Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

Sérgio Luís Dias Lima Gramacho	Date

Autonomic Formation of Large Scale Wireless Mesh Networks

By

Sérgio Luís Dias Lima Gramacho
Doctor of Philosophy

Computer Science and Informatics

Avani Wildani, Ph.D.
Advisor, Comittee Chair

Vaidy Sunderam, Ph.D.
Committee Member

Shun Yan Cheung, Ph.D.
Committee Member

Christian Maciocco, Intel Labs
Committee Member

Accepted:

Lisa A. Tedesco, Ph.D.
Dean of the James T. Laney School of Graduate Studies

Date

Autonomic Formation of Large Scale Wireless Mesh Networks

By

Sérgio Luís Dias Lima Gramacho
M.S., Emory University, 2019
M.S., Federal University of Bahia, 2014
MBA, Salvador University, 2009
B.S., Federal University of Bahia, 1992

Advisor: Avani Wildani, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2020

Abstract

Autonomic Formation of Large Scale Wireless Mesh Networks

By Sérgio Luís Dias Lima Gramacho

A Wireless Mesh Network (WMN) is an appealing network architecture for low-cost and wide geographical coverage. It serves as a potential alternative solution to improve worldwide connectivity in low to high-income countries. Theoretical studies predict, however, insufficient capacity for such network architecture at larger scales. Moreover, the inherently distributed nature of WMNs and their typical distributed network control mechanisms turned them hardened and inflexible to adapt to specific and varying control customization demands.

We propose the modernization of the WMN architecture by allowing the general applicability of Software Defined Networking (SDN) on the implementation of WMN control planes for increased control flexibility while also enforcing frequency diversity to promote throughput capacity. To achieve this, we devised autonomic agents that induce the formation of WMN topologies as a set of interconnected partitions, supporting a cooperative, multi-domain SDN-based WMN control plane able to operate at large-scales and low-cost for increased control flexibility. Moreover, the nature of this autonomic network based on WMN partitions also allows the enforcement of frequency diversity at low-cost and low-complexity for improved throughput capacity. The partitioned topology format is the result of the concurrent and distributed operation of our autonomic agents that manipulate the formation of the WMN using local information, without relying on any central controlling entity, characterizing a scalable and resilient solution. Partitions hold as invariants their maximum diameter and their maximum per node interface degree. These two induce an additional invariant: the maximum partition size in mesh nodes. Finally, the three properties permit limiting control latency and workload on Software Defined Network (SDN) control plane domains. Our agents have different objectives such as organize, heal, optimize; thus, they cooperate and compete to determine final WMN topologies. We show that the competitive/cooperative behavior of these agents converge to stable formations in bounded time even under extreme mesh node churn conditions. The solution relies on an in-network leader election and stochastic delays to achieve the eventual stabilization of formed WMN topologies.

Autonomic Formation of Large Scale Wireless Mesh Networks

By

Sérgio Luís Dias Lima Gramacho
M.S., Emory University, 2019
M.S., Federal University of Bahia, 2014
MBA, Salvador University, 2009
B.S., Federal University of Bahia, 1992

Advisor: Avani Wildani, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2020

Table of Contents

List of Figures

List of Tables

List of Definitions, Lemmas, Theorems

Chapter 1 Introduction

Elements presented in this dissertation are the object of the patent application "International PCT Application No. PCT/US2020/019722", with the title "Systems, Devices, and Methods for Autonomic Formation of Wireless Networks."

Research by The Economist Intelligence Unit (EIU) [1] in 2017 observed that only 25% of the population of medium-income and 8% of the population of low-income[1] countries are *connected*. From a worldwide perspective, 40% of the global population is poorly or not connected to communication networks. Moreover, this connectivity problem also exists in high-income and developed nations. In the United States, recent research by Microsoft (2019) [2] contests the official broadband penetration data produced by the FCC. While the FCC's data indicate that only 24.7 million people in the U.S. do not have broadband available, Microsoft claims that a whopping 162.8 million people do not use the Internet at broadband speeds. Both studies from EIU and Microsoft determined that the high cost for providing connectivity services is a critical factor. Communication networks heavily rely on expensive linkage technologies for network distribution such as fiber-optics, incurring a high cost for initial provision, a barrier for serving areas with a lower population density, or low average income. Microsoft suggests applying alternative architectures of wireless networks to solve the problem, achieving lower cost, reduced complexity solutions.

