities

8.4 Highlight

In this chapter, we have:

- Introduced a **new distributed algorithm for community detection**, **SHARC**, in order to partition networks where no global knowledge nor node coordination exist;
- **Addressed the current shortcomings** of the existing distributed algorithms used with this kind of networks;
- Analyzed that the introduction of the **neighborhood similarity** measure makes the assignment realized by SHARC **sharper and more robust** to initial condition changes;
- Shown that SHARC **prevents the community domination effect** by maintaining a meaningful number of size-bounded communities, even when they are not clear-cut;
- Presented and tested **SAw-SHARC**, an extension of SHARC that creates **communities with highly-weighted, or stable, internal links**.

In the next chapter, we will see why and how our contribution can be extended to successfully operate on dynamic networks. After testing the relevance of the proposed solution, we will investigate how it can correlate with the social incentives of urban mobility.

Contents

Chapter 9

Distributed community detection on dynamic graphs

> "A virtual community is a group of people who may or may not meet one another face to face [...] and who exchange words and ideas through the mediation of computer bulletin boards and networks."
>
> Howard Rheingold, The Virtual Community.

In this chapter, we will present SAND/SHARC (Stability And Network Dynamics over a Sharper Heuristic for Assignment of Robust Communities). This algorithm addresses the problem of distributed detection of community structures in networks by using the epidemic label propagation paradigm. But unlike previous contributions found in the literature, our algorithm is specifically designed to operate on dynamic graphs, where vertices and edges evolve over time. SAND/SHARC not only constantly adapts its solution to changes in the graph topology but also converges to communities that exhibit interesting properties in a dynamic environment, such as robustness and stability of intra-cluster links. Meanwhile, it provides an orientation of the community vertices towards a barycenter.

Finally, we will analyze to what extend the communities that are dynamically determined by our algorithm, based on network topology only, have a social signification. This fact is closely related to the underlying social motivations in user mobility.

9.1 Community detection on dynamic graphs

In this section, we will present the novel features composing SAND/SHARC (Stability And Network Dynamics over SHARC), and introduced in [HB11a]. This algorithm aims at mining communities in dynamic networks. It is built upon SHARC by using its epidemic label propagation constructive mechanism that favors the generation of robust communities.

9.1.1 Observations

As early experiments, we run the same algorithms as in section 8.3 in dynamic network scenarios (the latter will be presented in further details in subsubsection 9.1.3.2).

On those experiments, after the first iterations we experienced a high decrease in the modularity achieved by the algorithms. This plummet is due to apparition of *monster* communities as the simulation evolves: those algorithms tend to generate communities whose size is not properly bounded and cover an important part of the network.

We also observed that all algorithms suffered from what we denote as the *wandering community effect* (illustrated in Figure 9.1): if a set of vertices of the same community split into two disconnected subsets, they may keep being assigned to the same community by the algorithms and propagate this label to other communities, making the label diversity plummet. If it does not impact any application based on this assignment, as the subsets are not connected, this yields poorer results in terms of modularity Q or NMI as the simulation evolves.

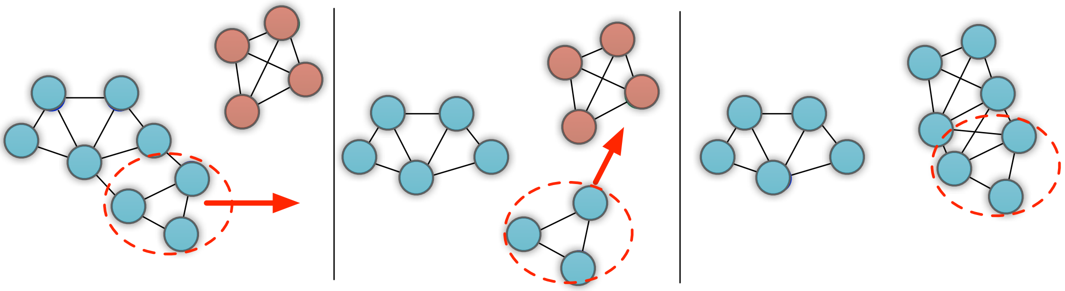


Figure 9.1: Graphical illustration of the wandering community effect

Despite this effect, SHARC is the algorithm that maintains the highest average and maximum modularity Q, as it limits as much as possible, but does not prevent, the *monster* community effect. However, it appears that dedicated measures are required to maintain efficient community detection over dynamic networks.

9.1.2 Contribution

To dynamically construct meaningful community structures and continuously adapt them to account for the evolution of a dynamic network, it required to consider information about the network history and topology. This requires an improved constructive mechanism (to prevent the monster community effect and improve robustness of the structure), more efficient organization of nodes inside the community, but also a dedicated process to cope with the changes in topology and characteristics of the network.

9.1.2.1 Community organization mechanism

Determining a coherently placed *center* of the social structure is important as this node can then serve as an efficient bridge from the ad hoc community to an infrastructure network, collecting, merging and reporting information.

A good candidate for playing this role is the community *center* in the sense of *proximity centrality*. This *center* is characterized by having the lowest average shortest path to all the other nodes of the community. However, computing this metric is hard to achieve in a dynamic and distributed environment.

In epidemic label propagation algorithms, nodes join a community by adopting the label shared by one of their adjacent vertices. We can hence introduce the notion of *originator* as the node that propagated its original label to all the nodes of its current community. However, the position of the originator is fixed and mainly depends on the order in which nodes update their community assignment. Therefore, originator and center position will likely not match, especially when the structure evolves over time.

A possible solution is to extend the notion of originator by dynamically updating its position so that it is always placed far from the community boundaries, in areas where the density and reliability of the connections between vertices are high.

Based, on the community assignment rule used in SHARC and presented in Equation 8.3, we have presented the *community membership strength*, which reflects, for any vertex $u \in \mathcal{N}$ assigned to community $C(u)$, the total count achieved by this community:

Definition 9.1 (Community Membership Strength)
The *community membership strength* of a given vertex $u \in V(\mathcal{N})$ is defined as the total of the community count, as computed in Equation 9.1:

$$m_{str}(u, C(u)) = \sum_{\substack{v \in N(u) \\ C(v)=C(u)}} n_{sim}(u, v) \tag{9.1}$$

The value of the *community membership strength* reflects the strength with which the considered vertex u is tied to the community $C(u)$. Simple graph considerations can show that nodes placed *inside* their community (i.e. having no edge to the outside of their current community) will tend to have a higher score than nodes placed at the border of their community. Passing the *originator* role to nodes with higher scores is hence a first step to place it closer to the community *center*.

The *community membership strength* is computed using local information (the information scope is limited to the two-hop neighborhood), hence nodes have no knowledge whether the chosen *originator* is a local or a global strength optimum over the community. It is therefore very likely that the procedure described above will converge to a node with a *community membership strength* locally optimal.

In order to address this effect, it is required to add diversity to the path followed by the *originator* role but without passing it to weakly assigned nodes. We therefore implemented a *local-optimum-favored weighted random walk* mechanism.

This algorithm described in Algorithm 3 performs as follows. If a node is a local *community* *membership strength* optimum, then it only passes the *originator* role with a probability equal to the ratio of the highest score found in the adjacent nodes over the score of the current node: this is the local optimum favor phase. If the current is not a local *community membership strength* optimum, or if the local optimum has chosen to pass the role, then a weighted random walk is applied: adjacent nodes with a higher score are more likely to be selected to become *originator* (as detailed in Algorithm 3). Note that if a node leaves its current community, it automatically loses its originator status.

Algorithm 3 Local-optimum-favored Weighted Random Walk

Data: u: vertex for which the neighborhood similarity is computed
Data: $C(u)$: current community of node u
Data: $m_{str}(u, C(u))$: membership strength of u in its current community
Data: $N(u)$: one-hop neighborhood of node u
Data: current_originator: vertex currently assuming the originator role
Data: $\mathcal{M}_{str}(u) = \{s | s = m_{str}(v, C(v)), \forall v \in N(u)\}$
Output: current_originator

```
if current_originator = $u$ then
    current_originator ← $u$
    r ← rand(0,1)
    if max$\mathcal{M}_{str}(u) \leq m_{str}(u, C(u))$ or r < $\frac{\max \mathcal{M}_{str}(u)}{m_{str}(u, C(u))}$ then
        r ← rand(0,$\sum_{v \in N(u)} m_{str}(v, C(v))$)
        for all $v \in N(u)$ do
            if r $\leq m_{str}(v, C(v))$ then
                current_originator ← $v$
            else
                r ← r - $m_{str}(v, C(v))$
            end if
        end for
    end if
end if
return current_originator
```

9.1.2.2 Community evolution and split mechanism

Epidemic label propagation is a simple and efficient way to disseminate a community label to vertices belonging to highly interconnected neighborhoods. However, due to the dynamics of the network, link density may decrease in some areas of the community, requiring the structure to split into different entities to maintain a meaningful assignment.

In first attempts to address the community detection problem, we faced the *wandering community effect* (see subsection 9.1.1): a community that had split in two or more disconnected components could propagate the same community label into different, unrelated places of the network. This effect proved to dramatically diminish the community label diversity, leading to poor results in dynamic simulations.

To counter this effect, we propose the implementation of a community *break mode*. Each originator includes an increasing *freshness counter* value, or a timestamp. Non-originator nodes simply propagate the highest received value for this counter in their next beacon. Whenever there is a split in a community, all the nodes not any more connected to the originator of their community will experience a stalling in their freshness and reassign themselves to a different community label. The global behavior of the break is summarized in Figure 9.2.

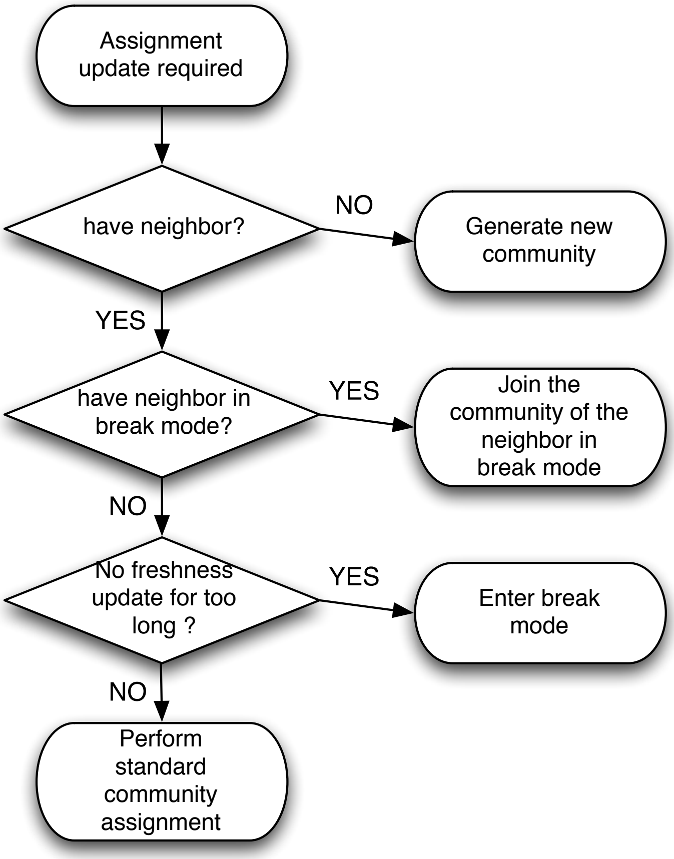

Assignment update required
have neighbor?
NO
Generate new community
YES
have neighbor in break mode?
YES
Join the community of the neighbor in break mode
NO
No freshness update for too long ?
YES
Enter break mode
NO
Perform standard community assignment

Figure 9.2: Community assignment using break mode

This *freshness propagation* mechanism allows to detect the split of a community in disconnected components. However, the connectivity of a dynamic network could evolve so that all the nodes with same label still are in the same connected component but that the link density distribution would require a subdivision in different communities for the assignment to be meaningful. One can envision this case in the example of a conference where, after a general session, people would move to attend separated tracks, still maintaining some connectivity between the different rooms.

We therefore complement our approach by using a *constrained propagation of the freshness counter*. Due to network dynamics, some community members or intra-community links may disappear, creating a zone of density drop. This area will act as border between the subsets emerging from the current community. Nodes around the density drop will experience an increased variance in the distribution of *neighborhood similarity* amongst their adjacent nodes.

To detect this increase in the distribution spread, each node keeps an up-to-date distribution of the computed neighborhood similarity metrics, particularly their average value and standard deviation. Adjacent vertices with a significantly low similarity will be ignored in the freshness counter update process. We place the threshold to the average value minus the standard deviation.

As a consequence, low link density barriers that may appear within a given community due to network dynamics are considered as *logical splits* of the community and will also trigger the regeneration of several community identifiers, using the *break mode*.

The required information for the break mode and constrained propagation of the freshness counter to operate is given in Figure 9.3.

Sender ID
Freshness counter or timestamp
Distance to originator
Community label
Break community label (or unset if not *break mode*)
Neighbor ID 1
. . .
Neighbor ID n

Figure 9.3: Example of beacon message to be used with SAND/SHARC

9.1.2.3 Implementation

In order to implement the procedures presented above in our simulation framework, the following classes have been added:

- `DynSharc` which implements the *break mode* based on constrained propagation of the freshness counter;
- `SandSharc` which introduces the local-optimum favored weighted random walk in order to dynamically improve the position of each community *originator*.

The complete class diagram of the algorithms derived from our original SHARC algorithm is given in Figure 9.4.

9.1.3 Performance evaluation

Although designed with a consideration of the stability of the generated communities, SAND/SHARC main objective is to operate on dynamic networks, such as the one created by owners of handheld devices that will form and communicate in a mobile social network, or a vehicular ad hoc network.

In this section, we will present the set of experimentations we have performed in order to assess SAND/SHARC performances in close-to-reality dynamic scenarios. We want to evaluate improvements related to the introduction of the break mode to deal with highly dynamic networks.

We will first describe our experimentation setup before presenting and analyzing results from the exhaustive simulation campaign we have performed.

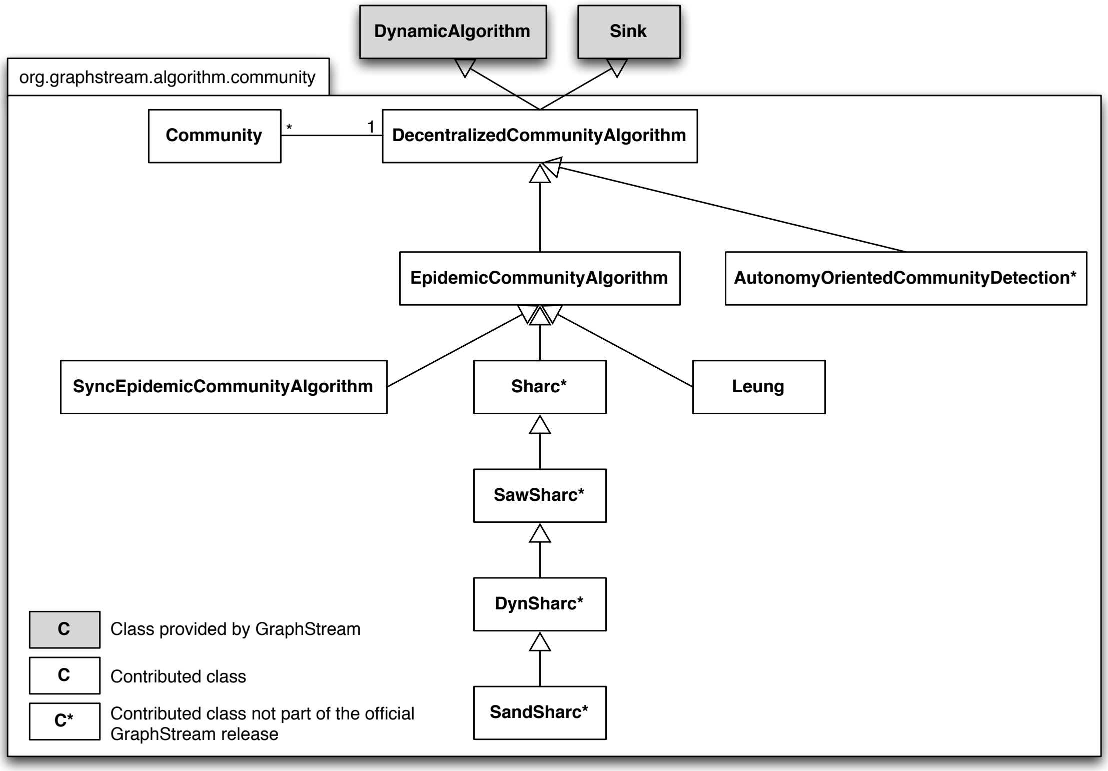

DynamicAlgorithm
Sink
org.graphstream.algorithm.community
Community
*
1
DecentralizedCommunityAlgorithm
EpidemicCommunityAlgorithm
AutonomyOrientedCommunityDetection*
SyncEpidemicCommunityAlgorithm
Sharc*
Leung
SawSharc*
DynSharc*
SandSharc*
C
Class provided by GraphStream
C
Contributed class
C*
Contributed class not part of the official GraphStream release

Figure 9.4: UML class diagram for the implementation of community detection algorithms in GraphStream

9.1.3.1 Dynamic scenarios description

We propose to evaluate SAND/SHARC, along with other epidemic label propagation algorithms on three human and nature-inspired dynamic scenarios.

The first two scenarios reflect the mobility of human users in two urban contexts: a *shopping mall* and a *highway section*. Both derive from the Human Mobility Model, that we presented in subsection 7.3.2. Their particular characteristics are summed up in Table 9.1.

In the *shopping mall* scenario, spots stand for shops, uniformly distributed over the mall area where users can interact by exchanging instantaneous advice about the stores they visit. This scenario generates highly dense wide areas and allows to study the behavior of protocols in dense network conditions. An example of this scenario is depicted in Figure 9.5, where vertices colors represent the current community attributed by SHARC, and their size the *membership strength* defined in subsubsection 8.1.1.3.

The *highway section* environment is characterized by nodes moving at a very high speed between a few number of spots (2 in our tests). Here, users in cars are interested in information about the road status and dynamically diffuse it using car-to-car communications. We have chosen this particular scenario as it produces topologies where most links are organized into chains of nodes moving in opposite directions, which favors domination of a single community. As nodes move very fast, link duration is also very short. This is therefore a very aggressive scenario to test the robustness of the algorithms. Due to the higher mobility speed and shorter link lifetime, the *highway section* scenario appears as much more demanding, and will allow to assess the

Parameter	**Mall**	**Highway**
Nodes	99	80
Area	300 × 300 m	750 × 900 m
Duration	120 s	120 s
Algorithms iterations	480	480
Iterations per second	4	4
Mobility speed	[1..10] km/h	[70..140] km/h
Avg. transmission range	43 m	46 m
Avg. node degree	8	10
Avg. link lifetime	48 s	3.4 s
Avg. link density	5.10^{-3} m^2	6.10^{-4} m^2

Table 9.1: Parameters of the "mall" and "highway" scenarios

efficiency of the break mode.

For both algorithms, the connectivity traces were obtained by using the Metropolitan Ad Hoc (Madhoc) network simulator, presented in subsection 6.3.3, allowing to take into account the impact of the spots on the physical propagation of radio waves.

As a different perspective, we also tested the performance of the algorithms on a nature-inspired scenario. We relied on the concept of *boids*, used by Reynolds in [Rey87], which are mathematical entities modeling the movements of birds. A large number of individuals of different species (we instantiated 1000 boids) moves in a sphere-shaped area. Based on whether they belong to the same species or to a predating specie, *boids* will attract themselves on tend to fly away in order to escape danger. A snapshot of the *boids* scenario is given in Figure 9.6. In this scenario, connectivity was modeled using the free-space radio propagation model.

For each scenario, we have generated four different dynamic networks with similar properties, based on the mobility and connectivity traces. For each network, we performed 25 runs in our simulation framework based on GraphStream, deriving from the algorithms implementation made for static networks. We used different initialization seed for each run, leading to a statistical analysis over 100 unique simulations per scenario.

9.1.3.2 Results and analysis

Results for those three scenarios described above are given in Figure 9.7, Figure 9.8 and Figure 9.9 respectively. While considering the topological quality of the performed assignment (using modularity results presented on the topmost graphs), it appears that SAND/SHARC achieves the best results. The more demanding the scenario, the bigger the performance gap in favor of our algorithm (this is especially true in the *highway scenario* which involves vehicular mobility and very sporadic connections between cars going in opposite directions).

The good performances of SAND/SHARC are explained by looking at the number of communities generated by the algorithms during the different scenarios (plotted in the middle graphs of Fig. 9.7, 9.8 and 9.9). The constrained freshness propagation regeneration mechanism of SAND/SHARC is able to detect local drops of link density, initiate the construction of new structures within a community and hence maintain the number of communities to a meaningful levels. Other algorithms still suffer from the *wandering community* effect that causes quick

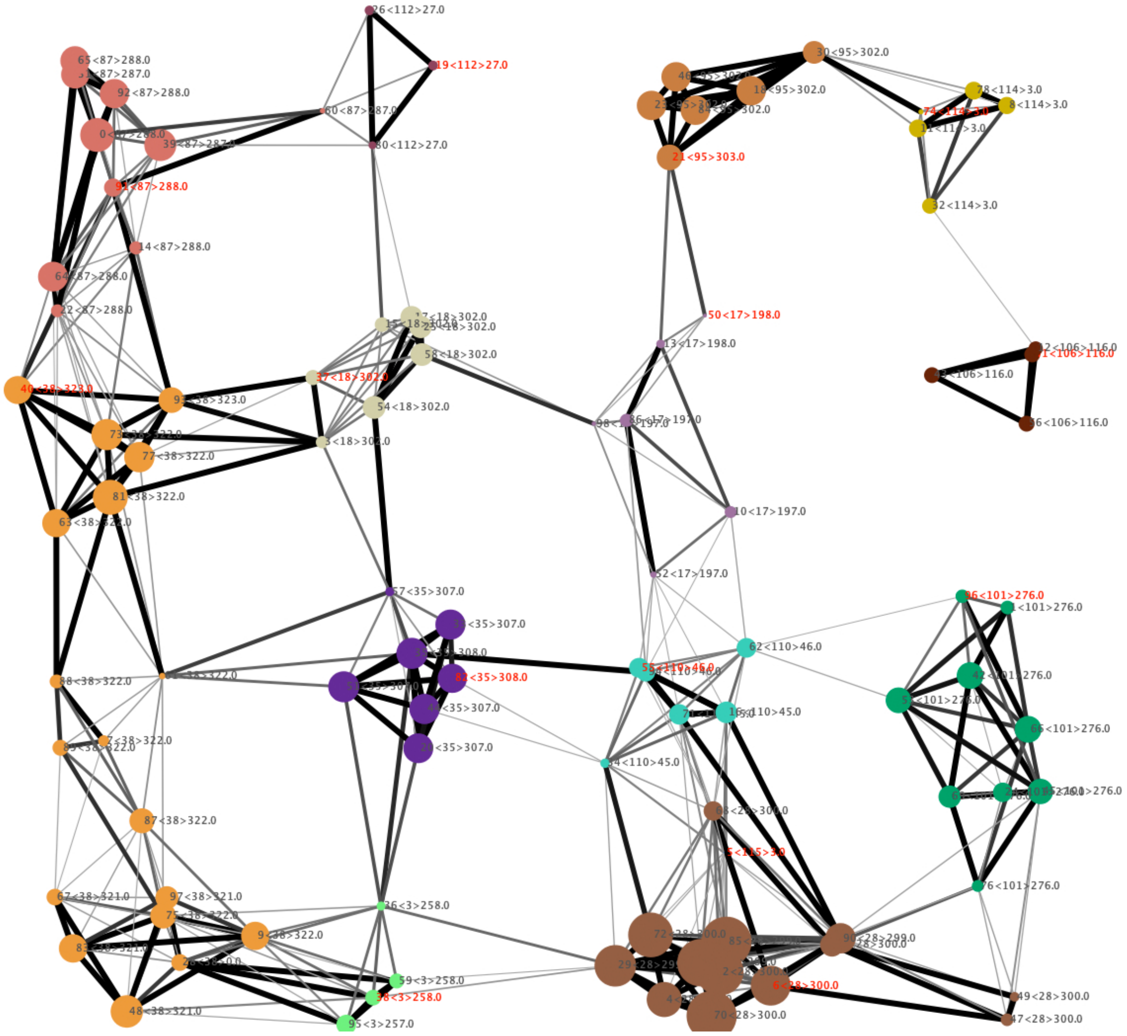


Figure 9.5: Graphical illustration of the assignment operated by SAND/SHARC on the mall scenario

drops of community labels diversity.

Besides, the communities generated by SAND/SHARC are also the most reliable. In those experiments, we used the age of the communication links as stability criterion. As shown on the bottom graphs of Fig. 9.7, 9.8 and 9.9, the communities constructed by SAND/SHARC are so that the average of inter-structure edges ages is the highest: our algorithm really favors the construction of structures where vertices are densely and reliably connected. On the contrary, inter-community edges are most of the time short-lived.

The characteristics of this assignment are illustrated in Figure 9.5. The size and darkness of edges represent their stability value, while the color and size of vertices depict respectively their current community and assignment strength. SAND/SHARC tends to group nodes that are connected with reliable links and leaves less stable connection as inter-community bridges. This snapshot of the SAND/SHARC process also shows the current position of the community originator (whose labels are in red). During the simulation, its position dynamically evolves using the algorithm detailed in subsubsection 9.1.2.1 and the algorithm ensures that a unique originator node is kept per community.

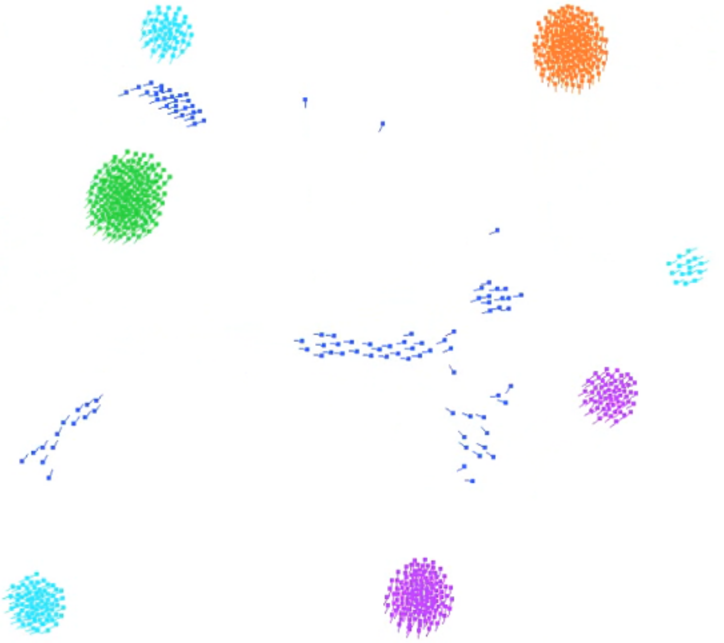


Figure 9.6: Graphical illustration of the *boids* scenario: colors show boids species (with dark-blue being predators), not community assignment

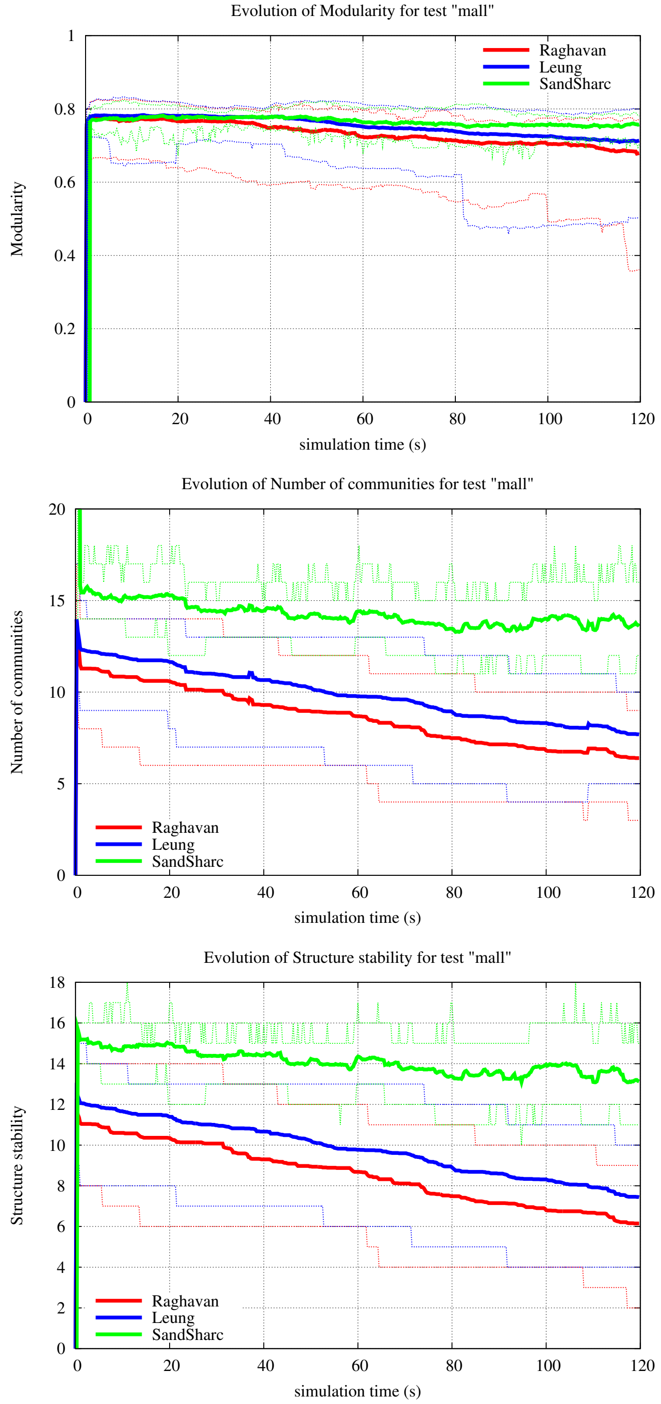


Figure 9.7: Comparison of SAND/SHARC with other distributed community detection algorithms on mall scenario

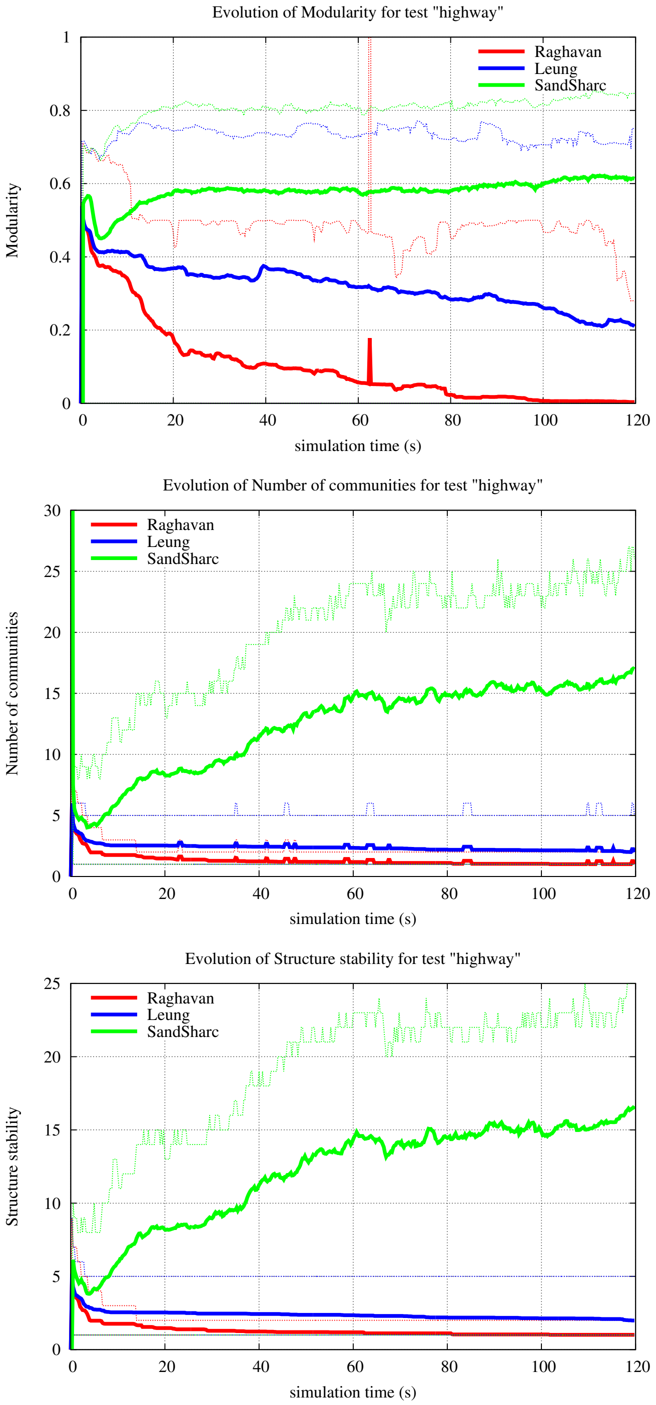


Figure 9.8: Comparison of SAND/SHARC with other distributed community detection algorithms on highway scenario

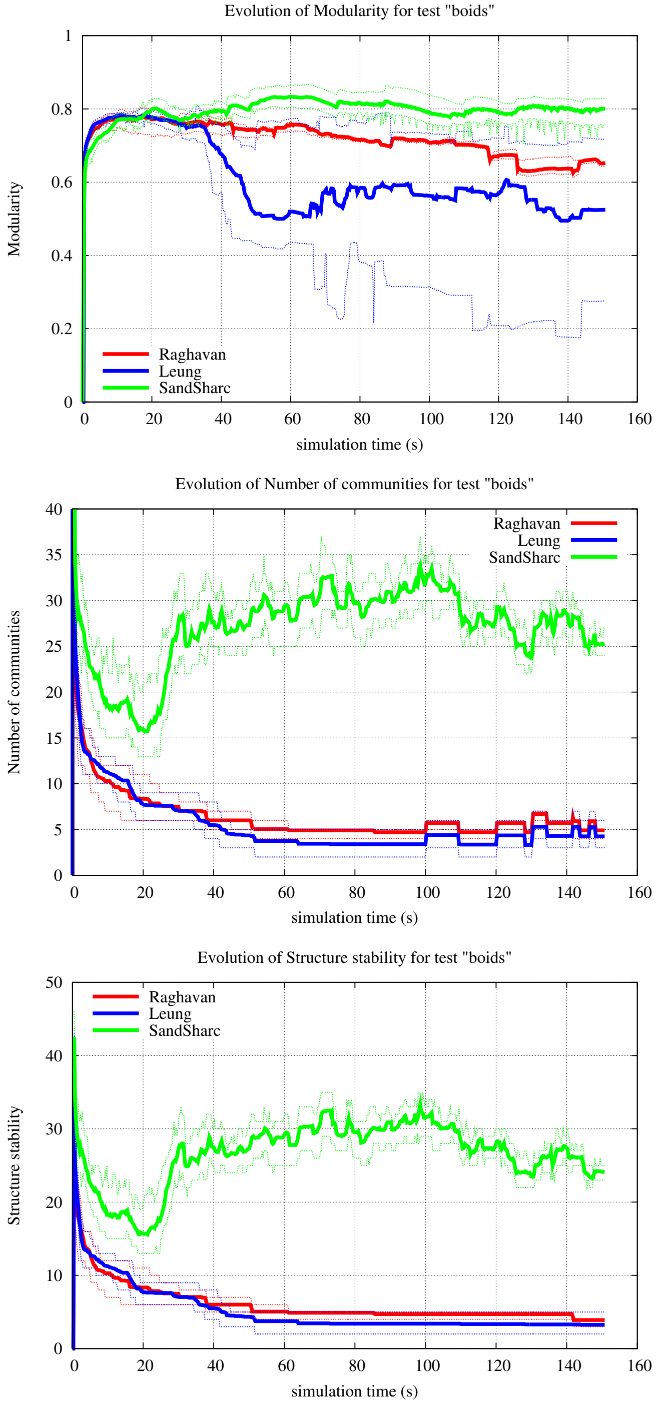


Figure 9.9: Comparison of SAND/SHARC with other distributed community detection algorithms on boids scenario

9.2 Social relevance of detected communities

In previous section, we have presented SAND/SHARC. Based on a simple yet novel heuristic for distributed community detection, and using the network topology only (i.e. the connectivity between all users) it allows to dynamically and individually group each entity of a communication network in the relevant social structure.

In this section, we will show how, based on the fact that in real-life scenarios users with social similarities and shared interest are more likely to get connected, it naturally groups users most likely to interact in the same communication entity.

In order to evaluate the social relevance of the communities dynamically determined by our algorithm, we will replay a set of real world traces, run our community detection on the dynamic communication network generated from the traces and analyze the correlation with social information about users.

9.2.1 Experimental setup

For this experiment (see also [HB11c]), we will rely on the connectivity traces collected by Chaintreau *et al.*[CHC$^+$07] during an international conference. It contains the connectivity traces of 70 participants carrying a Bluetooth device (called "iMote") over four days of the conference. This set is particularly interesting, not only for its precise and extended connectivity records, but also as each participant filled a questionnaire with a number of social information.

We converted the traces to a dynamic graph in GraphStream[DGOP07], on which we performed dynamic community detection using SAND/SHARC. We then look for correlation between the assignments dynamically generated and updated by our algorithm and social similarities (nationality, shared interest, co-location) between users, based on their answers to the related questionnaire.

9.2.2 Results and analysis

We will present our analysis of the social relevance of dynamically determined communities, based on the interaction network topology as follows. We have counted the fraction of the experiment time that each pair of users have spent in the same community and compared that against the number of common social aspects they share. We have decided to group the possible social similarities in different groups in order to assess their impact.

9.2.2.1 Nationality and language

An important factor that could lead users to have extended interactions is, of course their nationality and spoken languages. This would allow us not only to easily exchange ideas, without any cultural barrier, but also to have common subjects (national politics, news, sports, etc.) to talk about. The results about shared country, city, nationality, and spoken languages are presented in Table 9.2, Table 9.3, Table 9.4 and Table 9.5 respectively.

Similarity	Average	Std. dev.	Min.	Max.	Users
0	0.0907	0.0578	0	0.4028	3624
1	0.1108	0.1171	0	0.6835	408

Table 9.2: Fraction of experiment time in same community based on country

Similarity	Average	Std. dev.	Min.	Max.	Users
0	0.0905	0.0583	0	0.4145	3808
1	0.1313	0.1425	0	0.6835	224

Table 9.3: Fraction of experiment time in same community based on city

Similarity	Average	Std. dev.	Min.	Max.	Users
0	0.0914	0.0629	0	0.4028	3788
1	0.1135	0.1067	0	0.6835	242
2	0.0239	0	0.0239	0.0239	2

Table 9.4: Fraction of experiment time in same community based on nationality

Similarity	Average	Std. dev.	Min.	Max.	Users
0	0.0906	0.0575	0	0.4028	242
1	0.0923	0.0604	0	0.6459	2828
2	0.1006	0.0853	0	0.6835	866
3	0.0856	0.0599	0	0.2961	94
4	0.1009	0	0.1009	0.1009	2

Table 9.5: Fraction of experiment time in same community based on spoken languages

Results about current country of residence (see Table 9.2) and city of residence (see Table 9.3) is clear: users from the same country or city have naturally been grouped in the same community more often in average. Some users have spent around 70% of the experiment grouped in the same community. Had SAND/SHARC community detection been used as a low-level protocol to favor electronic communications between those users, then it would have greatly enhanced their exchange possibilities.

Results about nationality (see Table 9.4) are also interesting. They show that users with same nationality are automatically put in the same community for an average of 11% and up to 56% of the experiment. Note that only one pair of users shared two common nationalities, which makes the result not relevant for statistical analysis.

Finally, results about shared languages (see Table 9.5) present a wider range of possible social similarities between users (from none to four shared languages). In the range of statistical significant results, we can see that users sharing at least a common language tend to be more likely grouped in the same community.

9.2.2.2 Scientific similarities

Reminding that results were collected during a scientific conference, we also show the significance of the communities found by SAND/SHARC regarding the scientific position and studied fields of the participants in Table 9.6 and Table 9.7 respectively.

Similarity	**Average**	**Std. dev.**	**Min.**	**Max.**	**Users**
0	0.0828	0.0506	0	0.4366	2088
1	0.1055	0.0816	0	0.6835	1694

Table 9.6: Fraction of experiment time in same community based on scientific position

Results based on scientific position (see Table 9.6) are also clear, as participants having the same position (basically student or professor) spend, in average and at maximum, more time being grouped in the same community.

The same way, we can observe that, in the range where results are statistically significants (more than 100 pairs found), there is a global trend that groups together the users having more studied topics in common (see Table 9.7).

Similarity	Average	Std. dev.	Min.	Max.	Users
0	0.0897	0.0659	0	0.4145	1128
1	0.0902	0.0632	0	0.5642	1102
2	0.0836	0.0622	0	0.4439	706
3	0.0921	0.0685	0	0.5472	398
4	0.0994	0.0723	0	0.6835	266
5	0.1042	0.0743	0	0.4551	142
6	0.1046	0.0567	0	0.2147	56
7	0.0869	0.05467	0	0.2399	52
8	0.1186	0.1420	0.0558	0.6459	32

Table 9.7: Fraction of experiment time in same community based on studied topics

9.3 Highlight

As conclusion of this chapter, we can reckon the following:

- Distributed community detection algorithms need to implement **specific techniques** to operate successfully on dynamic networks, otherwise, they may be plague by effects like **wandering community**;
- We have introduced SAND/SHARC which implements a better **community organization** mechanism, allowing to elect an **originator** amongst the nodes the most strongly attached to their community;
- SAND/SHARC also introduces the **break mode**, a community structure split mechanism that responds to drops in the intra-community density, due to the network dynamics, using a **constrained propagation of a freshness counter**, updated by the originator node;
- SAND/SHARC has been **successfully tested** over a various set of human and nature-inspired dynamic scenarios, including urban mobility;
- Using real-world human interaction traces, we have shown that SAND/SHARC generates **community structures that also have a social meaning**.

In the next chapter, we will envision applications of the SAND/SHARC algorithm in various concrete urban communication systems, including mobile social networks and vehicular ad hoc networks.

Contents

Chapter 10

Application to urban communication systems

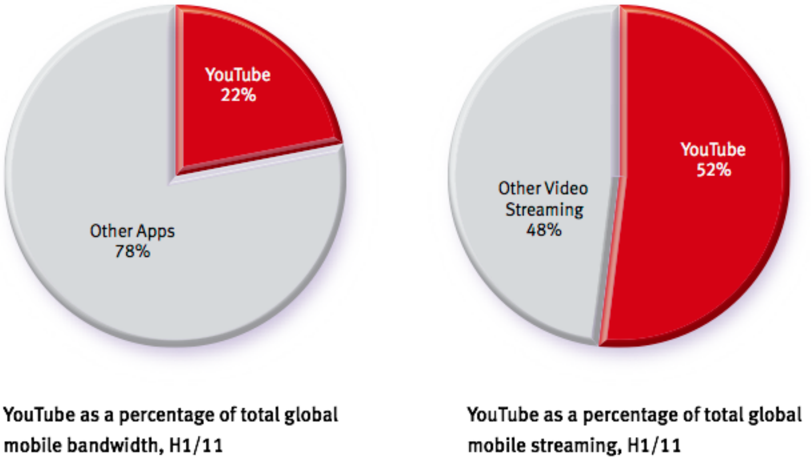
"We are not merely individuals struggling in isolation from each other,
but members of a community who depend on each other,
who benefit from each other's help,
who owe obligations to each other."
Tony Blair, Speech at Kennington Road, March 2005.

After presenting the internals and evaluating the performances of SHARC and SAND/SHARC, our contributions in the distributed detection of robust communities in dynamic graphs, we will introduce application of this solution to urban communication systems.

Based on concrete use-cases, in both mobile media diffusion and vehicular ad hoc networks, this chapter can be viewed as practical ideas or perspectives for actual implementation of our algorithms in real-life systems. The rationale for the use of communities in such communication frameworks is given as well as principles of realization.

10.1 Improving mobile media diffusion using communities

In subsection 4.1.2, we have described the current usage of mobile data traffic and presented its trends for the upcoming years. In the same section, we have reckoned wireless ad hoc networks as a solution to relieve part of the traffic volume occurring today on 3G networks, in an operation called *3G offloading*. A more detailed analysis of the traffic, reveals that around 40 percent of the current mobile traffic is related to video streaming, and that it is for more than its half concentrated towards YouTube, a popular website that proposes video streaming, as pictured in Figure 10.1.



YouTube as a percentage of total global mobile bandwidth, H1/11

YouTube as a percentage of total global mobile streaming, H1/11

Figure 10.1: YouTube fraction of mobile bandwidth and mobile video streaming volume for the first half of 2011 (source: Allot Communications)

In order to diminish a large amount of traffic that currently loads 3G and Wi-Fi networks, an efficient solution would be to allow users to exchange popular videos directly from device to device, using ad hoc communications. In this scheme, each device would use its hard drive space to cache recently viewed, favorites or most popular videos, making them available for device to device streaming. Moreover, those videos could be automatically downloaded at night, or whenever the phone detects it is connected to a private wireless network, such as a home network. This would allow to combine *on-the-spot offloading* with *delayed offloading*[LRL+10].

However, in previous chapters of this document (see subsection 4.3.2 and more generally chapter 6), we have highlighted the limitation of wireless ad hoc networks in terms of link stability and performance in multi-hop wireless paths.

In order to ensure sufficient reliability and to limit latency in delivery so that video streaming performances are acceptable by the end user, we propose to use the communities determined by SAND/SHARC in the following manner, provided that the communicating devices allow ad hoc communications:

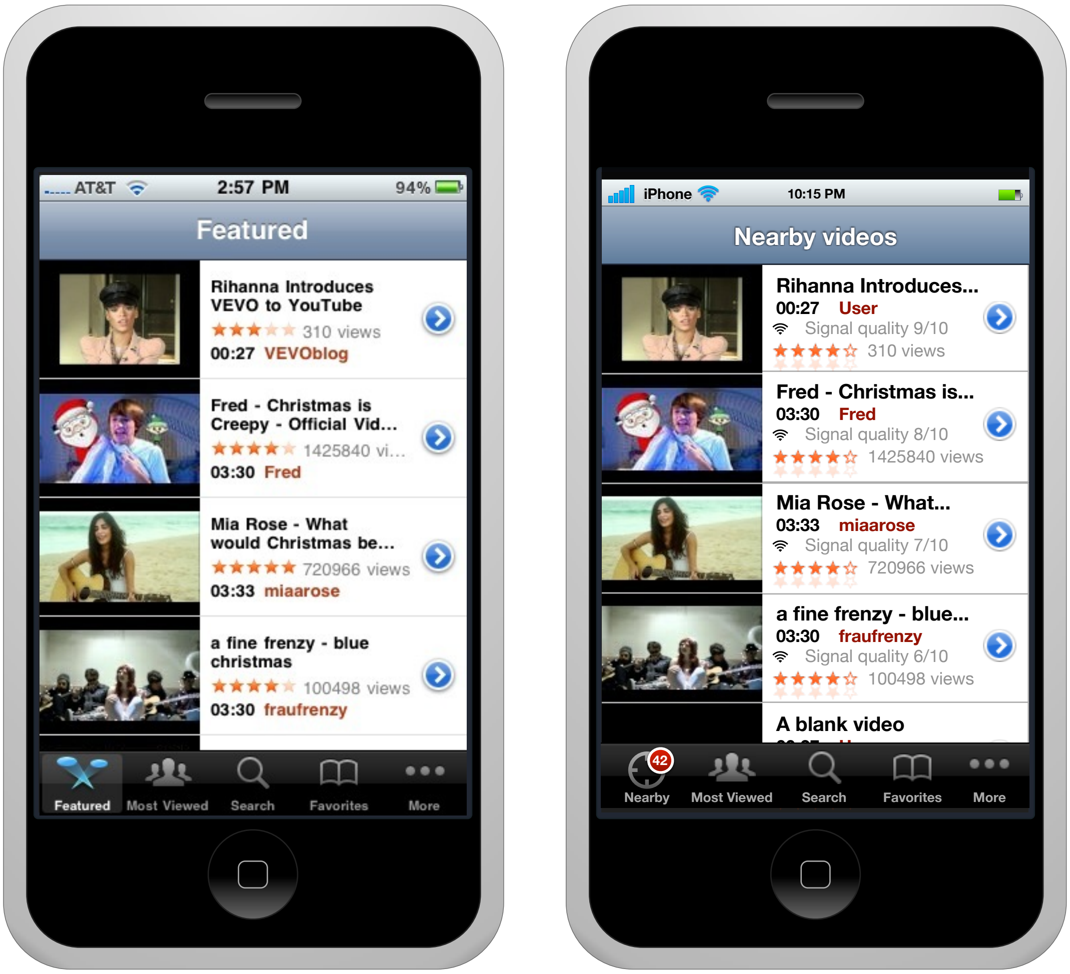
- Run the community detection algorithm so that each node belongs to a community including the most robust and dense links;
- Beacon the list of cached video in the device to the other members of the community;
- From the quality of the beacon messages (from the community detection or the video

list advertisement process) and the number of hops from current device to the streaming source, derive a link quality metric (for instance on a $[0..10]$ scale) for each available video;

- Present the list of available videos inside the community, ranked by decreasing quality;
- If the bandwidth permits it, start caching in background the video played by the user or the most requested videos, so they remain available even if the streaming source disconnects or leave the community.

For this approach to perform correctly it shall be made easy and transparent to the users, and be the default behavior of the application. The opening screen of the YouTube application on iOS devices, such as the iPhone, presents the "Featured" videos (the currently popular multimedia files). We propose to replace this screen with a "Nearby videos" screen presenting the videos available in the device current community, as explained above.

The changes in application are presented in Figure 10.2: the main screen of the application now shows the videos available in device-to-device streaming, sorted by decreasing link quality, which appears in the video information along title, length, number of views and user ratings.



(a) (b)

Figure 10.2: A screenshot of the YouTube application on iPhone: (a) Original version, (b) Modified version with ad hoc capabilities

Finally, opening the application automatically starts the available videos beaconing process, as well as the caching of the selected videos.

10.2 Improving traffic management using communities

In subsection 5.2.1, we have presented the concept of Vehicular Ad hoc NETworks (VANETs) as an ad hoc network composed of cars able to communicate to both roadside infrastructure (vehicle-to-infrastructure, or V2I, communications) and other vehicles (vehicle-to-vehicle, or V2V, communications) in oder to exchange safety, traffic or entertainment data.

In Figure 5.2, we have presented the infrastructure of such a system that would allow easy traffic management and planning. In this section, we will review shortcomings of the current solution and will see how the usage of robust communities can improve its operation.

10.2.1 Shortcomings

We will first describe the shortcomings of the proposed architecture, in the organization of both vehicle-to-infrastructure and vehicle-to-vehicle communications.

10.2.1.1 Shortcomings in V2I communications

The traffic management infrastructure presented in Figure 5.2 relies on V2I communications for cars of the network to report possible accidents or traffic perturbations to a centralized reaction chain composed of a centralized traffic management service that will gather and process all the information reported by cars that are part of the network. The gathered data can be completed by roadside units such as cameras or road sensors.