The statistics above have severe impacts. Studies [3] correlated the impact of access to broadband communication to jobs and GDP growth: low broadband availability correlates with low GDP and a weak job market. Moreover, [2] found that the U.S. counties with the highest unemployment rates also have the lowest use and access to broadband communication.

Wireless Mesh Networks (WMNs) are an attractive wireless network architecture for bridging the gap in worldwide connectivity. WMNs' most sought-after attributes are wide connectivity at low cost. However, despite many and distinct scientific investigations about WMNs, we do not yet see a massive application of large-scale and purely wireless mesh networks. Srivathsan *et al.* [4] claim that the scalability of a given network technology is a significant factor in its adoption and development. Previous studies about

[1] The terms *medium-income* and *low-income* countries are specific to the internal classification of the report, not necessarily correlating with concepts such as *developing* countries.

WMNs [5] assume application scenarios such as community networks, smart grid, military networks, transportation, security surveillance systems, building automation, health, broadband home networking[2]. This ordered list shows community networks as the most frequently studied application scenario.

When searching for alternatives for the limitations regarding the scaling of WMNs, our first insights into this dissertation research stemmed from the scaling of complex systems such as biological (any form of life), business organizations (companies), and urban settlements (cities). Bettencourt *et al.* [6, 7] found a surprisingly super-linear scaling characteristic on metrics related to social relations in cities. Arbersman *et al.* [8], through an analytical model, proposed that long-distance relations promoting the integration of people from diverse backgrounds explain cities' super-linearity. Moreover, Bettencourt *et al.* [6, 7] claim that *innovations* are the fuel that supports the continual growth in cities. In the field of networking, known for its slow evolution, Software Defined Network (SDN) is the paradigm that promoted flexibility that fuels innovation in networks [9].

A new trend also motivated my dissertation research: the concept of distributed organizations. The experiences with Open Source Software (OSS) paved the way for distributed ownership in software development based on collective agreement and networked organization. Roughly 20+ years later, collective development principled in OSS supports a potentially significant change in our social relations: cryptocurrencies and the blockchain present us powerful alternatives to achieve collective agreements without the reliance on central authorities. I argue that a distributed ownership and community-driven network should be pursued. I am not alone with this vision. Microsoft's *Airband* initiative aims at promoting the growth of broadband use by communities and small entrepreneurs in rural U.S. through advanced but low-cost wireless technologies. [10]. In fact, examples of Community Wireless Networks (CWNs) exist in Europe, Africa, Asia, the U.S, heavily relying on manual deployment and management, and reduced throughput capacity when using WMNs. I envision that more scalable, programmable, and autonomic WMN technologies can drive increased adoption by communities in need.

The goal of this dissertation research is the modernization of the WMN network architecture, assuming low-cost and low-complexity settings motivated to facilitate connectivity in underserved regions of the world. The aim is to supporting the general applicability of centralized control planes (such as SDNs) in a cooperative multi-domain approach to facilitate the implementation of programmable WMNs at large scales. The method devised produces a WMN architecture composed of inter-connected partitions in which each

[2]The work accounted for the studies about WMNs available at the IEEE Xplore repository.

partition becomes an SDN control domain. The SDN-based approach renders low-cost capacity improvement from multi-path routing and rich multi-layer metrics, and provides flexibility for innovation advent from the *programmability* of SDNs [11]. Another essential characteristic is the enforcement of frequency diversity into WMNs while maintaining the average number of per-node interfaces at the minimal range $(1, 2]$.

At the heart of our[3] approach is its uniqueness: we apply the principles of autonomic computing, minimizing provisional and management efforts. Our autonomic agents control the formation of WMNs using local information to enforce robustness to failures and support unbounded topological settings in size and density.

Our first initiative was conceiving and implementing a specialized platform for the evaluation of autonomic agents that manipulate the behavior of mesh nodes of WMNs. Chapter 3 describes this platform, which isolates the execution of agents into a fast prototyping environment, supporting the execution of wireless network protocols into well-know simulation engines for precise representation, control, and repeatability. Finally, we directed the execution of pre-existing software components into lightweight containers to minimize the effort and risk of transposing software to simulated settings.