This information can be hard and costly to process because of the following:

- The **volume** of information can be high: as no clustering is applied to the network, all cars facing an incident or perturbation are likely to report it. The number of concerned cars can be high if, for instance, an accident is completely blocking a highway in a given direction;
- This large volume of reported information generates an unnecessary **load** on the wireless (3G or Wi-Fi) networks used by the cars in V2I communications, and on the infrastructure network to the traffic management service. This induces an additional **cost** to the operators, in the deployment and management of those networks, as well as to the users of the service, as they all need a subscription to access those networks;
- The reported information can also be **complex** to process: as each node reports its own data, which may include false positives and false negatives, sending contradictory information for the traffic management service to process.

Besides, the proposed architecture provides only a weak feedback loop to the users. The situation analysis, resulting from the interpretation of the collected data by the traffic management system, is only indirectly available to the users through a webpage that is hard to consult or radio broadcasting service. Besides, both may present information that is already outdated.

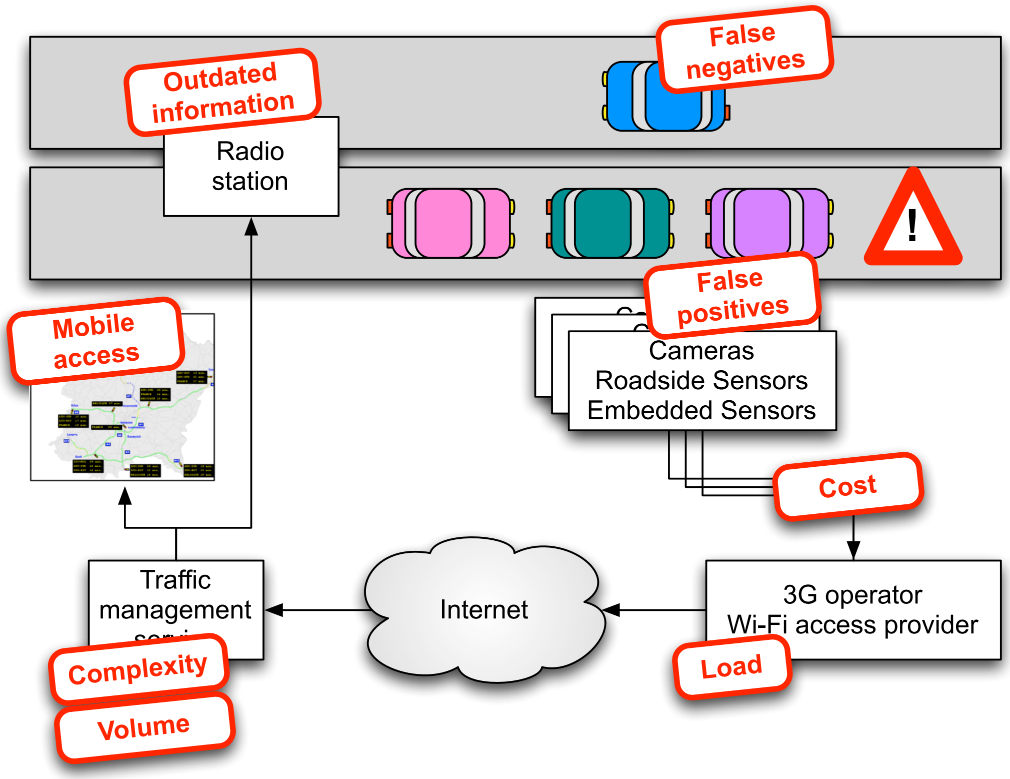


Figure 10.3: Shortcomings in vehicle-to-infrastructure communications

Those shortcomings are summarized in Figure 10.3.

10.2.1.2 Shortcomings in V2V communications

One might argue that using vehicle-to-vehicle communications to disseminate information may relieve the problems and limitations presented in the previous paragraph. In fact, V2V communications are generally used in vehicular ad hoc networks to quickly disseminate traffic and safety-related information to other vehicles.

However this pure ad hoc approach is subject to the following limitations:

- the **reliability** of wireless communications: as some communication opportunities are very short-lived (especially between cars going in opposite direction) or of low-quality (due to the extended distance between cars), it can be hard to guarantee that messages will be correctly delivered, which may cause a problem when the associated information is of importance for the users security;
- the **latency** in delivery can also be long if it uses long paths, leading to information to be outdated when it arrives at the intended destination;
- finally, the dissemination of many similar information, or the necessary amount of redundant transmission to improve delivery is also likely to generate **broadcast storms**.

In the scenario depicted in Figure 10.4, an incident on the main horizontal road creates a traffic jam. The information is propagated back from car to car using V2V communications. However,

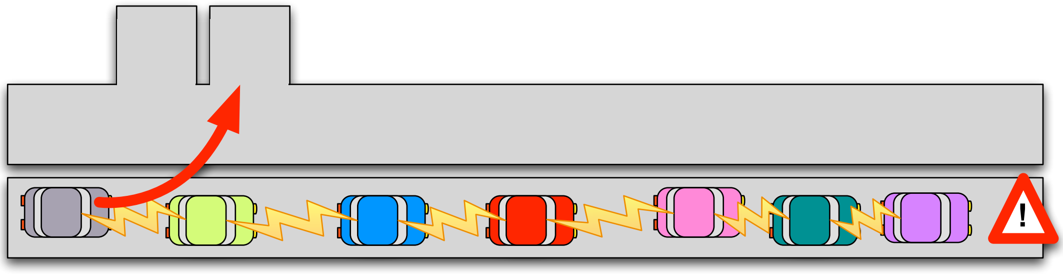


Figure 10.4: Example of limitations of V2V communications

due to the latency in the message propagation, many cars receive the information too late (after the crossroad) and are stuck without having the chance to opt for a different direction and avoid the incident.

10.2.2 Improvements using community structures

The introduction of the community detection, as a clustering algorithm to group cars that are able to reliably communicate in size-bounded structures, allows the following:

- Information reported by cars can be **analyzed and merged** at the community level, avoiding false negatives and false positives reports;
- The summarized information can then be **robustly disseminated** across the community. The delay in delivery is also kept low because of the limited span of each community;
- Central traffic management service can still be reached, but instead of each car reporting to it, **one node** (we propose the *originator*) can **send the aggregated information** concerning its community, minimizing cost and load on the infrastructure network;
- The **feedback loop can also be made more effective**, with the traffic management service responding directly to the originator of each community which will, to its turn, disseminate the reply in his group.

The latest point also prevents the effect presented in subsubsection 10.2.1.2: inter-community communication using infrastructure network can be viewed as "giant leaps" in information propagation, that will limit the latency in information delivery.

For even better performance and adaptation to this use-case, SAND/SHARC could easily be adapted to limit the diameter of the generated communities (by using the distance from the originator as limiting factor) and to wisely chose the node elected as originator (for instance, selecting the frontmost car of a lane or restricting the choice to cars with a valid 3G subscription in the current country).

10.3 Highlight

In this chapter, we have:

- Presented **actual use-cases** for community detection in both mobile data diffusion and vehicular ad hoc networks;
- **Motivated** the use of communities by an **analysis of the shortcomings** of current communication frameworks;
- **Provided solutions** to those shortcomings using community detection and ideas of implementation.

After those concrete application cases we will, in the next part, see how community detection techniques can benefit more globally to wireless ad hoc networks by making topology management, a set of techniques to control the network connectivity, more efficient.

V

Community structures and topology management

Contents

Chapter 11

Topology management

> "Management of many is the same as management of few.
> It is a matter of organization."
> Sun Tzu, The Art of War.

Due to the shared nature of the radio medium used by wireless ad hoc networks, as well as the broadcast nature of their transmission, nearby communicating devices are likely to *contend* for access to the communication channel. This problem has already been presented in subsection 5.1.1: nodes have to agree so that their transmission occurs on at least one different coordinate on the hyperspace composed by: space (contention area), time (transmission duration), frequency, message encoding.

Many of the distributed systems and their underlying protocols and algorithms see their performances collapse under some network conditions, particularly when the network density is high. One solution to cope with this problem is to manage the network topology, at the logical or physical level, in order to leverage those constraints.

As, by definition, communities are composed by denser areas in the communication network, controlling network density and avoiding the problems it may generate inside those structures is particularly important. We will also see how topology management solutions can benefit from information about communities in networks.

11.1 Network density and its effects

In this section, we will investigate the effect of the network density on its achievable performance (mostly related to the highest obtainable bandwidth). Those observations will serve as motivation for topology management.

11.1.1 Definitions

Before further analysis of the impact of network density on the network performance, we will more precisely define this notion and the related concept of contention area.

11.1.1.1 Network density

The density of a graph has been formally defined in definition 2.12.

In wireless communication networks, the existence of connecting edges between nodes is closely related to the distance between them (see section 6.2). In the simplest assumption (see subsubsection 6.2.1.1), the transmission range is set fixed in space, time and across nodes, leading to circular transmission ranges.

In those networks, the density is directly related to the physical density of the nodes, i.e. the number of nodes over the area in which they evolve. To eliminate reference to the actual range of transmission, all length unit can be normalized so that the transmission range is unitary.

11.1.1.2 Contention area

In many popular wireless communication systems, only space and time are available to diversify communications. Particularly, most wireless ad hoc systems consider that all nodes agreed *a priori* on a common transmission frequency and that their is no orthogonal coding used for communication. In this situation, two nodes in their respective transmission range will be contending for transmission.

Definition 11.1 (Contention area)
We can define a *contention area* as a zone within which all nodes are in contention with each other. As a result, devices in the same contention area will have access to the channel at different times.

The size of the contention area is directly related to the density of the network, whatever the considered definition of the density.

11.1.2 Network bandwidth

Definition 11.2 (Network bandwidth)

The *network bandwidth*, or *network throughput*, can be defined as the maximum value achievable by the sum of the data rates that are concurrently delivered from and to all the terminals of the network.

In the case of dense wireless networks, i.e. where there are a lot of nodes in each contention area of the graphs, measuring bandwidth with all terminals transmitting can lead to high delays in transmission (as each node may wait long for a transmission opportunity). When considering random source-destination paths, this results in low overall network bandwidth.

This effect has been formalized by Gupta *et al.* in [GK00] for a network composed of N wireless nodes in a unit disk area. In this model, as presented in subsubsection 2.2.1.2, all nodes have an identical circular transmission range and transmit at a data rate of W bits per second. Under those assumptions, the authors proved that the overall throughput obtainable between each node and a randomly chosen destination $\lambda(N)$ follows a law given by Equation 11.1:

$$\lambda(N) = \Theta\left(\frac{W}{\sqrt{N \log N}}\right) \tag{11.1}$$

As in the authors model, increasing the number of nodes in the fixed unit disk area increases the network density, this formal study corroborates our initial idea that the denser the network, the lesser the overall achieved performance in terms of throughput.

11.1.3 Broadcast storms

In subsubsection 4.3.1.1 we have already introduced the broadcast storm problem (see definition 4.5): many nodes of the network concurrently try to send broadcast messages, in order to give other nodes an up-to-date view of their neighboring network conditions.

This transmission of information can occupy a large amount of the available network throughput. Besides, as the broadcast process is not considered reliable, it is generally sent redundantly and some mechanisms (acknowledgments, dynamic Automatic Repeat reQuest, or ARQ) tend to increase the level of redundancy when packet loss increases.

As no efficient distributed channel access control mechanism can avoid collisions (see subsubsection 5.1.1.3), the denser the network, the more likely collisions will occur, as the number of nodes in contention for sending is higher. Then, due to redundancy, more broadcast messages are going to be sent. Therefore, a direct consequence of high network density is a vicious circle that strengthens the broadcast storm problem.

11.1.4 Solutions

A generic approach to leverage the problems related to high network density, and particularly broadcast storms, is to artificially control and limit the density of the network, either globally or on certain areas only. As per definition 2.12, this means diminishing the degree of some or all nodes or the network, i.e. tearing down some communication edges.

11.1.4.1 Logical solutions

One approach can consist in a logical control of the links between nodes: all links still physically exist, but some are inactivated or will not be used, or considered for communication. On the contrary, other edges will concentrate all communications. They will form a *virtual backbone*, by analogy with the spiral column, highway roads or even wired networks, where backbone infrastructures are entities of larger capacity draining most of the traffic.

Definition 11.3 (Virtual Backbone)
A virtual backbone is a subset of network elements (nodes, links) in charge of providing services to the complete network.

This solution is, at some point, easier to implement as it is logical only and does not involve any physical operation on or by the communicating devices. However virtual backbones generally result from the convergence of a distributed algorithm which, at its turn, has to deal with scalability and the possible dynamics of the network.

11.1.4.2 Physical solutions

Another approach consists in physically managing the network topology, i.e. actually designing the shape of the communication graph by tempering with the transmission capacities of the nodes. According to the definition given in the literature, this can be done either to optimize the network capacity, to avoid broadcast storms or improve the available bandwidth, or to limit the energy consumed globally by the network for its operation.

Definition 11.4 (Topology Control - Santi (San05))
Topology control is the art of coordinating nodes' decisions regarding their transmitting ranges, in order to generate a network with the desired properties (e.g. connectivity) while reducing node energy consumption and/or increasing network capacity.

Topology control methods physically reduce the number of edges of the communication graph, instead of logically inactivating them like in virtual backbones. This technique is a practical solution to leverage the scalability issues that may arise from the routing protocols.

However, a direct consequence is the increase of the characteristic path length $\bar{l}$ of the network, inducing more latency in information propagation across the network and a possible increased number of errors in the delivery of information packets, especially in dynamic networks, as longer paths are used. Finally, when transmission ranges are efficiently control, it is possible that all paths are subject to the *edge effect* we introduced in subsubsection 4.3.2.1.

11.2 Virtual backbones

In this section we will derive a more formal definition of virtual backbone from the practical definition and remarks we have given in subsubsection 11.1.4.1. We will also browse the different possible forms of virtual backbones and underline their specificities.

A far more complete overview of the concept of virtual backbones, associated performance metrics, and algorithms to generate them is available in [Sch10].

11.2.1 Definition

From the practical definition we have given in definition 11.3, we can derive that, from a graph theory perspective, a virtual backbone is simply a subgraph of the original communication graph that is able to offer communication services to the complete original graph.

Definition 11.5 (Virtual Backbone - Graph theory)
A *virtual backbone* $\mathcal{B}(\mathcal{G})$, or $\mathcal{B}$ for short, is a subgraph $\mathcal{B} = (V_{\mathcal{B}}, E_{\mathcal{B}})$ of the graph $\mathcal{G} = (V, E)$, with $E_{\mathcal{B}} \subseteq V_{\mathcal{B}} \times V_{\mathcal{B}}$, so that for each (X,Y)Path existing in $\mathcal{G}$, there exists a (X,Y)Path mostly through $\mathcal{B}$.

The condition about the existence of a path between two nodes of the backbone that had a path in the original communication graph is necessary to comply with definition 11.3.

This condition can be further expressed as the following:

1. $\mathcal{B}$ is a connected component;
2. Either:
   (a) $V_{\mathcal{B}} = V$;
   (b) $V_{\mathcal{B}} \subset V$ and $\forall u \in V \setminus V_{\mathcal{B}}\ \exists v \in V_{\mathcal{B}} / (u, v) \in E$

Statement 1 allows information to transit across the backbone: were the backbone disconnected, then it would not allow one part of the network to communicate with one another. Statement 2 ensures that all nodes have access to the backbone: in case (a) all vertices are already comprised in the structure, in case (b) nodes that are not part of the backbone are directly connected to it.

Without any formalism, it seems clear that one objective of the backbone structure is to contain as few entities as possible, while still maintaining network operations with a certain quality of service. The size objective can then be traded for robustness or resilience to network perturbations.

11.2.2 Taxonomy

From the different possibilities to meet the requirements of the virtual backbone definition, while optimizing its size, is born a different set of approaches that we will present here.

11.2.2.1 Tree-based virtual backbones

The case 2(a) presented in subsection 11.2.1 is chosen as paradigm for the construction. As all vertices of the graph are in the virtual backbone structure ($V_{\mathcal{B}} = V$), the set of edges $E_{\mathcal{B}}$ is to be chosen efficiently.

As statement 1 still applies, we are left with the following problem: finding the minimum set of edges $E_{min} \subseteq E$ so that the graph induced by (V, E_{min}) is connected. Then naturally E_{min} is chosen as the set of edges $E_{\mathcal{B}}$ of the backbone structure. As we will discuss later in chapter 12 (see particularly definition 12.1), this corresponds to the properties of the tree structure.

Tree-based virtual backbones are particularly efficient to solve routing scalability issues as they drain down the number of possible (X,Y)paths to one for any pair of vertices. They also help with broadcast or multicast as, to achieve those operations, it is simply necessary to follow the edges of the tree.

11.2.2.2 One-hop clustering

With the concept of communities, presented in section 3.3, we have already presented a unconstrained form of clustering. Communities have no pre-defined size and focus on grouping topologically-similar or densely interconnected nodes in the same cluster. We have envisioned application focusing on the most robust links, eliminating *weak ties*, which is similar to a topology management process using virtual backbones.

To respect the virtual backbone definition, it is easier to limit the span of the clusters to one hop: all generated clusters are then organized around a *clusterhead*, or cluster *leader* to which they are connected. All clusterheads are included in the set of backbone vertices $V_{\mathcal{B}}$. This allows to respect statement 2(b).

However, there is no guarantee that all elected clusterheads will be neighbors. From the previous description, two clusterheads a and b can be either neighbor, two-hop neighbor or three-hop neighbor. To ensure that Statement 1 is respected, additional nodes can be added to $V_{\mathcal{B}}$ until a set of edges $E_{\mathcal{B}} \subseteq V_{\mathcal{B}} \times V_{\mathcal{B}}$ so that $\mathcal{B}$ is connected can be found.

Another approach is to use *chained clusterhead election*: each node elects one neighbor as cluster-head in its direct neighborhood, their solution may lead to chains of cluster-head, i.e. a node u may elect v as cluster-head and v may elect x (u and x are not neighbors)[BAR07].

11.2.2.3 Dominating Set-based virtual backbone

The last category of approaches to create virtual backbone aims at creating a structure with the characteristics of a *connected dominating set*.

Definition 11.6 (Dominating Set)
A *dominating set* $\mathcal{D}(\mathcal{G})$, or $\mathcal{D}$ for short, for a graph $\mathcal{G} = (V, E)$ is a subset $\mathcal{D} \subseteq V$ so that $\forall u \in V \setminus \mathcal{D} \; \exists v \in \mathcal{D}$ so that $u \in N(v)$

Building a dominating set hence ensures the respect of statement 2(b).

Definition 11.7 (Connected Dominating Set)

A *connected dominating set* $\mathcal{D}_c(\mathcal{G})$, or $\mathcal{D}_c$ for short, for a graph $\mathcal{G} = (V, E)$ is a subset $\mathcal{D}_c \subseteq V$ so that $\forall u \in V \setminus \mathcal{D}_c \; \exists v \in \mathcal{D}_c$ so that $u \in N(v)$ and that, with $E_{\mathcal{D}_c} = \mathcal{D}_c \times \mathcal{D}_c$, the graph $\mathcal{G}_{\mathcal{D}_c} = (\mathcal{D}_c, E_{\mathcal{D}_c})$ is a single connected component.

Building a connected dominating set hence ensures the respect of statement 1.

Creating a backbone using $\mathcal{G}_{\mathcal{D}_c}$ is suitable as all devices may either be in the backbone or have at least a one-hop neighbor in the backbone. Moreover, the sub-graph induced by the components of the backbone is connected, i.e. a path only composed of backbone nodes exists between any pair of backbone nodes.

11.2.3 Equivalence

The maximum leaf spanning tree problem is equivalent to finding the minimum connected dominating set[Fuj03]. One-hop clustering algorithms create structures in which cluster-slaves have at least one cluster-head in their direct neighborhood (one-hop). Such characteristic corresponds exactly to the coverage property of a dominating set. If no weight values are considered, finding the minimum number of clusters (and cluster-heads) is equivalent to finding the minimum dominating set of the communication graph.

All those equivalences are illustrated by Figure 11.1.

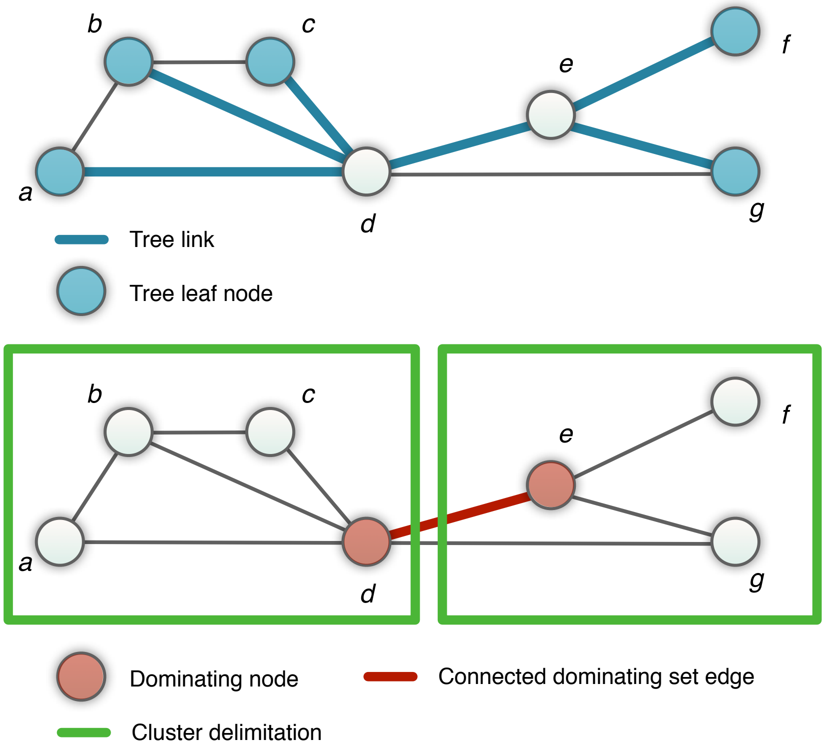


Figure 11.1: Equivalence between maximum-leaf spanning tree (top), dominating sets and clusters (bottom)

11.3 Topology control

In definition 11.4, we have given an operational definition of topology control. In this section, we will further detail its principles and present a short taxonomy of the existing approaches.

11.3.1 Principles

Wireless ad hoc networks are most of the time composed of devices that are not connected to a power source and rely on battery for operating. Besides, it is admitted that the transmission of data over the air is the most energy costly operation of those devices duty cycle. Reducing the transmission power while maintaining communication capabilities would hence extend the lifetime of the devices and of the network as a whole.

Due to the shared nature of the wireless medium, reducing the transmission power is also likely to reduce the number of collisions and the level of interference, and hence to favor successful communications. In that context, topology control techniques aim at increasing the network capacity by reducing the global amount of interferences.

This approach shall however be network-wide: reducing the consumed power by using a more energy-efficient transmission chain, or optimizing the energy used on a given path, or for a given transmission, is deemed not being topology control.

Besides, approaches resulting in a hierarchy in the network, like the election of some nodes, as in those presented in subsection 11.2.2, are not considered as topology control in the stricter sense.

11.3.2 Taxonomy

Topology control techniques can be classified depending on their underlying design to optimize network communications. This classification is illustrated in Figure 11.2.

11

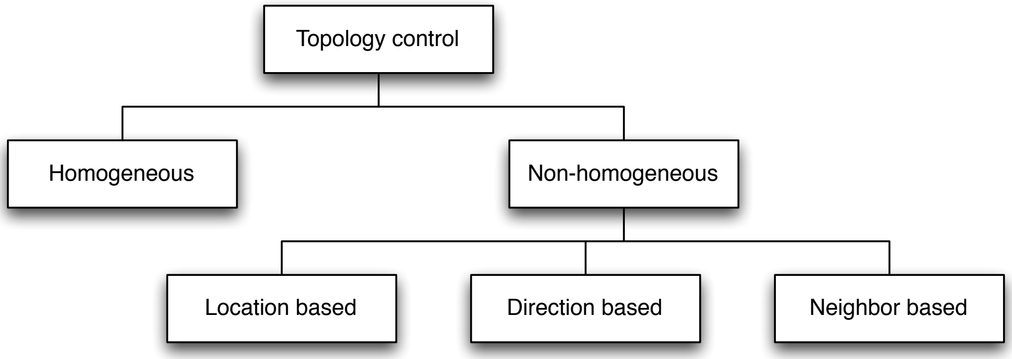


Figure 11.2: Taxonomy of topology control methods

11.3.2.1 Homogeneous methods

Homogeneous methods assume that all nodes will adapt their transmission power to a certain level, leading in the case of a simple geometric transmission function (like free-space path-loss) to a common transmission range r. The topology control problem consists then into determining the minimum value of r, often referred to as the *critical transmitting range* or CTR, that satisfies one or several network properties.

Generally, the CTR value is adjusted so that the network is connected. This allows the communication service to operate between any source-destination pair with a minimal transmission range, i.e. with minimal energy consumption for transmission, hence maximizing the network lifetime. This also reduces the size of contention areas, limits interferences and favors successful reception of the transmitted data.

The problem of critical transmitting range assignment is generally solved as a centralized problem, using global network knowledge. In fact, many formal expression have been derived for static and dynamic networks for simple geometric transmission function and various location assignment functions and mobility models[San05].

11.3.2.2 Non-homogeneous methods

Contrary to homogeneous methods, non-homogeneous methods are able to adopt different powers for transmission. The decision method can be centralized or distributed, with each nodes determining an optimal value based on the available local or global information.

Location based methods use accurate location information about the nodes and are often employed when GPS devices equip the communicating entities. Those methods can be centralized, similar to network planning, or fully-distributed[LHS03].

Direction based methods leverage the requirement about nodes position knowledge but assume that the communicating entities are able to estimate the relative direction of their neighbors, which can be done by equipping nodes with a direction antenna. As this method is based on physical signal estimation, it is more accurate than location based methods in environments with a lot of obstacles (where there is a little correlation between distance and received power). Besides, many distributed direction based algorithms[LHB$^+$01, BJ02] produce as good topologies as distributed location based methods in all cases.

Neighbor based methods rely on the adjacent nodes exchanging a minimum of information (identifier, distance, link quality) in order to rank the nodes. Based on this local information the transmission power is adapted. For instance, in the KNeigh algorithm of Blough *et al.*[BLRS03], a list of the K closest symmetric neighbors is constructed, based on received power information, and the transmission power is adapted to reach only those K nodes. Variations of this algorithm, like the XTC protocol [WBW01], are based on different metrics like experienced link quality and may consider the complete neighborhood instead of a subset of it for transmission.

11.4 Highlight

In this chapter, we focus on the main problems caused by network density in wireless networks and how to leverage them:

- High network density is likely to cause **drops in wireless network bandwidth** due to extended **contention areas**;
- Another important effect likely to appear is **broadcast storm** that can overload the network, or prevent convergence of algorithms relying on exchange of information between nodes;
- Network density can be **logically diminished** by selecting a subset of the communication graph entities and constructing a **virtual backbone**;
- Network density can also be **physically diminished** by adjusting the transmission power of the communicating entities using a **topology control** mechanism.

In the following chapters we will investigate **how the community structures can help** in those processes.

Contents

Chapter 12

Communities and spanning trees

> "Life is not found in atoms or molecules or genes as such,
> but in organization; not in symbiosis but in synthesis."
> Edwin Grant Conklin, Cell and Protoplasm Concepts: Historical Account.

In the previous chapter we have presented the main concepts of topology management, particularly the creation of virtual backbones. In this chapter, we will focus on the problem of generating a particular virtual backbone, called minimum spanning tree, that exhibits topological properties from which topology management can benefit.

However, the distributed generation of spanning trees is a complex problem, especially on dynamic graphs. After a more formal definition of this structure and a review of the existing algorithms for its generation, we will show how the concept of communities can improve this process, particularly in a distributed and dynamic system.

12.1 Minimum Spanning Trees

In this section, we will provide the formal definitions from graph-theory related to the tree structure.

12.1.1 Definitions

Consider a graph $\mathcal{G}$, composed of a single connected component (i.e. there exists a path between every pair of vertices of the graph). If the density of $\mathcal{G}$ is high enough, we can consider some edges as superfluous: the deletion of those links would not separate $\mathcal{G}$ into several disconnected components.

This consideration naturally raises the question of the minimum set of edges to keep in $\mathcal{G}$ so that $\mathcal{G}$ is connected. This is the informal definition of a *minimum spanning tree*.

12.1.1.1 Tree

In [BM08], the authors introduce the notion of tree as follows:

Definition 12.1 (Tree - Bondy and Murty (BM08))
A *tree* $\mathcal{T}$ is an acyclic graph (i.e. a subgraph with no cycle) that is *connected* (i.e. that forms a single connected component).

Then they deduce from this definition an interesting property of trees, stating that two vertices u, v of a tree $\mathcal{T}$ are connected by exactly one path.

12.1.1.2 Spanning tree

The same authors denote as *subtree*, any subgraph $\mathcal{T}(\mathcal{G})$ of a graph $\mathcal{G}$ that is a tree.

Definition 12.2 (Spanning tree - Bondy and Murty (BM08))
If $\mathcal{T}_s(\mathcal{G})$, or $\mathcal{T}_s$ for short, is a spanning subgraph (i.e. it contains all the vertices of $\mathcal{G}$ in a single connected component), and $\mathcal{T}_s(\mathcal{G})$ is a tree, then $\mathcal{T}_s(\mathcal{G})$ is called a *spanning tree* of the graph $\mathcal{G}$.

A spanning tree over graph $\mathcal{G} = (V, E)$ is a graph $\mathcal{T}_s$ with the same set of vertices V and with a set E' of edges comprised in the set E, such that there are no cycles in $\mathcal{T}_s$ and such graph $\mathcal{T}_s$ is connected.

Based on the observations made in subsubsection 12.1.1.1, $\mathcal{T}_s$ is a maximal set of edges of $\mathcal{G}$ that contains no cycle, or a minimal set of edges that connect all vertices of $\mathcal{G}$. Hence, $\mathcal{T}_s$ is often referred to as the *minimum spanning tree* over $\mathcal{G}$ and noted $MST(\mathcal{G})$.

12.1.1.3 Spanning forest

Whenever the considered network $\mathcal{G}$ is not composed of a single connected component or when, during the construction of a single spanning tree, there exist several disconnected subtrees across the network, the set of such existing entities $\mathcal{F} = \{\mathcal{T} \subset \mathcal{G}/\mathcal{T} \text{ is a tree}\}$ is referred to as a *forest*.

Definition 12.3 (Spanning forest)
When there exists a single subtree $\mathcal{T}_i$ per connected component $\mathcal{S}_{cc,i}$ of a graph $\mathcal{G}$, and that each tree $\mathcal{T}_i$ is a spanning tree for $\mathcal{S}_{cc,i}$, then the set $\mathcal{F}_s = \{\mathcal{T}_i\}$ is called a *spanning forest* of $\mathcal{G}$.

12.1.2 Motivation

The notion of spanning tree is very interesting in communication networks. For instance, in a wired network interconnecting cities of a country, the spanning tree represents the minimum set of links to maintain so that any other city can be (directly or indirectly) reached. Identifying such a set can be economically advantageous, although it reduced the level of redundancy and hence the reliability of the network (remind the discussion in subsection 4.2.2).

Spanning trees are also a natural solution for multicast communications: if a given vertex v wants to transmit a message to all the other vertices of the network, then, using the spanning tree edges as communication links is not only sufficient to ensure complete destinations coverage but also minimizes the utilization of network.

In social networks, identifying a spanning tree of an interaction graph can help prune down relations that can be considered as not essential as they are not required to be related, directly or through a path of relations, to a given person.

Spanning tree techniques generally rely on some weights to compute the most suitable edges. Such weights can be placed on nodes or edges depending on which scenario is considered. In such a context, the problem is generally renamed as *minimum weight spanning tree* and thus consists in finding a spanning tree whose cumulated weight is minimal.

Finally, spanning trees are used in ad hoc networks because, by using only a subset of edges, they reduce the complexity of routing protocols, without changing the overall connectivity of the network.

However, in the particular context of ad hoc networks, it is required that the spanning tree is constructed in a distributed manner and using a local subset of the information available about the network. Besides, as those networks are dynamic, the generated spanning tree has to constantly adapt its topology to be consistent with the evolution of the connections and nodes status and to respect the required properties (connectivity and absence of cycles).

12.1.3 State of the art

While the literature has widely covered the problem of centralized generation of spanning trees on static networks, only few distributed solutions have been presented. We will briefly review the different approaches for solving this challenge, emphasizing on the recently-proposed distributed solutions that are also applicable to dynamic networks.

12.1.3.1 Historical approaches

One of the earliest references to a problem analogous to finding the minimum spanning tree can be found in a 1926 study by Czech scientist Boruvka[NMN01]. Its purpose was an efficient electrical coverage of Moravia, an historical region in Central Europe in the east of the Czech Republic. This application can be related to the illustrative example provided in subsection 12.1.2.

But the most widely known algorithms are attributed to Prim[Pri57] and Kruskal[Kru56]. They are greedy algorithms running in polynomial time. Another well-known algorithm to solve the spanning tree problem has been proposed by Tarjan[CT76] and is based on a depth-first search approach.

12.1.3.2 Parallel algorithms

There is also a set of parallel algorithms that were designed to solve the spanning tree problem or its variations, especially the minimum weight spanning tree or the maximum leaf spanning tree.

The former, that we already introduced in subsection 12.1.2, considers that either nodes or edges, or both, are characterized by weight values. The objective is then to compute a valid spanning tree with the smallest aggregated weight, i.e. the sum of the spanning tree edges, or nodes, or both elements weight values.

The maximum leaf spanning tree problem requires to find the spanning trees containing the maximum number of leaves, i.e. vertices connected to only one vertex in the tree. Exact approaches to optimally solve this problem have been proposed in[Fuj03].

12.1.3.3 Distributed algorithms

Many distributed algorithms have also been created, especially to generate efficient structures for multicasting in wireless ad hoc networks. One of the earliest contribution is the Spanning Tree Protocol presented in RFC 1493 [DLRM93]. It is used by OSI link layer devices to create a spanning tree using the existing links as the source graph in order to avoid broadcast storms in classical networks.

Multicast routing: Other solutions are generally multicast extensions of ad hoc routing protocols. We can quote Multicast AODV[RP97] that extends the AODV mechanism presented in subsubsection 5.1.2.2, the Multicast Zone Routing Protocol (MZRP, see [ZJ04]) that uses a split of the operation area in zones as in the location-aided routing protocols presented in section 5.1.2.4 and the Associativity-Based Ad hoc Multicast routing (ABAM) protocol that uses link stability to construct a robust multicast tree, in an approach similar to what is presented for unicast routing in section 5.1.2.4. They are generally classified based on the fact that a single spanning tree is constructed for all multicast transmissions (*shared tree based*) or that each multicast source creates its own spanning tree (*source tree based*).

Graph relabeling: A systematic approach to guaranty cycle-free structures using dynamic graph relabeling has been proposed by Casteigts et al. (see [CC05, Cas06]) and extensively developed since then by Piyatumrong et al. in ([PBGL08, PBGL09]). The main idea is to build a spanning tree by iteratively merging smaller trees. The cycle-free property is obtained thanks to the usage of a token: each tree possesses a unique token which is regularly transferred from one node of the tree to another. Two trees are allowed to merge if and only if their respective token are located on neighbor nodes, i.e. the node u of tree $\mathcal{T}_u$ and the node v of tree $\mathcal{T}_v$ are neighbors and both have the token of their tree.

12.2 Distributed spanning tree generation using communities

In this section, we will try to show how the use of community structures can improve the spanning tree generation process. We have chosen to focus on the graph relabeling method, as it is independent of any routing protocol, is designed for general purpose, can easily be extended and propose a systematic and formal approach to the distributed generation of spanning tree.

12.2.1 DA-GRS and its shortcomings

DA-GRS (cf. [CC05, Cas06]) (Dynamicity Aware Graph Relabeling System) builds a spanning tree over some graph by taking into account locality of communications, i.e. it uses only one-hop knowledge. It also takes into account the dynamicity of the network.

DA-GRS provides a generic way to design higher level algorithms that must react to events, such as the appearance and disappearance of nodes in the neighborhood, by providing an interface that can be implemented at application level.

12.2.1.1 DA-GRS principles

DA-GRS uses tokens to build the spanning tree. There is one token per subtree. At the beginning all the nodes in the graph have a token and thus each node can be considered to be a subtree on its own. The merging process of two subtrees can only take place between two nodes that both have a token. By allowing only nodes that both have a token to merge their respective subtrees, DA-GRS ensures that a tree will be built. After the merging process is complete one of the tokens will be deleted. The tokens circulate through their respective subtree either randomly or based on some heuristic. DA-GRS converges when all the subtrees of a connected component have merged to form one big tree. This final tree is a spanning tree.

The building of the spanning tree by DA-GRS is based on four rules listed below and illustrated in Figure 12.1:

- *Token regeneration*: If a subtree doesn't have a token a new token will be regenerated
- *Breaking of a connection*: If a tree link between two nodes breaks, which means an edge is deleted, the node that is on the side of the link where the token currently is has nothing to do in order to maintain a token in his subtree, but the other node of the deleted edge has now lost the route to the token, so this node has to regenerate a token.
- *Merging of subtrees*: During the merging process of two subtrees, the two two of the two subtrees will merge and one of the token of the two nodes will be deleted.

- *Circulation of the token*: During the building process of the spanning tree each token in the subtree will circulate to its subtree randomly or based on some heuristics.

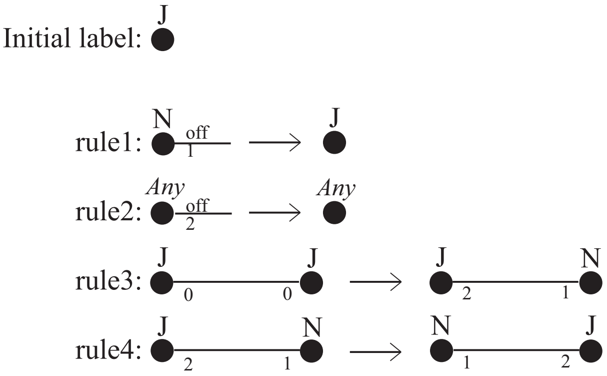


Figure 12.1: Visual representation of the DA-GRS rules

12.2.1.2 Shortcomings

To ensure the absence of cycles in the generated tree, DA-GRS relies on a token-passing system: in each subtree there exists a single token which moves, from node to node, along the tree edges. The tree merging process requires that, for a tree edge to be created, the two endpoint vertices own the token at the moment of merging.

While this process is proved to converge correctly[CC05], it is particularly inefficient, especially on large networks or when large subtrees have to be merged. As one token exists per subtree, the larger the subtree, the smaller the probability that the token resides at a given vertex at a given time. This makes the merging of two subtrees less and less likely to happen between two given vertices as the size of any of the two concerned structures increases.

More formally, an analysis of the probability of presence of a token t moving randomly along the edges of a tree $\mathcal{T} = (E, V)$ at a given vertex v, lead by Pigné in [Pig08] gave:

$$P(t, v) = \frac{\text{degree}(v)}{\sum_{v' \in E} \text{degree}(v')} \tag{12.1}$$

where $\text{degree}(v)$ denotes the degree, or number of neighbors of vertex v. The probability of presence of the token t at node v is the ratio of the degree of the considered vertex, over the total of the degree of all the vertices of the tree $\mathcal{T}$. For homogenous graphs in density, this confirms our initial analysis.

In the same publication, Pigné showed by simulation that the DA-GRS algorithm is very efficient to rapidly reduce the number of subtrees when there are many small entities (i.e. at the beginning of the spanning forest generation over a network). However, this efficiency plummets when the size of a tree increases and the number of tree to merge diminishes. For instance, Figure 12.2 shows the number of required token movements to reduce the number of trees on a graph composed by 170 vertices and around 1500 edges, starting with 10 subtrees. We can see that the number of movements to merge trees increases exponentially as the number of residual subtree diminishes.

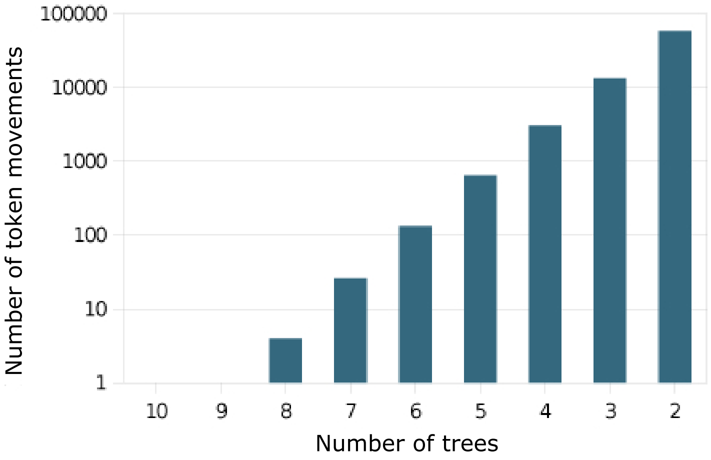


Figure 12.2: Number of required movements to reduce the number of subtrees[Pig08]

Trying to cope with this effect, the author tried to refine the token movement. One simple improvement to limit the wandering of the token is to use a tabu list that prevents back-and-forth movements within a certain history of positions. However, this only yields a marginal improvement, as shown in Figure 12.3. For various maximum authorized iterations, using a tabu list only diminishes the number of generated subtrees by 10 to 20%.

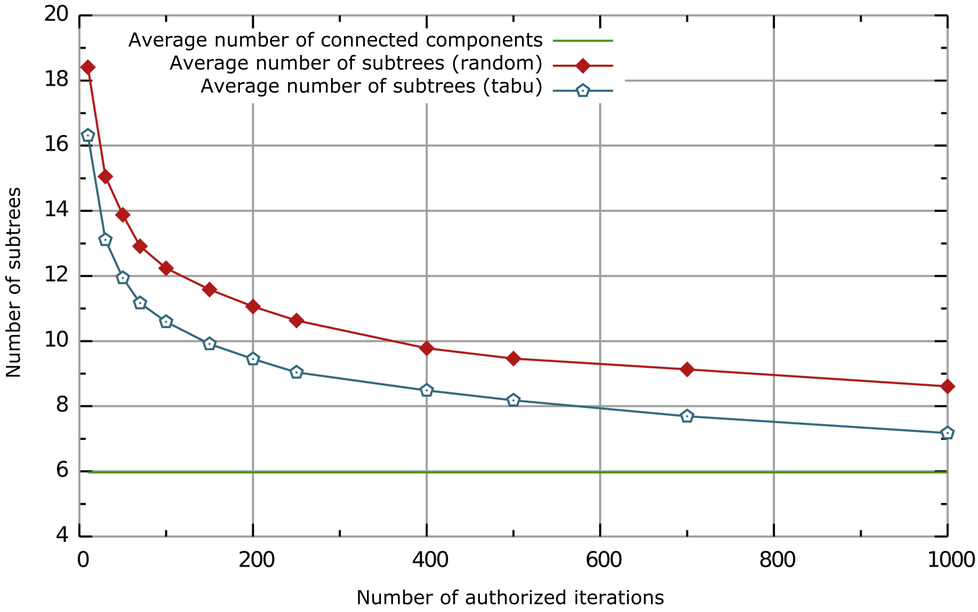


Figure 12.3: Number of subtrees after convergence for various token movements[Pig08]

Based on those results, a more suitable approach for improvement would be to reduce the size of the subtrees DA-GRS has to create and/or the size of the network on which it has to operate.

Communities are area of high link density inside a network, this means that many of the edges internal to a community can be considered superfluous (in the sense given in subsection 12.1.1). Therefore, this is *inside* a community that the creation of a spanning tree will conduce to pruning most links the quickest.

Besides, bounding the creation of a spanning tree to the extent of a community diminishes the number of nodes considered by each *instance* of DA-GRS and limits the expected size of the generated spanning tree (that covers only the nodes inside a given community, as opposed to the entire network).

Hence, we propose to use information from community detection to design modified versions of DA-GRS that reflects the community structure in the spanning tree, in order to increase its efficiency (prune more links), or increase the convergence speed of the protocol.

12.2.2 D2A-GRS

As a modification proposal, we present D2A-GRS, Dynamicity and Density Aware Graph Relabeling System. D2A-GRS uses the *neighborhood similarity* to detect density drops and favor the creation of tree links in the denser area of the graph first.

12.2.2.1 Principles

Consider two nodes u and v with both of them having a token (thus they are in different subtrees) and with a radio link. The main difference between DA-GRS and D2A-GRS is that DA-GRS merges trees as soon as two tokens meet. Thus DA-GRS would merge the subtrees of u and v whereas D2A-GRS would only merge the trees if nodes u and v have a high respective neighborhood similarity. Recall that the neighborhood similarity metric is a symmetric measure (definition 8.1).

However we need a high preference threshold to define when two nodes are similar enough. A node v is considered to be of high similarity by node u if and only if :

$$n_{sim}(u,v) \geq \overline{n_{sim}}(u) \tag{12.2}$$

where $\overline{n_{sim}}(u)$ is the average of all $n_{sim}(u,v)$ for all $v \in N(u)$.

The problem with the *neighborhood similarity* metric is that a node might wait forever for a highly similar node to obtain a token, because both nodes are already in the subtree. We refer to this as the *growing subtree problem*. As each node has local knowledge, a node has no easy way to determine whether another node is already part of the same subtree (see Figure 12.4 for an illustration of this problem). In this example nodes 6 and 13 have the possibility to merge their subtrees. They have low relative neighborhood similarity but in this case all the nodes with a higher similarity score are already in the same subtree, i.e. for node 13 nodes 7, 11 and 12 are already in its subtree and for node 6 nodes 0, 5 and 4 are also in its subtree. There is no easy way for nodes 6 and 13 to find out whether preferred nodes are already in the same subtree.

This can, however, be easily solved as a tree token may contain some information. Thus, we just need to keep track of the nodes that are part of each subtree, by adding this information to the token. In that case a node that gets the token can easily determine whether a preferred node is already part of the subtree and whether it makes sense to wait for an occasion to create a link with this node.

This information can be added easily to the token. At the beginning of the D2A-GRS algorithm each node is a subtree on its own, just as for the normal DA-GRS algorithm. Thus a node just has to add its own identifier information to the token. In subsequent merging processes, before

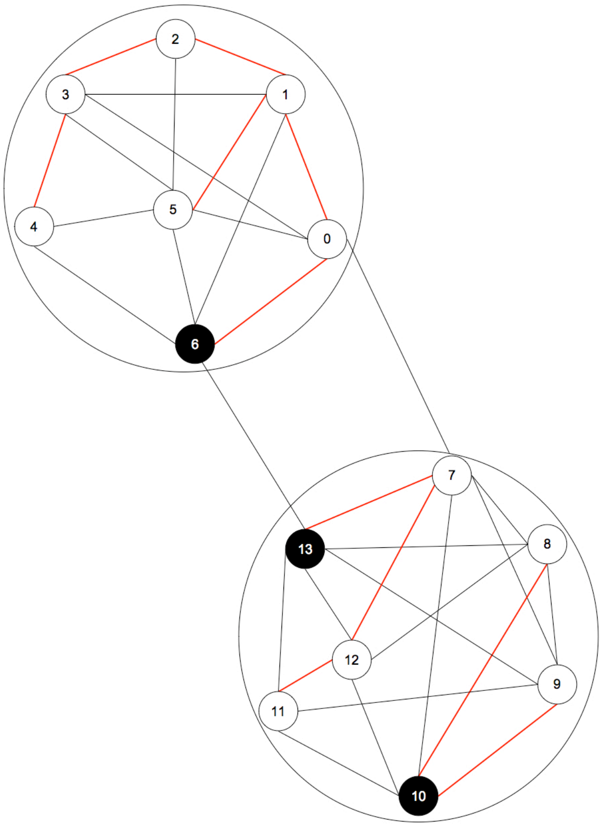


Figure 12.4: Illustration of the growing subtree problem

deleting a token, it is sufficient to add its information to the remaining token and we will have information about the complete subtree in the token. Note, however, that this algorithm works only efficiently for static networks, because after a breaking of a link, the information in the token will be outdated.

In D2A-GRS, if a node u has the possibility to merge trees by creating a tree link with another node v it does the following:

- determine the set $N(u)$ of one hop neighbors;
- compute its neighborhood similarity with all nodes in $N(u)$;
- remove all of the nodes that are already part of the same subtree from the table of similarities;
- calculate the average similarity score of the remaining nodes;
- if the similarity score of node v is above this average, be ready to create a new link (it will only be created if node v is also ready to create a link with u)

This method is very efficient, as we are determining the similarities of all neighbor nodes in parallel. The D2A-GRS algorithm is based on DA-GRS which has been designed in such a way that it uses only local knowledge and thus fits perfectly in a decentralized environment. Our modifications in D2A-GRS do not affect the decentralized design of the original DA-GRS algorithm. In order to be able to calculate a preference table a node needs to know the neighbor set of each neighbor. In a real world scenario nodes can advertise their neighbor sets by using periodic beaconing.

12.2.2.2 Experimentation setup and metrics

As mentioned in the section about D2A-GRS this algorithm has been designed for static networks in a first iteration. Therefore we will start our assessment with static networks. To do those tests, we relied on the Girvan-Newmann experiment, already presented in subsubsection 8.3.2.1. We implemented and tested the D2A-GRS algorithm using GraphStream[Gra].

To be able to evaluate the results we have passed the predefined communities to the simulator, but the simulator used those only after D2A-GRS had finished to build the tree to evaluate results. We present two metrics that we used to evaluate the results. It is due to these metrics that we need a priori knowledge about the correct community assignments:

Inter-community-tree-edge metric (ICTE) is based on the idea that if we build a tree of n nodes, then we need $n-1$ edges to link the different nodes. In the case of an optimal spanning tree, all of the nodes of one community will be in the same subtree. An optimal spanning tree can thus be broken into c subtrees (if c is the number of communities), where each subtree will group all of the nodes that belong to one community. If we want to reconnect the different spanning trees later on it becomes clear that we need $c-1$ tree links. Thus if a spanning tree is *commununity-optimal*, then there are only $c-1$ inter-community-tree-links. Based on this idea we suggest the following metric to evaluate spanning trees:

$$ICTE = \frac{c-1}{n} \tag{12.3}$$

where $ICTE$ is the quality of the results given as a value between 1 and 0 (1 being perfect and 0 disastrous), c is the number of communities and n is the number of inter-community-tree-links.

The advantage of this metric is that it is very efficient to compute. It has however a serious disadvantage in the sense that it is very aggressive especially for a very low number of communities. Another serious disadvantage is that it does not take into account the exact number of misplaced nodes.

The Misplaced Nodes metric (MN) considers the number of subtrees inside each community. In the case of an optimal spanning tree there is only one. If the spanning tree is not optimal we take into account the size of the different subtrees, the size of a subtree being given by the number of nodes in the subtree. The nodes that are part of the subtree with the biggest size are considered to be placed correctly, while all the other nodes are considered to be misplaced. This gives us the following metric:

$$MN = \frac{n - \sum\limits_{i=1}^{c} m_i}{n} \tag{12.4}$$

where MN is the quality of the results, c is the number of communities, m_i is the number of misplaced nodes in the i-th community.

The advantage of this metric is that it is much more accurate than ICTE as here we count the exact number of misplaced nodes. On the other hand this metric is not so efficient in terms of computation time.