In a second phase described in Chapter 4, we evaluated the throughput capacity scaling under an agent-based autonomic WMN formation and reorganization at the *physical* and *link* network layers. The agents, conceived as internal software processes to the mesh nodes, consume static or slow-varying information about the neighboring structure of the WMN, feeding this data to the self-organizing functions of agents. This use of *local* information is a characteristic inherent to the autonomic concept [12] that improves scalability regarding the size of the networks. The resulting topology structure is a set of WMN partitions of differing sizes in the number of nodes. We assumed that each partition is an instance of a WMN of a given size, which allows the creation of capacity scaling curves regarding the size of WMNs. To determine each partition's capacity, the agents operating inside the mesh nodes command random communications with other nodes in the same partition.

In the scaling analysis in Chapter 4, we find that the property of enforcing a node degree bound induces improved capacity scaling into WMNs. Although the mesh node degree control has been approached by previous research through directional antennas and transmit power control, our approach is new, relying on the partition membership decisions of our autonomic agents. Our best agent configuration improved the initially

[3] In this dissertation, I use plural forms of pronouns such as *we, our* to refer to parts of the dissertation research conducted with collaborators. I use singular forms such as *I, my* when referring to the dissertation itself or decisions accounted to myself, the author.

highly sub-linear scaling from $\beta = 0.08$ to $\beta = 0.28$, although still sub-linear. This result assumed fitting the experimental data as follows: $C(n) = w \cdot n^{\beta}$, C the capacity, w a constant representing the wireless link throughput, n the number of nodes on a partition, and β the scaling factor. We also verify that the capacity scaling trend differs from the one determined by previous research. On extending the capacity scaling study to larger settings from 12 to 50 nodes, we, first, confirmed the initial negative scaling trend previously found around $\beta = -0.68$. However, we find that the negative scaling trend changes to positive for larger networks.

Leveraging what we learned about the effective and useful properties to the autonomic agents, in Chapter 5, we concentrate on better defining and evaluating our autonomic agents regarding their ability to perform self-organization in realistic concurrency settings. Convergence is an essential outcome of a self-organizing network. A non-converging solution stresses the network control plane due to excessive routing information updates in the distributed routing schemes or on managing a large number of network control events on a SDN controller.

We show that the distributed execution of our self-organizing agent model in the mesh nodes, up to this point named *Smart*, can lead to stable partitioning solutions. We describe the requirements for such convergence. The agents also achieved a reasonably balanced partition structure and maintained the two invariants we defined for our WMN partitions: a bounded node degree and bounded partition diameter. Moreover, our distributed agent design resembles a consensus mechanism in the primary-backup class [13], relying on an in-network leader election mechanism, and exploiting properties of the Degree/Diameter Graph problem [14] to achieve both convergence and balanced partitioning outcomes. Finally, we use stochastic delays in the execution cycle of the agents as the minimal condition in which this agent system achieves convergence. Moreover, we also devise a partial ordering mechanism (yet using local information) to improve the efficiency of the convergence process regarding the number of node movements between partitions. We confirm these results for extreme node churn conditions of $>$ 90% of nodes activated in a short time frame, resembling a power outage return setting.

In Chapter 6, we address the connectivity of our segmented WMN composed of overlapping but disconnected partitions. We devise a self-healing agent that can maintain the invariants of our network partitioning scheme while enforcing their inter-connectivity. This self-healing agent, named *SmartHeal*, competes and cooperates with our previous self-organizing agent, now re-branded *SmartOrg*, to implement an integrated multi-agent type system. Although both agents are software processes inside the mesh node, each one manages the connectivity of a different wireless network interface. Therefore, the self-

healing behavior in our mesh nodes implies the existence of a second wireless interface. To support our low cost and low complexity goals, we devised a version of the *SmartHeal* agent that achieves full connectivity of large-scale WMNs with high probability requiring only 20% of the mesh nodes running healing agents. Critical to this achievement was the use of different node degree bounds for the *SmartOrg* and *SmartHeal* agents in the same partition, and particularly setting the *SmartHeal* degree bound $+1$ with respect to the *SmartOrg* degree bound.

Moreover, each agent is also independent regarding their execution inside the mesh node (independent timing on their execution cycle). This asynchronous characteristic simplifies the implementation of this agent system. Finally, in Chapter 6, we also explored variations on the liveness of *SmartHeal*. We found a trade-off between the efficiency of inter-connecting partitions and increased link reliability and capacity on wireless technologies with multiple speeds and modulations.

Beyond Chapters 3, 4, 5 and 6 described above, this dissertation has the Chapter 2 (Background) which starts by presenting information related to the definitions of WMNs, the scaling of complex systems, and the capacity scaling of WMNs. The Background chapter continues describing the application of Multiple Radio Multiple Channel (MRMC) into WMNs, routing paradigms and protocols applied to the control of WMNs, and Software Defined Networking and its application to the control of small scale WMNs. It ends presenting the principles of autonomic computing and the characteristics of CWNs. Chapter 7 consolidates work developed during the dissertation research regarding the application of SDN to WMNs, such as a reference architecture for SDN-based WMNs that supports the implementation of centralized control. Assuming the centralized control, we proposed a SDN-based Time Division Multiple Access (TDMA) scheduler, a contention aware multi-path WMN routing mechanism, and a contention minimization mechanism based on the centrality of small scale WMNs such as our partitions. In Chapter 8, I conclude the dissertation and provide my vision for future directions related to this research.

Chapter 2 Background

This chapter presents research related to the two major topics of interest in this work: the throughput capacity of WMNs and the flexibility of WMN traffic control. We comment on related research on the perspective of our goals of large-scale, low-cost, low-complexity WMNs. We start with definitions of WMNs and the factors affecting their capacity and control on the sections 2.2 and 2.3. We follow presenting research regarding the analytical capacity scaling of WMNs in minimal (Section 2.4) and advanced link diversity settings (Section 2.5). In Section 2.7, we characterize Autonomic Computing, a principle that we apply to our proposed solution. We continue presenting research related to the network control (WMN routing) of WMNs in the Sections 2.8 and 2.9. In the Section 2.10, we present the most commonly applied setting for WMNs, CWNs. Finally, we conclude this chapter describing the challenges related to the evaluation of solutions for WMNs in Section 2.11.

2.1 The scaling of complex systems

The study of the *scaling of complex systems* determines the variation of system metrics regarding the growth of the system size. Examples are the scaling of life (from cells to ecosystems), business organizations (companies), and municipalities (cities).

In most cases, complex systems show sub-linear scalings, such as the metabolism of living organisms regarding their weight [15] (scaling factor $\beta = 2/3$) and the revenue or profit of companies regarding their size in the number of employees [16]. Another case is the sub-linear scaling (scaling exponent $\beta \approx 0.8$) on the infrastructural metrics of cities such as road pavement area, number of gas stations, and others [6, 7].

However, Bettencourt *et al.* [6, 7] found a surprisingly super-linear scaling characteristic on metrics related to social relations in cities such as innovation, patents, wages, crime regarding the *city size* in the number of citizens. In an attempt to explain the super-linear scaling of cities, Arbersman *et al.* [8] design a model that analytically showed that the existence of long-distance relations is a crucial factor on cities' super-linearity. At long distances, the increased probability of relationships between people of diverse backgrounds foments innovation to fuel cities' growth.

Moreover, Bettencourt *et al.* [6, 7] observes that cities never cease to exist, contrasting to the well-known life-cycle of living organisms and companies that die or disappear after

its maturity phase characterized by slow to no growth. Continuous super-linear growth implies the starvation of the resources that support cities. Authors proposed, based on their data patterns of city growth, that cyclical breakthroughs promoted by innovations explain the sustainable increase of communal organizations.

2.2 On the definition of WMNs

Infrastructure mode WMNs (Infra-WMNs) and *Client mode* WMNs (Client-WMNs) are the subject of this research. In both cases, mesh nodes have fixed positions and forward packets on behalf of other nodes. However, they differ regarding the source and destination of data flows. A WMN in *infrastructure mode* has specific nodes in fixed positions that are responsible for packet forwarding using wireless technology in a multi-hop fashion: the infra or backbone nodes. The clients in an Infra-WMN are the second type of node which use other network interfaces of the infra-nodes to send and receive packets through the network [17].