12.2.2.3 Community matching and Observations

Before running some tests for D2A-GRS, we ran some tests on DA-GRS to see how good it performs with respect to community detection.

Original DA-GRS: Table 12.1 contains some of the results for DA-GRS. They represent experiments derived from the Girvan-Newman experiment, already presented in subsubsection 8.3.2.1. We have constructed several 90-node networks with 4 to 16 communities and a z_{out} of 2. Presented values are averages of 100 runs for each value.

Nr. of communities	4	8	12	16
ICTE	0.24	0.62	0.73	0.76
MN	0.88	0.94	0.94	0.92

Table 12.1: Inter-community-tree-edge metric and Misplaced Nodes for the DA-GRS algorithm and low z_{out}

As expected ICTE is much more aggressive, especially for a small number of communities. Our result files showed that for MN the size of the misplaced subtrees is usually one, which is another explanation why the quality of results when evaluating with ICTE is so much worse than when evaluating with MN. A size of one means that the number of excessive inter-community-tree edges is approximately equal to the number of misplaced nodes (i.e. $n_{exc} \simeq n_{mispl} = n$).

However, in the case of ICTE we compare this number n to the number of communities which is a very small number while in the case of MN we compare it to a much bigger number, namely the number of communities. As this size of misplaced subtrees is usually 1, MN seems to be much more precise than ICTE.

It seems to be very surprising that DA-GRS is performing so well with respect to the misplaced nodes metric, because DA-GRS was not designed to detect communities. However, as at the start of the algorithm each node is a subtree on its own and as the density of links inside the communities is much higher than between different communities, the probability that a node will merge subtrees with a node from the same community is very high, especially during the first phase of the tree building process. Thus DA-GRS detects communities quite well when the community structure is clear, i.e. inside the communities the density of links is much higher than between the different communities. We obtained similar numbers for different numbers of nodes. For a higher z_{out}, however, results are disastrous even for the second metric as is shown by Table 12.2, that contains some results for a z_{out} of 8.

Nr. of communities	4	8	12	16
ICTE	0.07	0.18	0.28	0.31
MN	0.49	0.60	0.68	0.64

Table 12.2: Inter-community-tree-edge metric and Misplaced Nodes for the DA-GRS algorithm and high z_{out}

D2A-GRS: For D2A-GRS we ran similar tests. Table 12.3 contains the results that we obtained for D2A-GRS when using the MN metric.

Nr. of communities	4	8	12	16
MN for $z_{out} = 2$	1	1	1	1
MN for $z_{out} = 8$	0.51	0.90	0.91	0.84

Table 12.3: Misplaced Nodes for the D2A-GRS algorithm and several z_{out} values

We can see that for networks with a clear cut community structure (as for z_{out} = 2) results are perfect with respect to community detection. The spanning tree that is built by D2A-GRS is always optimal in that case. However, when the density of links between different communities gets close to the density of links inside communities, as is the case for a z_{out} equal to 8, when the average degree of the nodes is 16, results get much worse. However, in that case one might argue whether it really makes sense to speak of different communities, because communities are defined as having a high density of links inside and a low density of links between them which does not hold true in that particular configuration.

12.2.2.4 Convergence Time of DA-GRS and D2A-GRS

Intuitively one would say that D2A-GRS will take a lot more time to converge than DA-GRS, because a node only merges subtrees when the other node is a preferred node, whereas DA-GRS merges with every node. We did some tests to find out whether our intuition is right. The same tests (i.e. on the same networks) were done for both DA-GRS and for D2A-GRS.

During these tests we used two token movement strategies. The first strategy is random token movement (RTM). The second is depth first token movement (DFTM), which works in the same way than a typical depth-first-search.

We measured the convergence time, by counting how many token movements are necessary to converge. Thereby we do not count all of the token movements, because most token movements are executed in parallel. We count how many times the final token (the token that will remain when the spanning tree is complete) has been moved around during the building process. Table 12.4 lists our results. All of the values are average values for 100 runs. The networks on which we did the tests, have 90 nodes, an average degree of 16 per node and an average z_{out} of 2.

Nr. of communities	4	4	8	8
DA-GRS version	orig.	D2A	orig.	D2A
DFTM	49.90	54.06	65.33	68.52
RTM	58.66	78.19	73.10	93.32

Table 12.4: Convergence time of DA-GRS and D2A-GRS

From these results we can observe that DFTM is a much better token movement strategy than RTM for both DA-GRS and D2A-GRS. The gap between DFTM and RTM is quite significant. Another observation that can be made is, that while there is a difference in convergence speed between DA-GRS and D2A-GRS, it is not of big significance.

Thus our intuition that D2A-GRS is converging much slower than DA-GRS is wrong. This can again be explained by the fact that, as at the start of the algorithm each node is a subtree on its own and as the density of links inside the communities is much higher than between different communities, the probability that a node will immediately have the possibility to merge subtrees with another node that has a high preference score is high, especially during the first phase of the tree building process. The marginal difference results from the last phase of the tree building process, because then it will take a bit more time before two nodes with high preference can merge their subtrees. Nevertheless the impact of this is very small as can be seen from the results.

12.3 Highlight

This chapter allowed us to present the following:

- We formally introduced the notion of **spanning tree** and underlined its importance regarding wireless ad hoc communications;
- We drew a **state-of-the-art** of the spanning tree algorithms for such networks;
- Particularly, we focused on **DA-GRS** (Dynamicity Aware Graph Relabeling System), presenting its principles, strengths and shortcomings;
- We showed how the **community information** can help in improving this solution for distributed generation of spanning trees of dynamic graphs.

In the next chapter, we will see how community structures can help in the construction of another topology management structure: virtual backbones.

Contents

Chapter 13

Communities and virtual backbones

"We could, however, also use [the Internet] as an Invisible College, the communicative backbone of real intellectual and civic change."
Clay Shirky, How Is the Internet Changing the Way You Think?.

In this chapter, we propose to study an approach to build a stable, size effective virtual backbone which benefits from existing community structures.

For the purpose of this study we will be using the SAND/SHARC algorithm presented in chapter 9 to generate communities using an epidemic label propagation scheme. On top of these communities we will implement a series of algorithms inspired by and using Blackbone2[SBA09] and its proposed optimizations[SDBA09] to build a more stable stable backbone.

We will present a brief literature review of distributed backbone creation algorithm, in order to justify our choice of Blackbone2 as a starting point, and then present the setup, and analyze the results of our experimentation.

13.1 Distributed creation of virtual backbone

Many distributed algorithms have been proposed in the literature, covering many different approaches and having many different characteristics. As in [Sch10], which presents a more detailed state-of-the-art, we will distinguish between self-pruning, maximum independent sets, multipoint relays, and algorithms for generating robust backbones.

13.1.1 Self-pruning algorithms

Self-pruning algorithms are composed of two phases, computed locally. In a first step, a connected dominating set is created by adding nodes using a certain heuristic, or *marking process*. In a second step, superfluous nodes, that entered the structure due to the approximations of the used local heuristic, are removed from it by applying a set of *pruning rules*.

Most self-pruning algorithms use a two-hop neighborhood knowledge for the marking process. For instance, in the Wu and Li algorithm[WL99], a node $n \in V$ is added to the connecting dominating set if and only if there exist vertices $u, v \in V$, so that $(u, n) \in E$ and $(n, v) \in E$, but $(v, u) \notin E$. This information is locally available at node n as it periodically receives information about $N(u)$ and $N(v)$ from u and v, respectively.

While the marking process aims at creating valid connected dominating sets, the efficiency of those algorithms in terms on generated structure size depends on the pruning rule. Wu and Li proposed a pruning rule based on neighborhood inclusion and identifier as tie break, later refined by Stojmenovic *et al.* who based the tie break on the nodes degree[SSZ02].

The major problem of those algorithms is that they necessitate the two phases to be efficient, which may not be suitable with highly dynamic networks and requires coordination between nodes. Besides, they may lead to incorrect or suboptimal results if the topology of the network noticeably changes between the marking and the pruning operations.

13.1.2 Maximum Independent Set algorithms

Those algorithms are building a connected dominating set by using the graph theory-related structure called *maximum independent set*. Maximum independent sets are also connected dominating sets.

Definition 13.1 (Independent set)
An *independent set* $I \subset V(\mathcal{G})$, is a set of vertices such that any pair of vertices $(u, v) \in I \times I$ is not adjacent. Equivalently, each edge of the graph $\mathcal{G}$ has one endpoint in I.

Definition 13.2 (Maximum independent set)
A *maximum independent set* $MI \subset V(\mathcal{G})$, is an independent set so that there is no independent set $I \subset V(\mathcal{G})$ such that $MI \subset I$.

Distributed maximum independent sets creation is first achieved through the election of one[BDTC05] or multiple leaders[AWF02]. The main principle is that one or several nodes decide to be *dominators* and send such a message. Upon receiving a message from a dominator, nodes are considered *dominatees* and send the corresponding message. Nodes receiving a message from a dominatee consider themselves as dominators and advertise it. The set of dominator nodes is an independent set.

Then, the set of dominating nodes is connected, to compose a connected dominating set. This can be achieved by identifying a path between dominators and adding all the dominatee nodes on this path to the final backbone set.

As self-pruning algorithms, maximum independent set algorithms require two passes, which can impair the proper construction of the connected dominating set in highly dynamic networks. However, the different phases of the algorithm do not require tight node synchronization.

13.1.3 Multipoint Relay algorithms

Distributed multipoint relay algorithms consist in each node independently selecting a subset of neighbors as relays to cover its two-hop neighborhood. This approach is used in the OLSR routing algorithm that we presented in subsubsection 5.1.2.3.

The *multipoint relays* (MPR) election is the first step of the algorithm. MPR are elected using heuristics that guarantee the respect of the two-hop neighborhood coverage rule, like those presented in Ajith *et al.* in [AJV05] and used in the OLSR protocol. A more efficient approach has been presented by Wu in [Wu03].

The application of MPR election heuristics generates too many MPR nodes and a second step consists in selecting a subset of MPR nodes to compose the final connected dominating set. In [AWF02], the selection is based on the node identifiers. In [Wu03], a criterion on the connectivity amongst neighbors is added to reduce even further the number of nodes remaining in the final connected dominating set.

13.1.4 Robust backbone algorithms

In order to improve the stability of the generated backbone, as well as improving the robustness by introducing redundancy of the paths existing through the backbone, algorithms have been designed to create k, m-connected-dominating-sets, or k, m-CDS, where:

- k denotes the redundancy of the connectivity in the backbone, meaning that $k-1$ nodes may fail, the backbone would remain connected;
- m is the vertex domination redundancy, meaning that each vertex $u \in V \setminus \mathcal{D}$ is connected to m vertices $v_1, ..., v_m \in \mathcal{D}$ by an edge.

Those solutions are particularly interesting as, with limited increase in the number of nodes composing the backbone and in the complexity, they allow to achieve better quality of service and fault-tolerance in the communications using the backbone structure.

13.1.4.1 Distributed approaches

Decentralized algorithms have also been proposed to solve some instances of the general k, m-CDS problem in [DW05, WWTL07, WL08].

Dai *et al.* proposed three localized $k = m$-CDS algorithms, where $k = m$. The first one is a probabilistic approach which requires the network size and the node density to compute a probability p_k of a node k to be in the backbone. The second one maintains a fixed node degree in the backbone and also requires network size and density. The third one is a deterministic approach which is an extension from the coverage condition introduced by the same authors to construct $1, 1$- CDS[DW05]. This approach checks if there are k independent paths composed of high priority nodes between each pair of neighbors.

In [WWTL07], the authors proposed a distributed version of their original algorithms CGA called DDA. However, DDA requires a huge number of messages, which is a major drawback for a real application in ad hoc networks.

In [WL08], a distributed algorithm, LDA, is proposed to overcome the problems of DDA. However, it requires the maximal node degree to be constant and it uses CDS-BD-D[KWYLD07], which implies the election of a root node in the network.

13.1.4.2 Blackbone and Blackbone2

In [SBA09, SDBA09], *Schleich et al.* introduce the Blackbone2 algorithm which partitions the graph into nodes of two different colors; black nodes, which are the nodes that form the backbone and white nodes, which do not belong to the backbone. It allows to create a 2-connected, m-dominating set. The advantage here is that each node requires only two-hop neighborhood knowledge of the graph in order to perform the needed computations.

The blackbone2 algorithm uses a pruning and a construction rule that can be applied independently to ensure the correctness of the generated structure.

The pruning rule checks if the subgraph induced by the neighboring black nodes and the edges connecting them is a m-dominating set and if it is the case, it checks if the graph induced by the useful black nodes, i.e. essential for the m-domination, is still biconnected. When these two properties are met, a node can get out of the backbone.

The construction rule is the opposite of the pruning rule. A node has to enter the backbone if its 1-hop nodes are not m-dominated or if the graph induced by the useful black nodes is not biconnected.

The blackbone2 algorithm has a very low time complexity (in $O(\Delta^2)$) and does not require extra messaging ($O(1)$ in message complexity, $O(\Delta)$ in bandwidth usage) which make it a very lightweight solution. Besides, as no random factor is part of the algorithm, it is also deterministic. Finally, this algorithm also shows very encouraging results for both quality of the solutions and convergence time.

13.2 Robust virtual backbone creation using communities

As each community built using SAND/SHARC regroups stable links, a virtual backbone established using only links internal to a community will more likely be resilient to changes due to network dynamics.

13.2.1 Proposals for improvements

Therefore, in order to take communities into account the original Blackbone2 algorithm[SBA09] has been altered to compute only for nodes inside the same community, creating Blackbone2C. This implementation gave a second baseline, in addition to the one obtained by applying a plain Blackbone2 instance, to assess robustness of the generated topology management structure.

We observed that the backbone created using Blackbone2C was connecting several communities but the connections seemed random at best. However, as a natural application of the concepts presented in chapter 8, two communities are connected by a backbone link if bridge nodes of two neighboring communities are part of this backbone.

The paper proposing the Blackbone2 algorithm [SBA09] contained some optimization proposals that tried to increase the stability of the backbone by reducing the blinking of nodes and fixing some nodes. We therefore took this path to design a similar approaches, using the community-related information to improve the backbone robustness.

13.2.1.1 B2CB

In order for the virtual backbone to be connected, it is sufficient to merge the different structures, one per community, into a single component. A simple approach is to fix bridge nodes as being part of the final backbone. Therefore, in the B2CB algorithm Blackbone2C is used to compute several backbone fragments, and then backbone adherence for bridge nodes is forced, which ensured network-wide connectivity of the final virtual backbone.

This scheme will artificially augment the size of the backbone. If we consider the nature of nodes we can argue that if communities are densely populated then it is likely that some neighbors of a bridge nodes are also bridge nodes. Furthermore, depending on the size and the spatial layout of communities, these neighboring bridge nodes might connect the same external communities. However, this scheme creates redundancy in regard to inter-community connection, which can be a desired property. This topic will be discussed in the next section.

13.2.1.2 B2CO

This implementation fixes the originator node instead of bridge nodes but works, in all other aspects, identical to B2CB. The repercussions of this property will be discussed when evaluating the simulation results.

From fixing the originator, the intended goal was to diminish the impact of fixing nodes on the backbone size while getting good results for the stability since a highly connected node, the originator, is forced to be in the backbone. We also hoped that this would improve convergence time, again improving stability.

13.2.1.3 B2CBO

We propose another implementation which combines the fixing of bridges and originator from B2CB and B2CO respectively. Combining both schemes unfortunately also means combining the related disadvantages. The problem of the artificially increased backbone size may hence become even worse.

For small communities we might also face the problem that the scheme forces the majority of the communities into the backbone which is suboptimal. However, the increase in stability might be worth the increase in size for larger communities. Fixing more nodes could also mean a faster convergence time. The simulations will confirm or infirm the merit of this implementation.

13.2.1.4 B2CBOT

The previous implementations mentioned the problem of the artificial increase of the backbone size when fixing bridge and originator nodes. B2CBOT applies heuristics in order to lessen the problem or maybe even solve it by fixing the bridge and originator nodes only if a certain condition is met.

The idea is to fix bridge and originator nodes at the start of the computation then the backbone will become connected and should converge to a stable state. Once this state is reached, we hope that releasing the forced network adherence will not lead to a domino effect that will see the backbone nodes toggle to another stable state, probably splitting the backbone into several fragments.

Since we are using only local knowledge in a two-hop neighborhood, the notion of *stable state* cannot be modeled by a global view on the network. The decision must be made locally. To do so we observe the number of one-hop intra-community neighbors that are part of the backbone. If that number at least equals a given threshold value then the bridge or originator node will no longer be forced into the backbone. This implementation will show if this scheme holds any merit. If this is the case we will then refine the notion of threshold and investigate the mater further.

13.2.1.5 COMBO

The algorithm that emerged after having simulated and analyzed is the COMmunity induced backBOne algorithm or COMBO for short. This implementation uses Blackbone2C for intra-community backbone establishment and a threshold scheme with fixed bridges only. The analysis is given in the next section when we detail the simulations and results. Our analysis showed that when fixing the originator, the stability measure fluctuated and was quite low. This was due to SAND/SHARC which applies a random walk scheme to the originator in order to find the node with the highest community score.

With the fixing of the originator this meant that we would induce several changes per computation of the community allocation algorithm. This increase in instability could not be observed in SHARC algorithms using fixed originators such that we were able to pinpoint the problem. It was therefore decided to fix bridge nodes only since the approach showed better results than fixing only the originator, mainly due to reasons already given.

The threshold scheme that we applied in B2CBOT did see some interesting results such that we decided to implement the scheme in our distilled solution. Releasing the locks worked well to reduce the backbone size while keeping the coverage at the same level and without negatively impacting the stability.

13.2.2 Experiments setup

In this section we will present the details and results of the performed experiments. We will also talk about the metrics used to evaluate our results and we will give a detailed analysis of the obtained data.

13.2.2.1 Scenarios

In order to test our ideas we decided to use the two human nature-inspired scenarios that were presented for analysis of community detection mechanisms over dynamic graphs in subsubsection 9.1.3.1.

Before starting to experiment with our proposed algorithms we computed 100 runs of the normal Blackbone2 algorithm with a given seed in order to establish a baseline to which we could compare our future results. After the baseline was computed, we performed 100 runs for each proposed algorithm using the same underlying seed in order to have comparable, statistically relevant data sets.

Since in real life we can not force an order of computation, we also associate to each one of those 100 runs a random seed to shuffle the set of nodes and thereby induce a random computation order for the nodes. This provides a more realistic behavior. It is however this random order that may lead to quirks in the observed computation traces. Depending on the order we might see some domino effects or blinking effects that are not reproducible since they only showed as a result of some rare random order. However, having such events is in our opinion paramount to have relevant data that could be used to compare real life applications and explain some of the behavior that would not have been observable without a random processing order.

13.2.2.2 Quality measures

We will now have a look at the metrics that we will use in order to evaluate the pertinence of our solutions. These measures are mostly identical to those proposed for evaluating the optimizations of the Blackbone2 algorithm in [SDBA09]. Other interesting measures like the diameter of the backbone compared to that of the network or the convergence time were not measured as we concentrated on core measures first.

Size: The size of the backbone is an important metric. We want the backbone to contain as less nodes as possible while still covering as many nodes as possible and keeping a decent stability. In order to be able to compare the efficiency of the algorithm independently of the total number of nodes in the graph we will express this size in percentage of nodes that are member of the backbone in relation to the total number of nodes.

Availability: The notion of availability is also presented in [SDBA09]. It is the ratio between the connected components of our graph and the connected components of the backbone induced subgraph.

Isolated Components: The isolated components measure is the number of nodes that are not connected to the backbone.

Stability: The stability is a measure that indicates how often in average a node changes its state per second. A change of state is performed each time a node enters or leaves the backbone.

13.2.3 Results

In this subsection, we will discuss and interpret the results obtained from our experiments. In a first phase we will see how far the chosen community detection algorithm affects the results and in a second part we will see what are the concrete results obtained by each algorithm and what we can learn about them.

13.2.3.1 General observations

It is important to mention here that, independently of the algorithms and of the considered scenario, the obtained results for the availability are always above 0.97 and the results for the isolated components are always under 0.0001. Therefore we will only focus on the size and on the stability of the backbone for the rest of this section.

13.2.3.2 Interpretation

We will now have a look at the results obtained by the different algorithms.

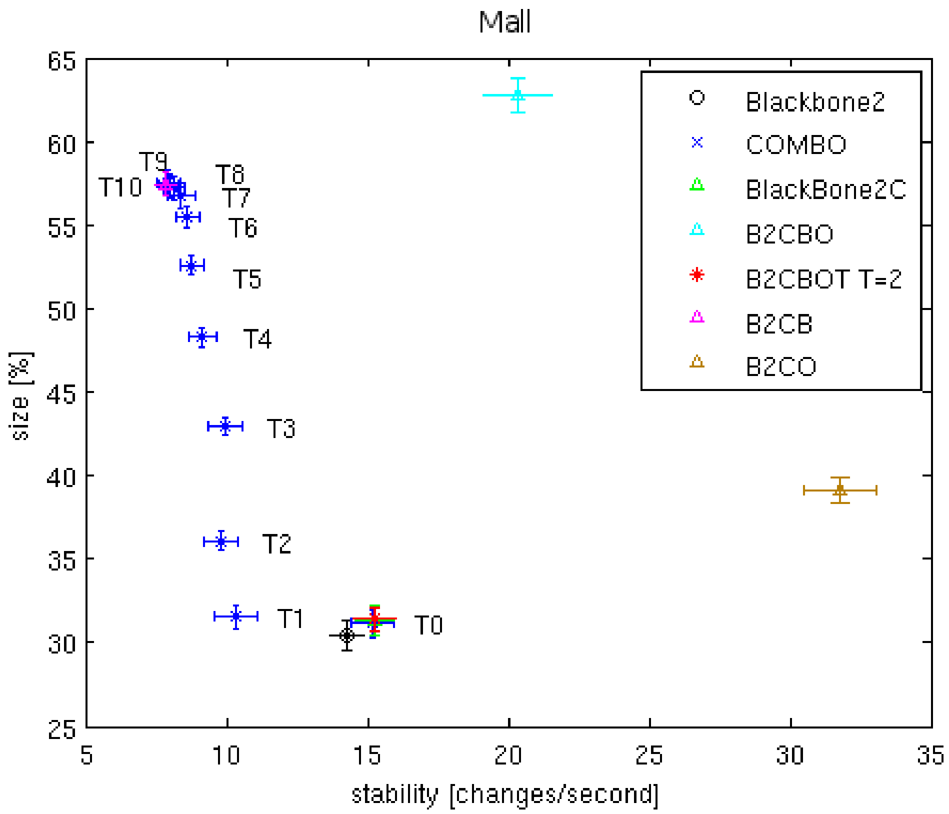


Figure 13.1: Algorithm performance using SAND/SHARC in the mall scenario

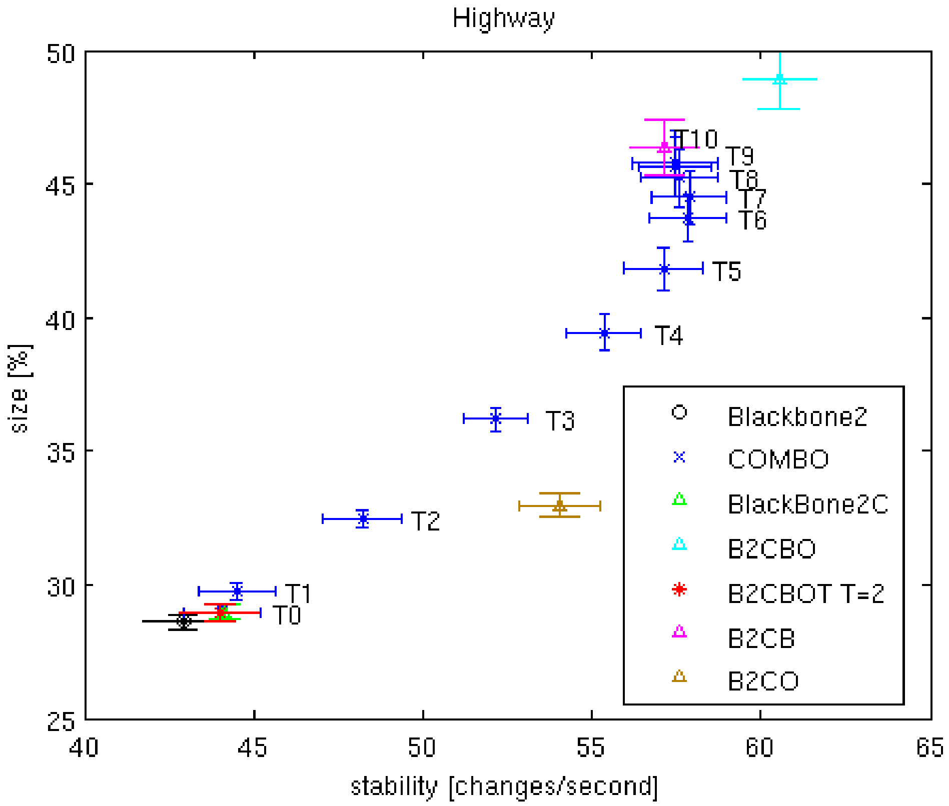


Figure 13.2: Algorithm performance using SAND/SHARC in the highway scenario

Blackbone2C: From Figure 13.1 and Figure 13.2 we see that the Blackbone2C algorithm is slightly worse than Blackbone2 for both highway and mall scenarios. The behavior is not surprising as an algorithm that takes decisions based on a local set of nodes might not connect to the rest of the graph.

This is due to the fact that bridge nodes are likely to change community and each time such a community change occurs the Blackbone2C needs to perform a re-computation of the backbone even if this would not be needed if the algorithm would have taken in consideration the entire graph.