Client-WMNs are ad-hoc wireless networks in which all nodes are fixed and can forward packets on behalf of each other [17]. However, the fixed ad-hoc mesh nodes of a *Client-WMN* are the source and destination of flows beside the task of packet forwarding on behalf of other nodes.

From now on, we will use the term WMN generically to refer to the two previous forms of multi-hop wireless network architecture.

We describe other forms of meshed wireless networks that are out of our scope to complement this characterization. Generic ad-hoc networks, differently from Client-WMNs, do not assume restrictions on the mobility of mesh nodes. The terms MANET (Mobile Ad-Hoc Networks) or VANET (Vehicular Ad-Hoc Networks) are commonly used when mobility is assumed for the generic mobile mesh node or the specific mesh node based on mobile vehicles, respectively.

Hybrid-WMNs are a third form of WMN [17]. Similarly to the Infra-WMNs, they assume two types of nodes (infra and client nodes). However, the client nodes of Hybrid-WMNs share the same wireless communication medium used by the wireless mesh infrastructure to originate and consume data flows. This latter form is also out of our scope.

2.3 Factors affecting the capacity and control of WMNs

Previous research showed that many factors contribute to the throughput capacity of WMNs such as the co-channel interference, the routing protocol overhead, the half-duplex nature of radio antennas, complexities in handling multiple frequency radio systems, deployment architecture, and medium access control. Other examples are the topology attributes such as denseness of nodes and the degree of nodes and how these vary with the growth of the network, the communication pattern such as locality and the number of hops, the network control paradigms, and routing metrics [4, 5, 17, 18].

The distributed nature of WMNs induced most control mechanisms, also known simply as routing protocols, to use distributed paradigms. However, later in this chapter, we show that centralized WMN control paradigms show a better exploration of WMNs' capacity. Moreover, drawing from the experiences with the SDN paradigm in other network settings, we see that flexibility, adaptability advent from the centralized control are also of utmost importance [11].

Another relevant aspect is the approach to evaluation. A large number of theoretical studies exist on the capacity scaling of WMNs. They provide ranges on the asymptotic capacity scaling of WMNs given assumptions made about the WMN topology attributes and the WMN implementation solution. However, analytical models do not capture all practical aspects of the operation of a WMN, thus serving as asymptotic bounds and indications of techniques that can promote capacity scaling. Experimental studies, on the other hand, restrict their evaluation of capacity to small ranges of network sizes, limiting the inference of capacity scaling trends. Therefore, *experimental capacity scaling* represents a relevant contribution given the lack of studies in this field.

We present a framework to organize and simplify the description of related research regarding WMNs' control mechanisms and throughput capacity. This framework assumes *a) density attributes on the WMN topology* and a *traffic pattern* as inputs. It also conceives two general tasks on improving WMNs capacity: the *b) increase of wireless link diversity* to promote the flow parallelism potential and *c) devising routing mechanisms* that can explore the existing flow parallelism potential advent from the enforced link diversity to provide higher capacity.

The WMN topology can be *densifying* or *extended* regarding its growth pattern [18], and show *homogeneous* or *inhomogeneous* density [19]. *Densifying* topologies assume an increasing node density while scaling in size (fixed area). *Extended* topologies assume an approximately constant node density in the area, implying increased area while the number of nodes scales up [18]. The *extended* topologies are consistent with our goal of achieving

increased coverage with the increase of WMNs in size. It is also consistent with the CWNs application scenario, which aims at broad coverage. While *homogeneous* topologies present a mostly uniform density of nodes over area, *inhomogeneous* are characterized by denser areas (node clusters) embedded into a general area of sparsely distributed nodes [19, 20]. We prefer not to impose any preference regarding homogeneity. Later we will see that our approach consists of manipulating the existing topology rather than assuming specific characteristics on it for implementing optimizations.

The enforcement of link diversity can rely in different domains: the *space* domain (extended networks, directional antennas, power control), the *frequency* domain (Frequency Division Multiple Access (FDMA) - multiple radios multiple channels), the *time* domain (TDMA), and the *code* domain (Code Division Multiple Access (CDMA), Multiple-input and multiple-output (MIMO)). Routing mechanisms can assume *unified* or *hierarchical* architectures, *distributed* or *centralized* paradigms, *proactive* or *reactive* routing, use *in-layer* or *cross-layer* routing metrics.

We leave out theoretical studies that assumed *heterogeneous* WMN topologies (the combination of wired and wireless links) and unbounded wired capacity [21] or mobile nodes with unbounded buffer sizes [22] as they impose conditions out of the scope of this research. While impractical and imposing an unbounded delay, the latter claims a linear capacity scaling.

2.4 Analytical capacity scaling of single frequency WMNs

The minimum degree of link diversity that allows implementing a WMN is the single-channel single frequency architecture in which only the space domain diversity exists (for extended topologies). Gupta *et al.* [23] are the first to characterize the capacity scaling of WMNs under this architecture. Authors use two different models of analysis: random and arbitrary. In common, the models assume a number n of mesh nodes in fixed positions, the per-node throughput capacity of w bits per second, and the nodes use a shared wireless channel to send packets in a multi-hop fashion from source to destination with unlimited demand (continual transmission).

In the *Random* model [23], the node placement is random, and source nodes choose their destination pair at random. Also, the transmission range (the range of the radio transmissions) is fixed, given the assumption of equal transmission power p. Equation 2.1 describes the asymptotic capacity scaling of the network, which has a sub-linear format.

$$\Theta(n) = w \cdot \sqrt{\frac{n}{\log n}} \qquad \text{network wide capacity} \tag{2.1}$$

In the *Arbitrary* model [23], nodes have arbitrary positions and destination pair choices. Also, transmission power levels are arbitrary, implying arbitrary transmission ranges for each source node. The authors denominate the combination of arbitrary destination nodes, transmission rates, and transmission power levels a *traffic pattern*. Equation 2.2 show the improved capacity scaling for the arbitrary scaling model with a scaling slope of $\beta = 0.5$.

$$\Theta(n) = w \cdot \sqrt{n} \qquad \text{network wide capacity} \tag{2.2}$$

Gupta *et al.* [23] suggest that, since the individual node capacity reduces for larger networks, network designers should consider limiting the WMN size growth to keep the applicability of WMNs in practical settings. Besides, the authors suggest using short-distance wireless transmissions over long wireless ranges to minimize the number of transmitting nodes under contention. This recommendation contrasts with the simplest (and most common) metric used by routing protocols: the number of hops. This metric will enforce the preference of short network *paths*, which implies choosing long-distance transmissions.

Therefore, we have here the insight that the choice of routing metrics will impact the capacity scaling of WMNs. In section 7.3, we present a centralized routing scheme based on the path with the least wireless contention. The idea stems from the insight here that short-range links might provide better WMN capacity.

Also, we have additional insight into our design: keep WMNs small. We built on this idea to assume a large scale WMN as the combination of multiple, small scale, single frequency WMNs. Section 2.5 presents research on the analytical capacity scaling of WMNs assuming the organization of nodes in clusters, showing impressive results regarding capacity scaling.

2.4.1 Degree manipulation through directional antennas and power control

An increase in the degree of spatial diversity through directional antennas and beamforming could improve the capacity of WMNs by allowing an increased number of concurrent transmissions [23]. Yi *et al.* [24] confirmed these expectations applying directional antennas to the *Arbitrary* and *Random* WMN models conceived in [23]. This approach represents one strategy of reducing the co-channel interference, allowing for more concurrent flows.

Assuming that α is the transmitting (tx) antenna angle and β the receiving (rx) antenna angle, the gain for random networks is $r_{tx} = \frac{2 \cdot \pi}{\alpha}$ for directional tx antennas,

$r_{rx} = \frac{2 \cdot \pi}{\beta}$ for directional rx antennas, and $g_{trx} = \frac{4 \cdot \pi^2}{\alpha \cdot \beta}$ for directional tx and rx antennas. The gain for arbitrary networks is $a_{txa} = \sqrt{\frac{2 \cdot \pi}{\alpha}}$ and $a_{rxa} = \sqrt{\frac{2 \cdot \pi}{\beta}}$ for directional tx and rx antennas respectively. These gains, however, represent only constant factors.