B2CB: For the highway scenario this algorithm performs poorly. This is due to the fact that in high mobility scenarios the network topology changes a lot and the set of bridge nodes changes quickly. However, for the mall scenario the stability is two times better which is due to the fact that here the algorithm has enough time to converge to a stable solution. We also observe that the size of the backbone nearly doubles because we keep every bridge in the backbone even if it is not needed.

B2CO and B2CBO: These algorithms perform poorly for the same reason. Fixing the originator gives us bad results in any scenario. This poor performance is due to the fact that the originator nodes constantly change since the SAND/SHARC-algorithm assigns the role of originator to the node that is the most highly connected to the community. These constant role-switches induce more changes to the backbone which in turn translates in more instability. Due to this observation we chose to drop the idea of fixing the originator as it resulted, in our experiments, in a decrease in performance. Since our experiments only used relatively small community structures these scheme might need to be revisited when dealing with larger communities.

B2CBOT: For this algorithm the results shown in Figure 13.1 and Figure 13.2 are worse than the original Blackbone2 algorithm. The main reason is the same as that explained in the analysis of B2CO and B2CBO. However, the results for this algorithm are similar or better than the simple fixing of bridges or the fixing of the originator. The only explanation is that the threshold value is responsible for the improvement. This observation was the reason why we opted to pursue the threshold scheme.

COMBO: From the algorithms we presented previously and their respective analysis we can make the following observations:

- Fixing the originator is a poor choice. The results obtained by the algorithms that fix the originator are worse than the ones obtained by fixing bridges in both size and stability;
- The threshold scheme improved the results of the algorithms it was applied to. Their poor performance might be linked to the fixing of the originator nodes and it might be worthwhile to investigate the impact of the threshold scheme while not fixing the originator.

These observations lead us to the creation of the COMBO which fixes only bridges and uses the threshold mechanism.

As we can see in both Figure 13.1 and Figure 13.2 this algorithm is the one that performs best among our new algorithms. In the case of the mall scenario we have a stability gain of approximately 30% when using a threshold of 1, whereas the backbone size increases by less than 5%. This gain of stability is due to the fact that the communities of this scenario evolve only slowly and therefore the bridge nodes are more stable and the Blackbone2 algorithm computed inside the communities has time to converge to a stable solution.

On the other hand when we run our algorithm on the highway scenario, the mobility is so high that nodes never stay bridges for long due to the quick topology changes which also keep the Blackbone2C algorithm from converging to a stable solution. This translates into a loss of stability for any threshold value since nodes are fixed according to their roles which in turn changes with the speed the network layout changes. The higher our threshold the longer we fix the nodes which translates, for the reasons given above, to an ever decreasing stability of the backbone.

The increase in size can be explained by the number of nodes that are fixed and the lack of time for the Blackbone2C algorithm to converge. This translates into the threshold value only being satisfied by fixing of nodes and nearly never by the construction of the intra-community backbone infrastructure such that a decrease in size observed for stabler community assignments can never be achieved.

13.3 Highlight

From this chapter, and particularly the study developed in section 13.2, we can conclude that:

- Using community structures may **ease the construction of a backbone structure**, make it **more robust** and **converge faster**;
- The backbone construction and community assignment are **heavily coupled**;
- The performance of the backbone in terms of stability will **depend solely on the community attribution** as will the robustness of the backbone structure;
- It is therefore imperative that the **community attribution algorithms is carefully chosen** in order to avoid undesired behavior.

Contents

Chapter 14

Communities and enhanced broadcasting process

"The advancement and diffusion of knowledge is the only guardian of true liberty."
James Madison.

Broadcasting is considered as one of the most important low-level operation in networking as many applications and even other protocols rely on its service. Generally, the problem associated with dissemination algorithms is the broadcast storm problem (see definition 4.5).

However, when dealing with mobile ad hoc networks, additional specific drawbacks must be considered (battery life, mobility of devices, limited transmission range, etc.). Thus, the main problem in broadcasting is not only reducing the number of forwarding but also trying to overcome all these undesirable aspects. Usually, dissemination algorithms include some mechanisms for preventing all nodes rebroadcasting the message in order to avoid the already mentioned broadcast storm problem. The efficiency of those mechanisms generally relies on the local knowledge of the nodes but also depends on many aspects such as the node density, the mobility, etc.

In this chapter, we propose to add some community knowledge to the nodes in order to enhance the forwarding decision, and therefore, improve the performance of the broadcasting process. We are proposing a new energy-efficient broadcasting algorithm, CEDBBT (Community Enhanced Distance-Based Broadcasting with Bridge Threshold), that is using community knowledge to enhance the performance of the dissemination process. Additionally, a hybrid multi-objective algorithm, CellDE, is used to optimize the configuration parameters of this broadcasting protocol. The optimization is done by maximizing the coverage achieved and minimizing at the same time both energy consumption and broadcast time.

14.1 Energy-efficient broadcasting

This study uses the contributions of various domains, such as energy aware broadcasting algorithms and decentralized community detection algorithms. Community detection algorithms were presented in subsection 3.3.3, chapter 8 and chapter 9. In this section, we will hence focus on energy-efficient broadcasting algorithms, that are part of the topology control techniques introduced in section 11.3.

14.1.1 State of the art

As we mentioned earlier in this document, the energy consumption in mobile ad hoc networks is a hot topic, since devices can run out of battery, provoking the network degradation. Some of the solutions that have been proposed for saving energy in broadcasting algorithms dealing with ad hoc networks are mentioned below.

In [GC07], it was shown that a variable transmission range can improve the network capacity while saving energy, and that there exists an optimum value, not necessarily minimum, which maximizes the capacity available of the nodes in presence of mobility. Also, an approach to estimate the local density using an analytical model is used in [LNM03] to set the transmission range according to this estimation. In [RB10b], the transmission power varies in order to reach the furthest one-hop neighbor, reducing not only the energy consumption but also the number of collisions without degrading the network connectivity. An adaptive version of this protocol was presented in [RB10a].

Extensive studies on energy-efficient algorithms for finding the minimum-energy broadcast tree (MEBT) have been proposed like [CHE02, CHE05], and also in [LBXW09] where a shared multicast tree built in a distributed fashion with minimum energy is presented. The transmission power is either fixed or adjustable.

Considering two-hop knowledge, we can find [CSS03] in which a restricted neighborhood graph is constructed, or [CFK03], where the source node examines whether it is worth excluding the furthest neighbor and using the new furthest node as a relay to successfully cover the two hop neighborhood.

In vehicular ad hoc networks it is also a tendency to adjust the transmission range. In [RR05] nodes exchange periodically beacons containing information about the path loss. Neighbors are sorted according to the average path loss and when a broadcast message is received, the node checks the transmission power necessary to reach a targeted number of nodes.

14.1.2 Energy-aware distance-based broadcasting

For our study, we will rely on the Energy-aware Distance-based Broadcasting algorithm, EDB. The distance-based approach was first introduced in [TNCS02], but extensive studies have been done in this area[LSK09, KJQ08]. Literature reveals many interesting works in energy-aware broadcasting algorithms as the battery life in mobile ad hoc networks is a constraint. For example, in [ZyZzMz07] an improved version of DB is presented adding counter based features to the border-aware scheme. It also considers the remaining energy of the nodes.

EDB is a distance-based broadcasting algorithm that reduces the transmission power when broadcasting in order to reach the furthest one-hop neighbor (see Figure 14.1). Reducing the transmission power not only reduces the energy consumption but also decreases the interference level of devices around. EDB is using a cross-layer design to inform the upper layers about the reception power of the received messages. Thus, in EDB, the threshold (called *neighbors threshold*) is not in terms of distance (m), but power (dBm). Additionally, we assume the loss a packet suffers for traversing in one direction is the same it will suffer for traversing in the opposite direction. So that, by sending all nodes the beacons with the default transmission power (and keeping track of the reception power), a node is able to estimate the power needed to reach a specific node (more details can be found in [RB10b]).

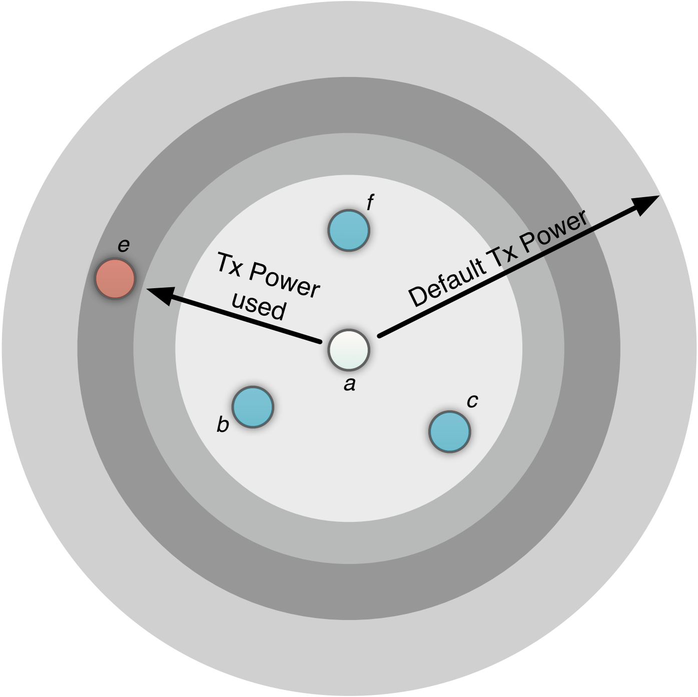


Figure 14.1: The mechanism of EDB

Figure 14.1 represents the behavior of EDB. When node a forwards a broadcasting message, the transmission power is reduced. Instead of using the default transmission power, node a sets the new transmission power as the one needed to reach the furthest node, in this case node e. Once node a sent the broadcasting message, none of nodes b, c and f will rebroadcast it, as they are not located in the forwarding area (grey area close to the limit of the transmission range). However, node e sets a random timer because it is a candidate of forwarding the message. If the timer finishes and no copy of the same message is heard, the message is forwarded. Otherwise, the new reception power is computed and if it is close to the new source node the message is dropped.

14.2 Enhanced broadcasting process using communities

In this section, we first present an efficient way to detect sudden drops of density based on community information, which will improve the EDB behavior in networks with non-homogenous density. In a second time, we provide the details of our algorithmic contribution and evaluate it, before introducing further optimizations.

14.2.1 Problem description

In most of the broadcasting approaches the designer tries to find a trade-off between the performance of the broadcasting algorithm, and the use of the network resources. On one hand, if only the broadcasting performance is considered under simplistic assumptions, it may lead to the broadcast storm problem[TNCS02]. On the other hand, considering only the network resources will end up in a poor performance. Even algorithm that are very good for certain networks may not work that good in different networks.

In the case of EDB, we checked that when the network is sparse or when there is a drop in the local density, the performance of the algorithm can be easily increased. In Figure 14.2, we can see one example configuration where the broadcasting message does not cover a reasonable percentage of the network even though there is sufficient connectivity for a complete coverage. All the nodes in the upper part of the network (black nodes) did not receive the message, while blue nodes received the message, green nodes received and rebroadcast it and red nodes received it but deferred retransmission due to the EDB process.

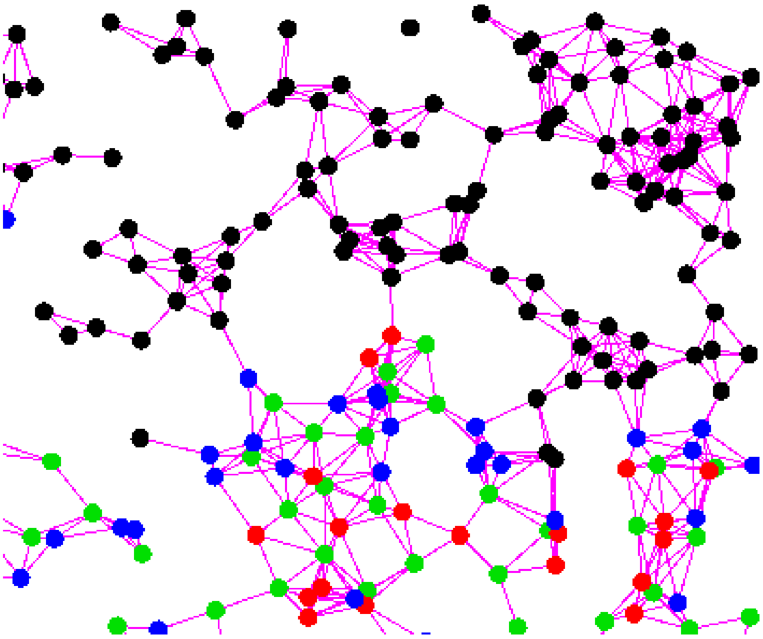


Figure 14.2: EDB encountering a sudden drop in the local density (dynamic graph)

The goal of our study is to find a solution that could help to improve the broadcasting process, at reasonable cost, using only local knowledge.

14.2.2 Proposed solution

We previously stated that sudden drop of density may induce a bad coverage. To overcome this problem, the idea would be to detect these drops of densities and add a mechanism to avoid the stop of the broadcasting process. This can be easily achieved with additional knowledge about the environment such as community information.

14.2.2.1 Rationale

By definition, the boundaries of communities are characterized by sudden drop of local density. The nodes composing these boundaries, which are essential to avoid the assessed problem, will be referred to as *bridge node* (see definition 14.1).

Definition 14.1 (Bridge node)
A *bridge node* u, belonging to community $C(u)$, is a node having at least one of its one-hop neighbor v such that $C(v) \neq C(u)$.

In the remaining of this paper, we used the SAND/SHARC community detection algorithm. Despite its two-hop knowledge requirement, this algorithm remains computer efficient and the quality of the detected communities is very high.

14.2.2.2 Contribution: CEDBBT

Detecting bridge nodes is an important first step. However, depending on the topology, these nodes may represent a non-negligible percentage of the whole network. As a consequence, forcing all of them to broadcast would induce a huge increase of messages, which obviously is not suitable. For comparison purposes, in the remaining of this paper, the naive approach will be referred as CEDB. Many heuristic choices were tested during this work to prioritize some bridge nodes and CEDBBT (Community Enhanced Distance-Based Broadcasting with Bridge Threshold) obtained the best results.

The main idea of CEDBBT is to apply in a first time a specific heuristic algorithm for the bridge nodes, and if the decision is to not relay the message, then the original EDB algorithm is triggered. By doing so, the CEDBBT algorithm will induce the emission of at least the same amount of message than EDB. To limit the eligible number of relaying bridge node, timers have been introduced.

This additional timing mechanism prevents bridges to forward in case they get the same message more than once from neighboring bridge nodes belonging to their own community. These timers, which are similar in function to those used before in the EDB algorithm, limit the number of times CEDBBT tries to forward when encountering a bridge node. Since the number of bridge nodes in denser parts of the network is quite high, bridge nodes often appear in parallel, bridging the same gap, and are often even located in the same one-hop neighborhood. This led to the introduction of the aforementioned timers to prune some of the redundant bridges.

Based on the same scenario, comparing the results of CEDBBT in Figure Figure 14.3 with the results of EDB in Figure 14.2, we see that the coverage is augmented for this part of the network. Furthermore the bridge nodes did forward where EDB failed to do so. The pseudo-code of CEDBBT is presented in Algorithm 4.

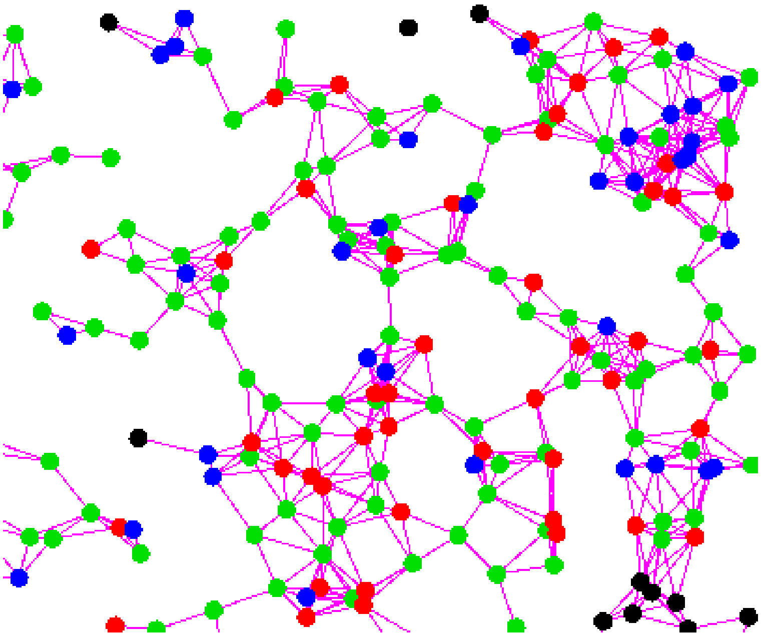


Figure 14.3: CEDBBT achieving high coverage on the same dynamic graph as in Figure 14.2

14.2.3 Experimental setup

In this section we present the different choices we made for our experiment, i.e. the simulation environment and parameters, the mobility model and the propagation model.

14.2.3.1 Simulation environment and parameters

For simulating different broadcasting schemes as well as the algorithm presented in subsubsection 14.2.2.2, a custom network simulation based on GraphStream[Gra] was used. Each algorithm was tested on densities varying between 100 and 700 nodes within a 4km^2 area.

To get comparable results for each density, each algorithm was executed on the same 100 predefined graphs for each density, and starting the broadcast process on a set of predefined nodes. All the tested values of the algorithm parameters can be found in Table 14.1.

Parameter	**Value**
Number of devices	[100..700]
Area	2000 × 2000 m (4 km^2)
Transmission power	16.02 dBm
$end\_Threshold$	−95 dBM
$borders\_Threshold$	−90 dBM
$margin\_Threshold$	0.5 dBm
EDB & Bridge timer delays	[0..1] s
Direction change	every 20 s

Table 14.1: Simulation Parameters for CEDBBT

Algorithm 4 Pseudocode of CEDBBT

Data: m: the incoming broadcast message
Data: r: the node receiving the broadcast message
Data: s: the node that sent m
Data: p: the received signal strength of m sent by s
Data: borders_Threshold: the signal strength threshold

```
if $m$ is received for the first time then
    calculate p
    update pmin
    if $pmin > borders\_Threshold$ then
        if $r$ is bridgeNode then
            $waiting$ = true
            wait random time
        else
            $r \rightarrow drop\ message$ m
        end if
    else
        $waiting$ = true
        wait random time
    end if
else if $waiting$ then
    calculate $p$
    if $p > pmin$ then
        update $pmin$
    end if
    if bridgeNode(r) & sameCommunity($s$, $r$) then
        $msgFromOwnCommunity$ = true
    end if
end if
if bridgeNode(r) & $msgFromOwnCommunity$ = true then
    estimate power to reach furthest neighbor
    transmit $m$
else if $pmin > borders\_Threshold$ then
    $r \rightarrow drop\ message$ m
else
    estimate power to reach furthest neighbor
    transmit $m$
end if
$waiting$ = false
```

14.2.3.2 Mobility model

For the mobility model, we used the Brownian motion mobility model [GAN06]. In this model, devices move with a randomly chosen speed and direction during a fixed amount of time (20s). If a device hits the borders of the simulation area, it rebounds with a reflexive angle and speed. The speed chosen for the simulation varies in $[0..2]$ m/s.The area to which the devices were confined was a space of 4km^2 in a two-dimensional plane without any obstructions (except for the surrounding walls) blocking any movement once the device was set in motion.

14.2.3.3 Propagation model

The propagation model is based on the Friis equation (see Equation 14.1) which is a good compromise between computation costs and approximation of real 2.4 GHz WiFi transmissions. The maximum transmission range is limited to about 150m. The exact specifications can be found in [LH06].

This equation allows to compute the amount of received power emitted by an omni-directional antenna, working under the assumption of idealized conditions. P_r is the available power at the receive antenna terminals, P_t is the power delivered to the transmit antenna. G_t and G_r are the gains of the transmitting and receiving antennas, respectively. Finally λ is the wavelength and R the distance between both antennas.

$$\frac{P_r}{P_t} = G_t G_r \left(\frac{\lambda}{4\pi R} \right)^2 \tag{14.1}$$

14.2.4 Numerical results

In this section, we present the results obtained for the proposed protocol CEDBBT, as well as the upper bound and the original EDB. In order to have reliable results, each algorithm was executed in 100 different graphs, and for each different seed the broadcasting process was started from 8 different nodes. The presented results are the average of all the obtained values. Moreover, the algorithms were tested in different network densities, from 100 to 700 devices. For higher densities, EDB is almost covering the full network, therefore, the use of the community knowledge is not needed.

Figure 14.4 shows the coverage achieved by the dissemination process for each of the considered protocols. We can see that, both CEDB and CEDBBT highly outperform EDB in terms of coverage from 300 to 500 devices. The highest improvement is obtained for the 400 network devices, where CEDB increases the coverage achieved by EDB by 50.46% and CEDBBT by 42.109%. Note that all algorithms perform poorly at low densities (under 300 devices), as the sparse connectivity physically prevents any network-wide broadcast process.

As we mentioned before, we can consider CEDB is an upper bound for the coverage achieved, since all the bridge nodes rebroadcast the broadcasting message. However, it will be also the one with higher number of forwarding nodes.

The percentage of nodes that rebroadcast the message is shown in Figure 14.5. EDB is always better in terms of number of re-broadcasts. In both cases, EDB and CEDBBT, this number decreases as the network density increases. This is the proper behavior as we can benefit from high density to better disseminate the message.

As CEDB is a very naive approach, the algorithm increases as the number of bridge nodes also does. Thus, the highest difference between EDB and CEDB is found for 700 devices, where CEDB uses 40.45% more devices than EDB. We should mention that at this density the average number of bridge nodes is higher than 71%. In the case of CEDBBT, the highest difference with EDB is found for 400 devices and is 10.22%.

We also measured the energy used by the broadcasting process. In this case, Table 14.2 presents

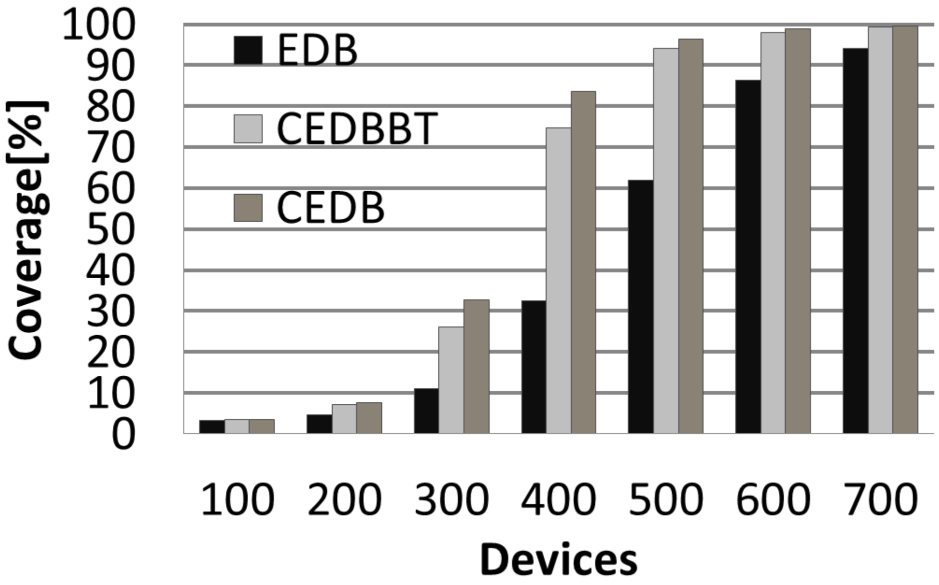


Figure 14.4: Overall coverage achieved by EDB, CEDB (Pure Bridge Broadcasting) and CEDBBT

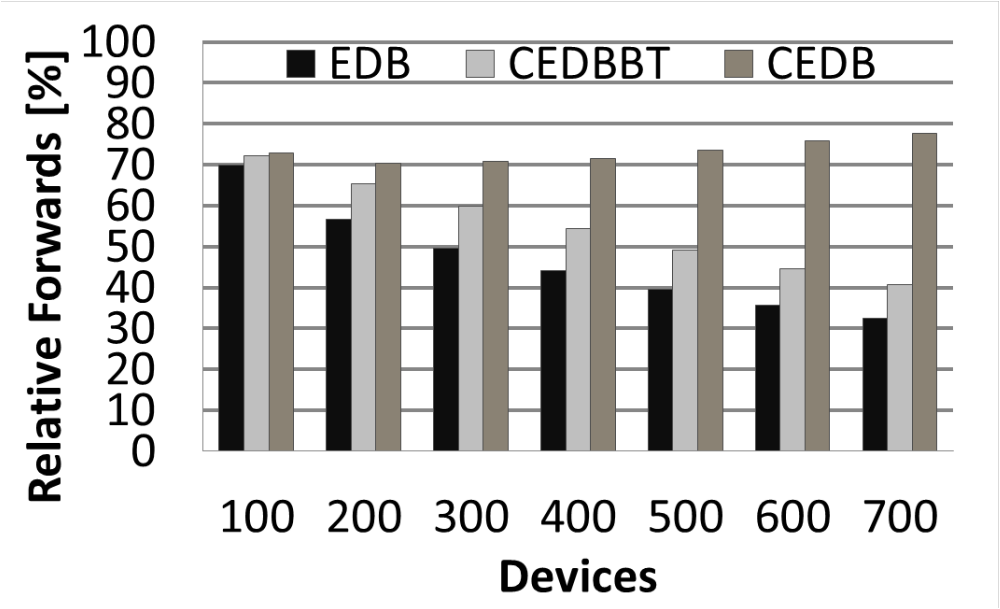


Figure 14.5: Relative forwarding achieved by EDB, CEDB (Pure Bridge Broadcasting) and CEDBBT

the energy used per forwarding. As it was already checked in [RB10c], generally the higher the number of re-broadcast, the lower the energy per forwarding (in not very dense networks). This is due to the fact that the higher the number of nodes forwarding in a specific area, the higher the probability of using nodes whose furthest neighbor is closer, and therefore, in average, the energy per forwarding decreases. However, the values of EDB and CEDBBT are very similar.