Li *et al.* [25] revisit the application of directional antennas on transmitting nodes and present an improved gain of $O(\log n)$ for the *random* model. Differently from [24], this gain is not a constant factor, therefore, contributing to an increased capacity scaling for WMNs. This new gain relies on advanced antennas with multiple beams that can cover all possible directions, and are selected accordingly with the desired transmission direction. This description is known as advanced antenna beamforming.

Advanced forms of beamforming are complex and expensive solutions due to the requirements on the antenna system (multiple antennas) and complex radio processing, limiting the applicability to settings in which low-cost, low-complexity are relevant requirements.

Using transmission power control to enforce spatial diversity and manipulate the degree of network nodes, Kleinrock *et al.* [26] theoretically showed that a node degree of six (the exact value is 5.89) is optimum for a multi-hop wireless network (such as a WMN). Authors claimed that this degree is the best compromise between the length of paths and the wireless channel spatial reuse. In the optimal degree setting, authors claimed to achieve a network capacity proportional to $\sqrt{n}$, the same capacity scaling demonstrated by Gupta *et al.* [23] for the arbitrary model. Royer *et al.* [27] validate the optimum degree claim experimentally using a stationary network.

Our autonomic agents-based solution builds on the idea of controlling the degree of mesh nodes. However, we rely solely on the behavior of our self-organizing, self-healing agents at the physical and link layers to achieve a practical degree manipulation and density control. Therefore, our degree control neither relies on directional antennas or power control.

2.4.2 Concentrated traffic pattern

Jun *et al.* [28] contribute to the study of WMNs capacity scalability by adding more practical constraints to the traffic pattern. They claim that most WMNs are used for access to some external network such as the Internet. Such traffic pattern will rely on a subset of WMN nodes that will behave as gateways. The gateways will concentrate most of the traffic, a form of high-probability destination nodes. Authors show that in the presence of such hotspot gateway nodes, the scaling of a WMN decreases to the bounds presented in Equations 2.3 , 2.4. Also, Jun *et al.* validate the analytical results through simulations.

$$O(n) = \frac{1}{n} \qquad \text{per node capacity} \tag{2.3}$$

$$O(n) = 1 \qquad \text{network wide capacity} \tag{2.4}$$

Moscibroda *et al.* [29] corroborate some of the results from [28]. Authors pose their analysis for the context of Wireless Sensor Networks in which many sensors (source nodes) send data to sink nodes that perform data aggregation, for archival or further forwarding using different network resources. Moscibroda *et al.* find the same best-case capacity scaling (Equation 2.5) of $O(n) = \frac{1}{n}$ for arbitrary positioning of nodes. Authors provide lower bounds for the worst-case positioning of nodes (Equation 2.6). Finally, they also provide new bounds for random positioning (Equation 2.7). We note the exponential difference of results for arbitrary positioning between the worst and best cases (Equations 2.5 and 2.6).

$$O(n) = \frac{1}{n} \qquad \text{per node capacity} \tag{2.5}$$

$$\Omega(n) = \frac{1}{\log^2 n} \qquad \text{per node capacity} \tag{2.6}$$

$$\Theta(n) = \frac{1}{\log n} \qquad \text{per node capacity} \tag{2.7}$$

2.5 Capacity scaling under node clustering and multiple diversity mechanisms

Further improvement of the WMN capacity scaling can occur using the knowledge of nodes' locations on an advanced packet scheduling mechanism to reduce collisions [23]. However, such an organization can be challenging due to the need for global knowledge.

Modern implementations of networks apply a centralized network control paradigm such as SDN. Assuming this paradigm, global knowledge of the network is possible by the network controller node, turning the appropriate packet scheduling less challenging than in a purely distributed setting. Later we describe the challenges of implementing the centralized network control in unbounded size WMNs, due to long and high latency control paths.

We briefly side-step to the setting of data-center networks to describe the potential attributed to the SDN paradigm. Google showed the potential of capacity increase using

the centralized paradigm in data-center networks. In this setting, the network control was dominated by the distributed paradigm, such as the spanning-tree protocol whose goal is avoiding loops in the switched local-area networks (LANs) of data-centers. The spanning-tree protocol limits the traffic concurrency potential by disabling parallel links that cause loops.