In Table 14.3, the results obtained for the broadcast time are presented. It was also tested in [RB10c], that broadcast time was decreasing as the network density was increasing. The reason is that the probability of having the furthest node closer to the limit transmission range

Densities	**EDB**	**CEDB**	**CEDBBT**
100	36.6138 ±4.4550	36.3150 ±4.6312	36.3450 ±4.5987
200	35.4937 ±35.4937	34.4687 ±4.0375	34.5100 ±3.9238
300	35.3450 ±2.9125	34.3950 ±2.3650	34.4275 ±2.4912
400	35.8337 ±1.9862	35.1650 ±1.1637	35.1450 ±1.2687
500	36.3175 ±1.3037	36.0550 ±0.6313	35.9263 ±0.6675
600	36.8500 ±1.0137	36.7888 ±0.4375	36.6388 ±0.4762
700	37.4062 ±0.5150	37.3750 ±0.2512	37.2262 ±0.3188

Table 14.2: Average energy per forwarding in mW

in a dense network is higher, and therefore, the dissemination is faster (many forwarding operations are also suppressed). For EDB, the broadcast time started to decrease at around 600 devices network. The fastest algorithm is EDB for the first 4 network configurations, then the naive approach is taking less time to broadcast the message. For networks with 300 and 400 devices EDB covers less network, thus, the broadcast time must be lower. However, for 500 devices and more, the number of forwarding operations of CEDB is much higher, whereas the difference in coverage is not that high. That implies, the covered network is more or less the same but using many more re-broadcast. Thus, again, the probability of using the furthest nodes is higher.

Densities	**EDB**	**CEDB**	**CEDBBT**
100	0.6942 ±0.7935	0.7315 ±0.8005	0.7576 ±0.8765
200	1.6752 ±1.6191	2.445 ±2.2663	2.5392 ±2.3902
300	3.8419 ±3.3435	7.4305 ±5.2396	7.3926 ±5.4848
400	8.0255 ±5.7295	11.500 ±4.2540	13.320 ±5.3778
500	11.418 ±5.7118	9.228 ±2.0829	11.744 ±2.7945
600	11.761 ±3.7163	7.7111 ±1.3269	10.074 ±1.9180
700	10.762 ±2.6821	6.7276 ±1.0317	8.9461 ±1.4475

Table 14.3: Average broadcast time in seconds

14.2.5 Optimization

As we mentioned before CEDBBT is based on EDB, and therefore, it has the same parameters:

1. **borders_Threshold** is the value considered to decide if a node is a candidate of forwarding or not. If the reception power is higher than the borders_Threshold the message is dropped;
2. **margin_Threshold** is a value added to the calculated new transmission power, to compensate for some possible errors in the estimate;
3. **minimum_Delay** is the minimum possible value that can be assigned to the random delay;
4. **maximum_Delay** is the maximum value that can be assigned to the random delay.

These parameters were already optimized for EDB in [RDV$^+$ne].

In this section, we try to improve the performance of the proposed broadcasting algorithm by using some multi-objective optimization. For that, we need first to specify the goals we want to achieve during the broadcasting process. The objectives considered in this work are:

- coverage: the number of nodes covered by the dissemination process;
- energy consumption: the total energy used by the broadcasting;
- broadcast time: the total time needed to spread the message.

Obviously, we would like to maximize the coverage consumption and minimize both, the used energy and the broadcast time.

14.2.5.1 Optimization algorithm

Metaheuristics [GP10] are iterative stochastic optimization tools that are able to provide good solutions in reasonable time for highly complex optimization problems. Generally, metaheuristics make no assumptions about the problem to solve, so they are generic tools that only need an adequacy, or fitness function, to guide the search towards better solutions.

Evolutionary Algorithms (EAs)[BFM97] are a popular family of metaheuristics. One important feature of EAs is that they work with several candidate solutions at the same time, therefore simultaneously exploring several different regions of the search space. This allows EAs to better explore the search space and reducing the probabilities of getting stuck in local optima with respect to other metaheuristics families.

In this analysis, our problem is defined as having three objectives, since to optimize the protocol performance we need to maximize its coverage, and minimize the energy used and the broadcasting time. We use CellDE to solve it, because it is a highly competitive evolutionary algorithm for multi-objective optimization that has proven to perform specially well for three-objectives problems with continuous variables [DNLA08], as it is the case of our problem.

The use of CellDE will allow us to find a finite set of non-dominated configurations (none is better than the others for the three objectives) for our CEDBBT protocol containing the solutions with the best possible trade-off among the three objectives. This will help in understanding the impact of the different parameters on the behavior of the protocol, as well as choosing the best possible solution for our particular scenario.

14.2.5.2 CellDE Configuration

We are using the CellDE algorithm contained in the jMetal framework [DNA10]. The algorithm configuration we used in our experiments is the one proposed by the authors in [DNLA08]. The stopping condition was set to 10000.

For quantifying the quality of the solution, CellDE calls our simulation environment developed inside the GraphStream framework. When a simulation is finished, GraphStream returns the values obtained for the different objectives using this configuration. In order to have reliable results we are running the broadcasting algorithm in 10 different networks every time, CellDE checks the quality of the solution (this happens 10,000 times).

We choose an interval for each parameter (explained above) in order to find reasonable solutions and limit the search space. These values are shown in Table 14.4. The algorithm originally creates a set of random feasible solutions (values chosen from the intervals shown in Table 14.4), and automatically evolves them to better solutions.

Variable	**Domain**
Minimum delay	[0..1] s
Maximum delay	[0..5] s
$borders\_Threshold$	$[-95..-70]$ dBM
$margin\_Threshold$	[0..3] dBm
$neighbor\_Threshold$	[0..50] dBm

Table 14.4: Domain of the variables to optimize

14.2.5.3 Provided solutions

Table 14.5 and 14.6 show some of the non-dominated solutions being of special interest since their parameters achieve a good results in at least one of the three objectives.

In solution 2. the delay parameters were set to be equal. In a real life scenario, this would encourage the broadcast storm problem since too many message would be sent within a short time-frame. However, this is not considered within the optimization presented in this paper due to the absence of collisions.

Sol.	**Coverage (%)**	**Time (ms)**	**Energy (mW)**
1	96.15	14650	12777
2	75.575	4580	9746
3	81.95	39020	5803
4	82.525	8440	7287.755

Table 14.5: Objective values of the non dominated solutions found

Sol.	**min. delay (s)**	**max. delay (s)**	**borders_Threshold**	**margin_Threshold**
1	0.272	0.400	-78.429	1.920
2	0.1	0.1	-84.370	1.439
3	0.823	1.912	-90.222	0.069
4	0.1	0.104	-92.075	0.213

Table 14.6: Parameter values of non dominated solutions found

14.3 Highlight

In this chapter, we have highlighted the following:

- The EDB broadcasting process has an **important coverage issue** occurring when there is a **sudden drop in local density**;
- We used a **decentralized community detection algorithm** to detect where potential problems may happen and we **proposed an algorithm** based on EDB, namely CEDBBT, which obtains very promising results;
- We **fine-tuned our algorithm** thanks to a state-of-the-art **multi-objective technique** using broadcast time, number of messages, coverage and total used energy;

VI

Conclusion

Contents

Chapter 15

Conclusion and perspectives

> "I propose to beg no question, to shrink from no conclusion,
> but to follow truth wherever it may lead."
> Henry George, Progress and Poverty.

15.1 Conclusions

In this thesis, we studied the characteristics of wireless mobile ad hoc networks (MANETs) in an urban environment and presented a solution to regroup densely and reliably connected users in order to enable interactive and robust device-to-device communication. Our objective was to overcome problems related to lack of link stability and increased delay in delivery, especially in wireless multi-hop paths.

We formally presented our contribution, based on notions from social network analysis, and its applications in both theoretical and practical perspectives.

The contributions of the dissertation to the addressed problem can be summarized as below.

- An introduction to graph theory, and more particularly to social network analysis, in order to model dynamic communication networks as dynamic graphs. This allowed to underline the existence of various network topology models, that have a direct impact on the communication performances. We focused on models that match social and communication networks, like small-world and free-scale networks, and to introduce the notion of communities.
- A presentation of the principle, challenges and application of wireless ad hoc networks. Particularly, we highlighted the advantages coming from their distributed nature, mitigated

by their dynamic characteristics. We also insisted on the fact that such networks can help in offloading traffic from current existing infrastructure networks.

- By presenting efforts in modeling mobile communication networks as dynamic graphs, using a transmission function (or propagation model) and a dynamic location function (or mobility model), we showed how the urban environment and the social incentives behind human mobility make the network topologies presented above appear. This justification for our use of the community structure as base for our contribution was achieved by:
  - An extensive state of the art on mobility models, especially focusing on models that integrate social rationales for human-based mobility;
  - The design and presentation of a social-based human mobility model, especially designed for urban environments, and that uses the notion of user profile to govern macro mobility.
- We presented a set of algorithms for distributed community detection, based on epidemic propagation of community label.
  - SHARC (Sharper Heuristic for Assignment of Robust Communities), intended for static networks, introduces the neighborhood similarity metric that allows sharper and more robust determination of the community structures;
  - SAw-SHARC (Stability Aware SHARC) takes into account the stability of links as weight through a node-centric stability estimator and allows to group users sharing the most reliable links in the same community;
  - SAND/SHARC (Stability And Network Dynamics over SHARC) presents procedures specifically designed to improve operation on dynamic networks: a community organization mechanism using a local optimum favored random walk and a community split mechanism using a constrained propagation of freshness counter.

  SAND/SHARC is, to our knowledge, the first online distributed community detection algorithm suitable for operation on dynamic networks.
- We investigated how the information about community assignment provided by SAND/SHARC can help other distributed operations on wireless ad hoc networks, like virtual or physical topology management or the improvement of energy-aware broadcast.

15.1.1 SHARC

SHARC aims at addressing current shortcomings of the existing distributed algorithms used with this kind of networks. According to results from our extensive simulation campaign, SHARC performs community assignment with a better accuracy than those of equivalent solutions. Besides, the introduction of the neighborhood similarity measure makes the assignment realized by SHARC sharper and more robust to initial condition changes. Finally, SHARC has shown to prevent the community domination effect.

However, SHARC still relies on simple set operations and only uses local information that can be easily obtained through one-hop beaconing. The algorithm runs in $O(\Delta^2)$, where Δ is the network average degree. It is therefore perfectly suitable for deployment on the handheld communicating devices composing urban wireless ad hoc networks.

15.1.2 SAw-SHARC

We have considered the utility of creating communities of densely and reliably connected nodes, for mobile social networks or vehicular ad hoc networks. Such applications require stable connections between a size-limited subset of nodes. However, no existing decentralized algorithm performing community detection was suitable for this particular use case, especially because none of them especially addresses problems related to stability.

We have therefore presented SAw-SHARC, based on SHARC, a distributed community algorithm able to dynamically group users that are densely and reliably connected, thanks to the usage of a node-centric relative link stability estimation.

15.1.3 SAND/SHARC

Considering the network as evolving over time also requires an extension of the sole concept of communities. As components of the graph may appear and disappear, particularly edges, not only shall the communities also evolve over time, but their assignment shall reflect both the high density and the high durability of the links between nodes. In sociological terms, that originally inspired this research field, it relates to consider both the tightness and the duration of relationships within a circle of friends. We therefore proposed to define communities as groups of vertices having high and stable internal connectivity and much sparser and shorter-lived links to the outside of the group.

Under those considerations, we proposed SAND/SHARC, a distributed algorithm able to detect reliably-connected communities in a dynamic environment. SAND/SHARC is built upon SHARC and SAw-SHARC.

But SAND/SHARC introduces particular mechanisms designed for operation on dynamic networks. The constructive mechanism uses node-centric stability estimation in order to favor vertices connected with a highly-weighted edge to be grouped in the same community. When edge weight reflects a stability criterion, this tends to create stable community structures. SAND/SHARC also implements a regeneration mechanism using constrained freshness propagation, able to detect appearing drops of link density within an existing community and generating new structures accordingly.

The decentralized nature of our algorithm, the weak requirements on the network knowledge and its limited computational and memory complexity make SAND/SHARC particularly suited for deployment on devices used in mobile ad hoc networks, like MoSoNets or VANets.

SAND/SHARC also proposes an organization of each community structure, around a special node called originator that tends to be located around the vertices most strongly attached to their current community. Those nodes can be used for many applicative purposes like merging or disseminating information, or act as bridges to infrastructure networks.

15.2 Perspectives

In this work, we focused on the development of a distributed solution, based on topological information only and that will operate on a distributed and unreliable communication system.

This context is therefore highly demanding which may explain why wireless ad hoc networks are not widespread, despite the multiplication of wireless communicating devices and the increase of traffic load on infrastructure networks.

By relieving one or several of the constraints presented above, we can envision the following perspectives to our work:

- The concept of communities can be extended to integrate the mutual interest between users in the community assignment process, as we did with the stability of links. This social similarity metric could be based on the nature of the available content, or the social mobility pattern of the users. This would allow to integrate social considerations beside the already existing topological information.
- Our algorithms could also benefit to the introduction of overlapping communities, allowing each user to belong to several communities at a time.
- It is also our belief that community detection can benefit to communication systems other than wireless ad hoc networks. As currently no handheld wireless device gives an easy access to ad hoc mode, we could investigate how standard wireless systems could benefit from community information. We also think that peer-to-peer applications, even relying on decentralized wired networks like the Internet, could be improved using community detection.

References

[AB02] R. Albert and A. L. Barabasi. Statistical mechanics of complex net- works. *Review of Modern Physics*, 74:47–97, January 2002.

[AJV05] Cedric Adjih, Philippe Jacquet, and Laurent Viennot. Computing connected dominated sets with multipoint relays. *Ad Hoc Sensor Wireless Networks*, 1, 2005.

[AMSK03] S. Ali, A. A. Maciejewski, H. J. Siegel, and Jong-Kook Kim. Definition of a robustness metric for resource allocation. pages 10 pp.+, 2003.

[And03] H. R. Anderson. *Fixed Broadband Wireless System Design*. John Wiley Co., 2003.

[AWF02] Khaled M. Alzoubi, Peng-Jun Wan, and Ophir Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *Mobile Ad Hoc Networking and Computing*, pages 157–164, 2002.

[BA99] A. Barabasi and R. Albert. Emegence of scaling in random networks. *Science*, 286, 1999.

[Bar64] Paul Baran. On distributed communications. Memorandum 1, The RAND Corperation, Santa Moncia, California, 1964.

[BAR07] M. R. Brust, A Andronache, and S Rothkugel. Waca: A hierarchical weighted clustering algorithm optimized for mobile hybrid networks, 2007.

[Bau04] Tim Baugé. ad hoc networking in military scenarios. Whitepaper NET040501-2, Thales Research, 2004.

[BB05] Jim Bagrow and Erik Bollt. A local method for detecting communities. *Phys. Rev. E, 72 046108 (2005)*, 2005.

[BDGO06] Cyrille Bertelle, Antoine Dutot, Frédéric Guinand, and Damien Olivier. Organization Detection Using Emergent Computing. *International Transactions on Systems Science and Applications*, 2(1):61–69, 09 2006.

[BDTC05] Jeremy Blum, Min Ding, Andrew Thaeler, and Xiuzhen Cheng. *Connected Dominating Set in Sensor Networks and MANETs*, page 329. 2005.

[Bel58] Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[Bel62] Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *J. ACM*, 9:61–63, January 1962.

[Ber05] Hugues Bersini. *Des réseaux et des sciences - Biologie, informatique, sociologie : l'omniprésence des réseaux*. Vuibert Informatique, 2005.

[Bet01a] Christian Bettstetter. Mobility modeling in wireless networks: categorization, smooth movement, and border effects. *Mobile Computing and Communications Review*, 5(3):55–66, 2001.

[Bet01b] Christian Bettstetter. Smooth is better than sharp: a random mobility model for simulation of wireless networks. In *MSWiM*, pages 19–27, 2001.

[BFM97] T. Bäck, D.B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. Oxford University Press, 1997.

[BFO96] Giuseppe Bianchi, Luigi Fratta, and Matteo Oliveri. Performance evaluation and enhancement of the csma/ca mac protocol for 802.11 wireless lans. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 1996.

[BGA$^+$08] A. Beach, M. Gartrell, S. Akkala, J. Elston, J. Kelley, K. Nishimoto, B. Ray, S. Razgulin, K. Sundaresan, B. Surendar, M. Terada, and R. Han. WhozThat? evolving an ecosystem for context-aware mobile social networks. *Network, IEEE*, 22(4):50–55, July 2008.

[BHKL06] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 44–54, New York, NY, USA, 2006. ACM.

[BJ02] Steve A. Borbash and Esther H. Jennings. Distributed topology control algorithm for multihop wireless networks. In *World Congress on Computational Intelligence*, 2002.

[BLRS03] Douglas M. Blough, Mauro Leoncini, Giovanni Resta, and Paolo Santi. The k-neigh protocol for symmetric topology control in ad hoc networks. In *Mobile Ad Hoc Networking and Computing*, pages 141–152, 2003.

[BLW77] N. L. Biggs, E. K. Lloyd, and R. J. Wilson. Graph theory 1736-1936. *Historia Mathematica*, 4:480–481, 1977.

[BM08] Adrian Bondy and U.S.R. Murty. *Graph Theory (3rd corrected printing)*. Graduate Texts in Mathematics. Springer-Verlag, 2008.

[BWS06] Tanya Y. Berger-Wolf and Jared Saia. A framework for analysis of dynamic social networks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 523–528, New York, NY, USA, 2006. ACM.

[BXFJ03] B. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2):267–285, April 2003.

[Cas06] Arnaud Casteigts. Model driven capabilities of the da-grs model. In *International Conference on Autonomic and Autonomous Systems*, 2006.

[CBD02] Tracy Camp, Jeff Boleng, and Vanessa Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2:483–502, 2002.

[CC05] Arnaud Casteigts and Serge Chaumette. Dynamicity aware graph relabeling systems (da-grs), a local computation based model to describe manet algorithms. In *Parallel and Distributed Computing Systems*, pages 231–236, 2005.

[CEL$^+$08] Andrew T. Campbell, Shane B. Eisenman, Nicholas D. Lane, Emiliano Miluzzo, Ronald Peterson, Hong Lu, Xiao Zheng, Marco Musolesi, Kristof Fodor, and Gahng-Seop Ahn. The rise of people-centric sensing. *IEEE Internet Computing Special Issue on Sensor Networks*, 2008.

[CFK03] Xiaohu Chen, Michalis Faloutsos, and Srikanth V. Krishnamurthy. Power adaptive broadcasting with local information in ad hoc networks. In *Conf. on Network Protocols*, page 168. IEEE Computer Society, 2003.

[CHbA05] Reuven Cohen, Shlomo Havlin, and Daniel ben Avraham. *Handbook of Graphs and Networks*, chapter Structural properties of scale-free networks, pages 85–110. Number 4. Wiley-VCH Verlag GmbH Co. KGaA, 2005.

[CHC$^+$07] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, and James Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6:606–620, June 2007.

[CHE02] Mario Cagalj, Jean-Pierre Hubaux, and Christian Enz. Minimum-energy broadcast in all-wireless networks: Np-completeness and distribution issues. In *The 8th Int. Conf. on Mobile Computing and Networking (MobiCom)*, pages 172–182. ACM, 2002.

[CHE05] Mario Cagalj, Jean-Pierre Hubaux, and Christian C. Enz. Energy-efficient broadcasting in all-wireless networks. *Wirel. Netw.*, 11(1-2):177–188, 2005.

[Cis10] Cisco. Cisco visual networking index: Global mobile data traffic forecast update, 2009-2014, February 2010.

[CJ03] Thomas Clausen and Philippe Jacquet. Optimized link state routing protocol (olsr). RFC 3626, Internet Engineering Task Force, October 2003.

[CL02] F. Chung and L. Lu. Connected components in random graphs with given degree sequences. *Annals of Combinatorics*, 6:125–145, 2002.

[CL03] F. Chung and L. Lu. The average distances in random graphs with given expected degrees. *Internet Mathematics,*, 1:91–114, 2003.

[CL04] F. Chung and L. Lu. *Complex Networks*, chapter The small world phenomenon in hybrid power law graphs, pages 91–106. Springer-Verlag, 2004.

[CME05] D. M. Centola, M. W. Macy, and V. M. Eguíluz. Cascade dynamics of multiplex propagation. In P. Garrido, J. Maroo, & M. A. Muñoz, editor, *Modeling Cooperative Behavior in the Social Sciences*, volume 779 of *American Institute of Physics Conference Series*, pages 200–200, July 2005.

[CNM04] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111, 2004.

[Col88] James S. Coleman. Social capital in the creation of human capital. *The American Journal of Sociology*, 94:pp. S95–S120, 1988.

[CPV03] Corina Cortes, Daryl Pregibon, and Chris Volinsky. Computational me- thods for dynamic graphs. *Journal of Computational Graphical Statistics*, 12(4):950–970, December 2003.

[Cra80] R. K. Crane. Prediction of attenuation by rain. *EEE Transactions on Communications*, 28:1727–1732, September 1980.

[CSS03] Julien Cartigny, David Simplot, and Ivan Stojmenovic. Localized minimum-energy broadcasting in ad-hoc networks. In *IEEE Computer And Communications, INFOCOM*, pages 2210 – 2217, 2003.

[CT76] D. Cheriton and R.E. Tarjan. Finding minimum spanning trees. *SIAM J. Comput.*, 5:724–742, 1976.

[DA05] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Physical Review E, vol.*, 72:027104,, 2005.

[dBvKOS00] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer Verlag, 2000.

[DDDGA05] Leon Danon, Jordi Duch, Albert Diaz-Guilera, and Alex Arenas. Comparing community structure identification. *J. Stat.*, Mech.:P09008, 2005.

[DGOP07] Antoine Dutot, Frédéric Guinand, Damien Olivier, and Yoann Pigné. GraphStream: A Tool for bridging the gap between Complex Systems and Dynamic Graphs. In *Emergent Properties in Natural and Artificial Complex Systems. Satellite Conference within the 4th European Conference on Complex Systems (ECCS'2007)*, Dresden Allemagne, 10 2007. ANR SARAH.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. 10.1007/BF01386390.

[DLRM93] E. Decker, P. Langille, A. Rijsinghani, and K. McCloghrie. Definitions of managed objects for bridges. RFC 1493, IEEE Nework Working Group, 1993.

[DM04] Luca Donetti and Miguel A. Munoz. Detecting network communities: a new systematic and efficient algorithm. *J. Stat.*, Mech.:P10012, 2004.

[DNA10] J. J. Durillo, A. J. Nebro, and E. Alba. The jMetal framework for multiobjective optimization: Design and architecture. In *IEEE World Congress con Comp.Intelligence (WCCI)*, pages 4138–4325, 2010.

[DNLA08] J. Durillo, A. Nebro, F. Luna, and E. Alba. Solving three-objective optimization problems using a new hybrid cellular genetic algorithm. In *Parallel Problem Solving from Nature*, pages 661–670. Springer, 2008.

[DW05] F. Dai and J. Wu. On constructing k-connected k-dominating set in wireless network. In *IEEE International Parallel and Distributed Processing Symposium*, 2005.

[EH06] V. Erceg and K. V. S. Hari. Channel models for fixed wireless applications,. Technical report, IEEE 802.16 Broadband Wireless Access Working Group, 2006.

[EKKO08] Frans Ekman, Ari Keränen, Jouni Karvo, and Jörg Ott. Working day movement model. In *MobilityModels '08: Proceeding of the 1st ACM SIGMOBILE workshop on Mobility models*, pages 33–40, New York, NY, USA, 2008. ACM.

[ER60] P. Erdos and A. Renyi. On the evolution of random graphs. *Pub- lications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[Fer02] Afonso Ferreira. On models and algorithms for dynamic communication networks : The case for evolving graphs. In *4e rencontres francophones sur les Aspects Algorithmiques des Telecommunications (ALGOTEL'2002)*, France, May 2002.

[FFF99] C. Faloutsos, P. Faloutsos, and M. Faloutsos. On power-law relationships of the in- ternet topology, 1999.

[For56] L.R. Ford. Network flow theory. Paper P-923, The RAND Corperation, Santa Moncia, California, August 1956.

[For10] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486:75–174, 2010.

[Fuj03] Tetsuya Fujie. An exact algorithm for the maximum leaf spanning tree problem. *Computers and OR*, 30(13):1931–1944, 2003.

[GAN06] R.B. Groenevelt, E. Altman, and P. Nain. Relaying in mobile ad hoc networks: The brownian motion mobility model. *J. of Wireless Networks*, pages 561–571, 2006.

[GC07] Javier Gomez and Andrew T. Campbell. Variable-range transmission power control in wireless ad hoc networks. *IEEE Trans. on Mobile Computing*, 6(1):87–99, 2007.

[GdWFM02] M. Gerharz, C. de Waal, M. Frank, and P. Martini. Link stability in mobile wireless ad hoc networks. *Local Computer Networks, Annual IEEE Conference on*, 0:0030, 2002.

[GFPG10] Eugenio Giordano, Raphael Frank, Giovanni Pau, and Mario Gerla. Corner: a realistic urban propagation model for vanet. In *Proceedings of the 7th international conference on Wireless on-demand network systems and services*, WONS'10, pages 57–60, Piscataway, NJ, USA, 2010. IEEE Press.

[GK00] P. Gupta and P.R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[GN02] Michelle Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99:7821–7826, 2002.

[GP10] M. Gendreau and J.-Y. Potvin. *Handbook of Metaheuristics*, volume 146 of *Int. Ser. in Operations Research & Management Science*. Springer, 2010.

[Gra] Graphstream project webpage.

[Gra70] M.S Granovetter. *Changing Jobs: Channels of Mobility Information in a Suburban Community*. PhD thesis, Harvard University, 1970.

[Gra73] Mark S. Granovetter. The strength of weak ties. *The American Journal of Sociology*, 78(6):1360–1380, 1973.

[Har59] F. Harary. Status and contrastatus. *Sociometry*, 22:23–43, 1959.

[Hat81] M. Hata. Empirical formula for propagation loss in land mobile radio services. *IEEE Transactions on Vehicular Technology*, 29:317–325, 1981.

[HBG06] Luc Hogie, Pascal Bouvry, and Frédéric Guinand. An overview of manets simulation. *Electronic Notes in Theoretical Computer Science*, 150(1):81 – 101, 2006. Proceedings of the First International Workshop on Methods and Tools for Coordinating Concurrent, Distributed and Mobile Systems (MTCoord 2005).

[Hek06] Ramin Hekmat. *Ad-hoc Networks: Fundamental Properties and Network Topologies*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[Her03] K Herrmann. Modeling the sociological aspects of mobility in ad hoc networks. *Proceedings of the 6th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, pages 128–129, 2003.

[HGB04] Luc Hogie, Frédéric Guinand, and Pascal Bouvry. A heuristic for efficient broadcasting in the metropolitan ad hoc network. *Knowledge-Based Intelligent Information and Engineering Systems*, pages 727–733, 2004.

[HHF05] F. Hong, L. Hong, and C. Fu. Secure olsr. In *Proceedings of the 19th International Conference on Advanced Information Networking and Applications*, 2005.

[HL04] Yi An Huang and Wenke Lee. Attack analysis and detection for ad hoc routing protocols. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID'04)*, 2004.

[Hog05] Luc Hogie. The madhoc simulator. Technical report, Université du Havre, 2005.

[HTR$^+$04] A. Hafslund, A. Tonnesen, R. B. Rotvik, J. Andersson, and O. Kure. Secure extension to the olsr protocol. In *Proceedings of the OLSR Interop and Workshop*, 2004.

[JBRAS03] Amit Jardosh, Elizabeth M. Belding-Royer, Kevin C. Almeroth, and Subhash Suri. Towards realistic mobility models for mobile ad hoc networks. In *Proceedings of MOBICOM*, San Diego, CA, September 2003.

[JM96] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In Thomasz Imielinski and Hank Korth, editors, *Mobile Computing*, volume 353, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[JMC+01] Philippe Jacquet, Paul Mühlethaler, Thomas Clausen, Anis Laouiti, Amir Qayyum, and Laurent Viennot. Optimized link state routing protocol. In *IEEE INMIC'01, 28-30 December 2001, Lahore, Pakistan*, pages 62–68. IEEE, IEEE, December 2001.

[Kar82] Richard M. Karp. Dynamic programming meets the principle of inclusion and exclusion. *Operations Research Letters*, 1(2):49 – 51, 1982.

[KJQ08] I.A. Khan, A. Javaid, and Hua Lin Qian. Distance-based dynamically adjusted probabilistic forwarding for wireless mobile ad hoc networks. In *Wireless and Optical Communications Networks*, pages 1–6, 2008.

[KK06] Minkyong Kim and David Kotz. Extracting a mobility model from real user traces. In *Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'06*, 2006.

[KLN08] Brian Karrer, Elizaveta Levina, and M. E. J. Newman. Robustness of community structure in networks. *Phys. Rev. E*, 77:046119, 2008.

[KPL08] Jochen Koberstein, Hagen Peters, and Norbert Luttenberger. Graph-based mobility model for urban areas fueled with real world datasets. In *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[Kra92] D Krackhardt. *Networks and Organizations: Structure, Form,and Action*, chapter The Strength of Strong Ties: The Importance of Philos in Organizations, pages 216–239. Harvard Business School Press, 1992.

[Kru56] J.B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.

[KWYLD07] D. Kim, Y. Wu, F. Zou Y. Li, and D.-Zhu Du. Constructing energy efficient connected dominating sets with bounded diameters in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 2007.

[LBDC+01] Jinyang Li, Charles Blake, Douglas S.J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, pages 61–69, New York, NY, USA, 2001. ACM.

[LBXW09] Weifa Liang, Richard Brent, Yinlong Xu, and Qingshan Wang. Minimum-energy all-to-all multicasting in wireless ad hoc networks. *Trans. Wireless. Comm.*, 8(11):5490–5499, 2009.

[LF09] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: a comparative analysis. 08 2009.

[LH06] Mathieu Lacage and Thomas R. Henderson. Yet another network simulator. In *Workshop on ns-2: the IP network simulator*, New York, 2006.

[LHB+01] Li Li, Joseph Y. Halpern, Paramvir Bahl, Yi-Min Wang, and Roger Wattenhofer. Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks. In *Symposium on Principles of Distributed Computing*, pages 264–273, 2001.

[LHLC09] Ian X.Y. Leung, Pan Hui, Pietro Lio', and Jon Crowcroft. Towards real-time community detection in large networks. *Phys. Rev. E*, 79:066107, 2009.

[LHS03] Ning Li, Jennifer C. Hou, and Lui Sha. Design and analysis of an mst-based topology control algorithm. In *IEEE INFOCOM*, 2003.

[Lim02] Geunhwi Lim. Link stability and route lifetime in ad-hoc wireless networks. In *Proceedings of the 2002 International Conference on Parallel Processing Workshops*, pages 116–124, Washington, DC, USA, 2002. IEEE Computer Society.

[LKS06] Giuseppe Lugano, Jorma Kyppö, and Pertti Saariluoma. Designing people's interconnections in mobile social networks. In *Proceedings of the First International Conference on Multidisciplinary Information Sciences Technologies (In-Scit)*, pages 500–504, Badajoz, Spain, 25–27 October 2006.

[LNM03] Xiaoyan Li, Thu D. Nguyen, and Richard P. Martin. Using adaptive range control to optimize 1-hop broadcast coverage in dense wireless networks. In *SenSys*, pages 314–315, 2003.

[LRL+10] Kyunghan Lee, Injong Rhee, Joohyun Lee, Yung Yi, and Song Chong. Mobile data offloading: how much can wifi deliver? In *ACM SIGCOMM Conference*, pages 425–426, 2010.

[LSK09] Dimitrios Liarokapis, Ali Shahrabi, and Andreas Komninos. Diba: An adaptive broadcasting scheme in mobile ad hoc networks. In *Communication Networks and Services Research*, pages 224–231, 2009.

[MHM04] Marco Musolesi, Stephen Hailes, and Cecilia Mascolo. An ad hoc mobility model founded on social network theory. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, MSWiM '04, pages 20–24, New York, NY, USA, 2004. ACM.

[Mil67] S. Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[MLF+08] Emiliano Miluzzo, Nicholas D. Lane, Kristof Fodor, Ronald A. Peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, and Andrew T. Campbell. Sensing meets mobile social networks: The design, implementation and evaluation of the cenceme application. In *Proc. of 6th ACM Conference on Embedded Networked Sensor Systems (SenSys '08)*, Raleigh, NC, USA, Nov. 5-7 2008.

[MM04] C. Siva Ram Murthy and B. S. Manoj. *Ad Hoc Wireless Networks: Architectures and Protocols*. Prentice Hall International, 2004.

[MM06] M Musolesi and C Mascolo. A community based mobility model for ad hoc network research. *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, pages 31–38, 2006.

[Mon92] J.D. Montgomery. Job search and network composition: Implications of the strength-of-weak-ties hypothesis. *American Sociological Review*, 57:586–596, 1992.

[Mon94] J.D. Montgomery. Weak ties, employment, and inequality: An equilibrium analysis. *American Journal of Sociology*, 99:1212–1236, March 1994.

[MZ99] A. Bruce Mcdonald and Taieb Znati. A path availability model for wireless ad-hoc networks. In *Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE*, pages 35 – 40, September 1999.

[NG04] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, 2004.

[NMN01] Nesetril, Milkova, and Nesetrilova. Otakar boruvka on minimum spanning tree problem: Translation of both the 1926 papers. *DMATH: Discrete Mathematics*, 233, 2001.

[OSM] Openstreetmap project. http://www.openstreetmap.org/.

[P.101] Recommendation ITU-R P.1546. Method for point-to-area predictions for terrestrial services in the frequency range 30 mhz to 3000 mhz,. Technical report, International Telecommunication Union, 2001.

[PBGL08] A. Piyatumrong, Pascal Bouvry, Frédéric Guinand, and K. Lavangnananda. Trusted spanning tree for delay tolerant manets. In *Embedded and Ubiquitous Computing*, pages 293–299, 2008.

[PBGL09] Apivadee Piyatumrong, Pascal Bouvry, Frédéric Guinand, and Kittichai Lavangnananda. A study of token traversal strategies on tree-based backbones for mobile ad hoc - delay tolerant networks. In *Ultra Modern Telecommunications*, pages 1–8, 2009.

[PDB10] Yoann Pigné, Grégoire Danoy, and Pascal Bouvry. A platform for realistic online vehicular network management. In *IEEE International Workshop on Management of Emerging Networks and Services*, pages 615–619. IEEE Computer Society, 2010.

[PDB11] Yoann Pigné, Grégoire Danoy, and Pascal Bouvry. A vehicular mobility model based on real traffic counting data. In Thomas Strang et al., editor, *Nets4Cars/Nets4Trains 2011*, volume 6596 of *Lecture Notes in Computer Science*, pages 131–142. Springer, 2011.

[Per01] Charles E. Perkins. *Ad Hoc Networking*. Addison-Wesley, 2001.

[Pig08] Yoann Pigné. *Modélisation et traitement décentralisé des graphes dynamiques : Application aux réseaux mobiles ad hoc*. PhD thesis, Université du Havre, 2008.

[PR99] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications, WMCSA '99, February 25-26, 1999, New Orleans, Lousiana, USA*, pages 90–100. IEEE, IEEE, February 1999.

[Pri57] R.C. Prim. Shortest connection networks and some generalization. *Bell Sytem Technical Journal*, 36:1389–1401, 1957.

[PS97] S. Pallottino and M. G. Scutellà. Shortest path algorithms in transporta- tion models : classical and innovative aspects. Technical Report TR-97-06, Università di Pisa - Dipartimentao di Informatica, 1997.

[RAK07] Usha Nandini Raghavan, Reka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76:036106, 2007.

[Rap57] Anatol Rapoport. Contributions to the theory of random and biased nets. *Bulletin of Mathematical Biophysics*, 19:257–277, 1957.

[Rap02] T. Rappaport. *Wireless Communications, Principles and Practice*. Upper Saddle River Prentice-Hall PTR, 2002.

[RB08] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proc. Natl. Acad. Sci. USA*, 105(4):1118–1123, 01 2008.

[RB10a] Patricia Ruiz and Pascal Bouvry. Distributed energy self-adaptation in ad hoc networks. In *IEEE Globecom*, pages 539–543, Miami, USA, 2010.

[RB10b] Patricia Ruiz and Pascal Bouvry. Enhanced distance based broadcasting protocol with reduced energy consumption. In *Int. Conf. on High Performance Computing and Simulation (HPCS 2010)*, pages 249–258, 2010.

[RB10c] Patricia Ruiz and Pascal Bouvry. On the improvement of the enhanced distance based broadcasting algorithm. *Int. J. of Comm. Net. and Distributed Systems*, 2010.

[RDK$^+$08] Patricia Ruiz, Bernabé Dorronsoro, D. Khadraoui, P. Bouvry, and L. Tardón. Bodyf: A parameterless broadcasting protocol over dynamic forest. In *Workshop on Optimization Issues in Grid and Parallel Computing Environments, part of the High Performance Computing and Simulation Conference (HPCS)*, pages 297–303, Nicosia, Cyprus, 2008.

[RDV$^+$ne] P. Ruiz, B. Dorronsoro, G. Valentini, F. Pinel, and P. Bouvry. Optimisation of the enhanced distance based broadcasting protocol for manets. *J. of Supercomputing*, available online.

[Rey87] Craig Reynolds. Flocks, herds and schools: A distributed behavioral model. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 21(4):25–34, 1987.

[RP97] Elizabeth M. Royer and Charles E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Mobile Computing and Networking*, 1997.

[RR05] H. Reumerman and M. Runi. Distributed power control for reliable broadcast in inter-vehicle communication systems. In *2nd Int. Workshop on Intelligent transportation*, 2005.

[RWG97] K. J. Rizk, J. Wagner, and F. Gardiol. Two- dimensional ray-tracing modeling for propagation prediction in microcellular environments. *IEEE Trans. On Vehucular Tech.*, 46(2):508–518, 1997.

[San05] Paolo Santi. *Topology Control in Wireless Ad Hoc and Sensor Networks*. John Wiley Co., August 2005.

[SBA09] Julien Schleich, Pascal Bouvry, and Le Thi Hoai An. Decentralized fault-tolerant connected dominating set algorithm for mobile ad hoc networks. In *International Conference on Wireless Networks*, pages 354–360, 2009.

[Sch10] Julien Schleich. *Robust Dominating Set based Virtual Backbones for Wireless Ad hoc Networks*. PhD thesis, Université du Luxembourg, September 2010.

[SDBA09] Julien Schleich, Grégoire Danoy, Pascal Bouvry, and Le Thi Hoai An. Blackbone2, an efficient deterministic algorithm for creating 2-connected m-dominating set-based backbones in ad hoc networks. In *ACM International Workshop on Mobility Management and Wireless Access*, pages 91–98, 2009.

[SH96] A Sen and M L Huson. A new model for scheduling packet radio networks. In *Proc. 15th Joint Conference of the IEEE Computer and Communication Societies (INFOCOM)*, pages 1116–1124, 1996.

[SH97] Arunabha Sen and Mark L. Huson. A new model for scheduling packet radio networks. *Wireless Networks*, 3:71–82, March 1997.

[SSZ02] Ivan Stojmenovic, Mahtab Seddigh, and Jovisa D. Zunic. Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 13:14–25, 2002.

[St1] Gordon L. Stüber. *Principles of mobile communication (2nd ed.)*. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[STT05] Q. Sun, S. Y. Tan, and K. C. Teh. Analytical formulae for path loss prediction in urban street grid microcellular environments. *IEEE Transactions on Vehicular Technology*, 54(4):1251–1258, July 2005.

[SUM] Sumo: Simulation of urban mobility.

[TBK$^+$03] Chin-Yang Tseng, Poornima Balasubramanyam, Calvin Ko, Rattapon Limprasittiporn, Jeff Rowe, and Karl Levitt. A specification-based intrusion detection system for aodv. In ACM Press, editor, *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 125–134, New York, NY, USA, 2003.

[TBWK07] Chayant Tantipathananandh, Tanya Berger-Wolf, and David Kempe. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 717–726, New York, NY, USA, 2007. ACM.

[TCF09] Junwei Tian, Duanbing Chen, and Yan Fu. A new local algorithm for detecting communities in networks. *Education Technology and Computer Science, International Workshop on*, 2:721–724, 2009.

[THB^{+}02] J. Tian, J. Hahner, C. Becker, I. Stepanov, and K. Rothermel. Graph-based mobility model for mobile ad hoc network simulation. In *Proceedings of 35th Annual Simulation Symposium, in cooperation with the IEEE Computer Society and ACM*, San Diego, California, April 2002.

[TNCS02] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless Networks*, 8:153–167, March 2002.

[TvS02] Andrew S. Tanenbaum and Maarten van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.

[WBW01] Roger Wattenhofer, Paramvir Bahl, and Yi-Min Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *IEEE INFOCOM*, 2001.

[WCF08] Ying Wan, Duanbing Chen, and Yan Fu. A new efficient algorithm for detecting communities in complex networks. *Network and Parallel Computing Workshops, IFIP International Conference on*, 0:281–286, 2008.

[WCL07] Xutao Wang, Guanrong Chen, and Hongtao Lu. A very fast algorithm for detecting community structures in complex networks. *Physica A: Statistical Mechanics and its Applications*, 384(2):667 – 674, 2007.

[WL99] Jie Wu and Hailan Li. A dominating-set-based routing scheme in ad hoc wireless networks. *Telecommunication Systems*, 1999.

[WL08] Y. Wu and Y. Li. Construction algorithms for k-connected m-dominating sets in wireless sensor networks. In *Proceedings of the 9th ACM International Symposium on Mobile ad hoc networking and coputing (MOBIHOC)*, pages 83–90, May 2008.

[WS98] D. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–32, 1998.

[Wu03] J. Wu. An enhanced approach to determine a small forward node set based on multipoint relays. In *proc. of the IEEE Vehicular Technology Conference*, volume 4, pages 2774–2777, 2003.

[WWTL07] Y. Wu, F. Wang, M. T. Thai, and Y. Li. Constructing k-connected m-dominating sets in wireless sensor networks. In *Military Communication Conference (MILCOM 2007)*, pages 1–7, October 2007.

[YJB02] S. Yook, H. Jeong, and A. Barabasi. Modeling the internets large-scale topology. In *Proceedings of the National Academy of Sciences*, volume 99, pages 382–386, 2002.

[YLG03] Dan Yu, Hui Li, and I. Gruber. Path availability in ad hoc network. In *Proceedings of the 10th International Conference on Telecommunications (ICT2003)*, pages 383–387, February 2003.

[YLL10] Bo Yang, Jiming Liu, and Dayou Liu. An autonomy-oriented computing approach to community mining in distributed and dynamic networks. *Autonomous Agents and Multi-Agent Systems*, 20(2):123–157, 2010.

[YLN03] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications (INFOCOM 2003)*, volume 2, pages 1312–1321, April 2003.

[You96] C. D. Young. Usap: A unifying dynamic distributed m ultichannel tdma slot assignment protocol. In *IEEE Military Communications Conference Proceedings (MILCOM 1996)*, volume 1, pages 235–239, October 1996.

[You99] C. D. Young. Usap multiple access: dynamic resource allocation for mobile multihop multichannel wireless networking. In *IEEE Military Communications Conference Proceedings (MILCOM 1999)*, volume 1, pages 271–275, Nov 1999.

[Zac77] Wayne W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

[ZJ04] Xiaofeng Zhang and Lillykutty Jacob. Mzrp: An extension of the zone routing protocol for multicasting in manets. *Journal of Information Science and Engineering*, 20:535–551, 2004.

[ZyZzMz07] CAO Zhi-yan, JI Zhen-zhou, and HU Ming-zeng. An energy-aware broadcast scheme for directed diffusion in wireless sensor network. *J. of Comm. and Comp.*, 4,(5):28–35, 2007.

Publications

[HB09] Guillaume-Jean Herbiet and Pascal Bouvry. Urbisim: a framework for simulation of ad hoc networks in realistic urban environment. In *GIIS'09: Proceedings of the Second international conference on Global Information Infrastructure Symposium*, pages 373–378, Piscataway, NJ, USA, 2009. IEEE Press.

[HB10] Guillaume-Jean Herbiet and Pascal Bouvry. SHARC: community-based partitioning for mobile ad hoc networks using neighborhood similarity. In *IEEE WoWMoM 2010 (IEEE WoWMoM 2010)*, Montreal, Canada, 6 2010.

[HB11a] Guillaume-Jean Herbiet and Pascal Bouvry. On the generation of stable communities of users for dynamic mobile ad hoc social networks. In *The International Conference on Information Networking (ICOIN 2011)*, Kuala Lumpur, Malaysia, 2011.

[HB11b] Guillaume-Jean Herbiet and Pascal Bouvry. *Social Networks: Computational Aspects and Mining*, chapter Social Network Analysis Techniques for Social-oriented Mobile Communication Networks. Number 53. Springer, 2011.

[HB11c] Guillaume-Jean Herbiet and Pascal Bouvry and Frédéric Guinand. Social Relevance of Topological Communities in Ad hoc Communication Networks. In *The International Conference on Computational Aspects of Social Networks (CASoN 2011)*, Salamanca, Spain, 2011.

Nomenclature

(u, v)	An edge of between vertices u and v of a graph
β	Path-loss exponent in a free space propagation environment
Δ	see $\Delta(\mathcal{G})$
$\Delta(\mathcal{G})$	The average degree, of the graph $\mathcal{G}$
$\delta(x, y)$	The Kronecker function
Δ_w	see $\Delta_w(\mathcal{G})$
$\Delta_w(\mathcal{G})$	The weighted average degree of the graph $\mathcal{G}$
Γ	see $\Gamma(\mathcal{G})$
$\Gamma(\mathcal{G})$	The density of the graph $\mathcal{G}$
$\gamma(v)$	Clustering coefficient of a vertex v
$\hat{s}_u(v)$	Node-centric stability estimation of vertex v with respect to node u
λ	Wavelength, in meters, of a radio wave
$\mathbb{T}$	A temporal base
$\mathbf{A}$	A graph adjacency matrix
$\prime(v)$	Obstruction set of the node v in the Jardosh, Belding-Royer *et al.* obstacle-based propagation model
$\mathcal{B}$	see $\mathcal{B}(\mathcal{G})$
$\mathcal{B}(\mathcal{G})$	A virtual backbone on the graph $\mathcal{G}$

$\mathcal{C}(\mathcal{G})$	A community assignment on the graph $\mathcal{G}$
$\mathcal{D}$	see $\mathcal{D}(\mathcal{G})$
$\mathcal{D}(\mathcal{G})$	A dominating set on the graph $\mathcal{G}$
$\mathcal{D}_c$	see $\mathcal{D}_c(\mathcal{G})$
$\mathcal{D}_c(\mathcal{G})$	A connected dominating set on the graph $\mathcal{G}$
$\mathcal{F}$	A forest, set of trees of a graph
$\mathcal{F}_s$	A spanning forest of a graph
$\mathcal{G}$	A graph
$\mathcal{G}_s$	see $\mathcal{T}_s(\mathcal{G})$
$\mathcal{G}_t$	A static image of the graph $\mathcal{G}$ à time $t \in \mathbb{T}$
$\mathcal{G}_{er}$	A random graph, using the Erdös-Rényi model
$\mathcal{G}_{lat}(N, p)$	A regular lattice graph, with N nodes and a probability p for an edge to be created between grid-adjacent vertices
$\mathcal{G}_p(N)$	A random graph, with N nodes and a probability p for an edge to be created
$\mathcal{G}_{sf}(N, \alpha)$	A scale-free graph, with N nodes and a degree distribution following a power-law of parameter α
$\mathcal{G}_{sw}$	A small-world graph
$\mathcal{J}$	A journey
$\mathcal{N}$	A network
$\mathcal{N}_{p(r_{u,v})}$	A geometric random network with a probability p to create an edge between vertices u and v at distance $r_{u,v}$
$\mathcal{P}(\mathcal{G})$	A path in the graph $\mathcal{G}$
$\mathcal{P}_w(\mathcal{G})$	A weighted path in the graph $\mathcal{G}$
$\mathcal{P}_{min}(u, v)$	The shortest path between vertices u and v
$\mathcal{R}$	A region for a network
$\mathcal{S}(\mathcal{G})$	A subgraph of $\mathcal{G}$
$\mathcal{S}_{\mathcal{G}}$	The evolutive from of the graph $\mathcal{G}$
$\mathcal{S}_C(\mathcal{G})$	A community of the graph $\mathcal{G}$
$\mathcal{S}_{cc}(\mathcal{G})$	A connected component of the graph $\mathcal{G}$
$\mathcal{T}$	A tree graph
$\mathcal{T}(\mathcal{G})$	A subtree, tree subgraph, of the graph $\mathcal{G}$

$\mathcal{T}_s(\mathcal{G})$	A spanning tree of the graph $\mathcal{G}$
d	The dimension of a network region
$\Pr$	The probability operator
$\overline{\gamma}(\mathcal{G})$	Clustering coefficient of a graph $\mathcal{G}$
$\bar{l}(\mathcal{G})$	Characteristic path length of a graph $\mathcal{G}$
σ	Power fluctuation, in dB, of a transmitted signal due to log-normal shadowing
ξ	Parameter of the geometric random graph model, ratio of the signal power fluctuations and the path-loss exponent
$C(v)$	Current community of the vertex v
$d(v)$	see $d_{\mathcal{G}}(v)$
$d_{\mathcal{G}}(v)$	The degree of vertex v in the graph $\mathcal{G}$
$d_w(v)$	see $d_{w,\mathcal{G}}(v)$
$d_{w,\mathcal{G}}(v)$	The weighted degree of a vertex v in the graph $\mathcal{G}$
E	see $E(\mathcal{G})$
e	An edge of a graph
$E(\mathcal{G})$	The set of edges, or nodes, of a graph $\mathcal{G}$
I	An independent set
L	A location function
$l(\mathcal{P})$	The length of a path $\mathcal{P}$
$l(u, v)$	The length of the shortest path between vertices u and v
L_{dyn}	A dynamic location function or mobility model
M	The size (number of edges) of a graph
$m_{str}(v, C(v))$	Membership strength for the vertex v in community $C(v)$
$m_{w,str}(v, C(v))$	Weighted membership strength for the vertex v in community $C(v)$
$MST(\mathcal{G})$	The minimum spanning tree of the graph $\mathcal{G}$
N	The order (number of vertices) of a graph
$N(v)$	The neighborhood of a vertex v
$N^2(v)$	The two-hop neighborhood of a vertex v
n_c	The number of communities found on the graph $\mathcal{G}$
$N_w(v)$	The weighted neighborhood of a vertex v
$n_{sim}(u, v)$	Neighborhood similarity between vertices u and v

$n_{w,sim}(u, v)$	Weighted similarity measure between vertices u and v
P	An evolution process
Q	The modularity metric
Q_w	The weighted modularity metric
$r_{u,v}$	The distance between two vertices u and v
T	A transmission function
u	A vertex of a graph
V	see $V(\mathcal{G})$
v	A vertex of a graph
$V(\mathcal{G})$	The set of vertices, or nodes, of a graph $\mathcal{G}$