In [30], Google presents an architecture for their data-center LANs using a massive increase in path parallelism using a *Clos* architecture associated with a centralized algorithm for path determination. The outcomes are increased scale in the possible size of their data-centers in the number of servers, massive throughput increase, and dramatically reduced latency, using general network equipment (low-cost). These properties allowed the effective use of large clusters of servers in their critical applications. In [31], Google applies the centralized control paradigm on the wide-area connectivity of their data-centers with the Internet, again showing benefits of the flexible choices of routes that maximize the efficient use of WAN links.

Franceschetti *et al.* conceive an analytical WMN model which enforces diversity in the time and code domains [32]. Under this new diversity settings, they show that the random WMN model could present a similar scaling to the arbitrary model using TDMA for packet scheduling and a pairwise coding on each path hop. The packet scheduling algorithm should associate long and short distance transmissions in parallel. The proof involves *Percolation* theory. The solution improves the scaling of the random model to: $w \cdot \sqrt{n}$.

MIMO is an advanced wireless mechanism that explores signal multi-path spatial diversity and coding diversity to promote parallel communications over a single channel frequency. Multiple antennas transmit different streams of data in parallel using orthogonal coding, and multiple receiving antennas decode the transmitted signal streams based on the different properties of the physical path they traversed and the orthogonal coding.

Özgür *et al.* [18] present an advanced and complex theoretical MIMO scheme which they claim to achieve nearly linear scaling on dense WMNs. The scheme has also nearly linear scaling in extended WMNs if the signal attenuation is low (free-space attenuation or $\alpha = 2$). It deteriorates to the previously known $\sqrt{n}$ network scaling when the attenuation factor increases: $\alpha \geqslant 3$. However, the complexity of the scheme is a limitation for practical utilization. It relies on a hierarchical organization for data flow and on *Cooperative MIMO*: clusters of nodes behave as MIMO arrays for transmission and reception, supporting parallel communications. Such collective MIMO operations demand pre and post cooperations (relying on TDMA for parallel transmissions) of the nodes to perform the MIMO encoding and decoding processes. The authors arrive at a scaling function in

the form $n^{\frac{h}{h+1}}$. For a large number of hierarchical steps h, this function tends to linear scaling.

Also, authors claim that the super-linear scaling of $O(n \cdot \log n)$ is not attainable by an *Information theoretic* prove. Therefore, they claim that the nearly linear scaling achieved by their scheme is nearly optimum.

Ghaderi *et al.* [33] contests the results by [18], showing that the solution's scaling function has a factor inversely proportional to the number of hierarchies h. Therefore, the linear scaling for a large h is not attainable.

We perceive from these two contributions the use of a hierarchical transmission scheme to specialize the local and remote communications accordingly. The next two papers explore the properties of inhomogeneous topologies to define partitioning schemes and apply hierarchical routing. Moreover, the logical clustering of nodes facilitates the definition of local and remote communication contexts.

Inhomogeneous node density in networks is a particular characteristic commonly found in human-built or natural systems [19]. Alfano *et al.* [19] assume inhomogeneous WMN topologies with denser areas (the node clusters) embedded into a general area of sparsely distributed nodes. They present results of $\frac{1}{\sqrt{n}}$ per-node throughput when the cluster node density is higher than a critical factor. Such results are similar to the bounds for homogeneous networks in [23] and [32]. However, in [19], improved packet scheduling and routing techniques proved to be a critical factor.

Liu *et al.* claimed a breakthrough result [20] in a three-dimensional inhomogeneous network setting with hierarchical routing. Exploring the topology characteristics of the inhomogeneous networks, they apply network partitioning centered around three-dimensional clustering of nodes to form the local routing regime. They claim to achieve nearly linear scaling enforcing spatial diversity through transmission power control and time diversity through TDMA on the intra-group communication (local routing regime). For inter-group (the remote routing regime), near linearity is achieved without power control due to the sparsity of the clusters. Node distribution is based on the Shot Noise Cox Process (a form of inhomogeneous Poisson node distribution). Authors used TDMA on an advanced routing and scheduling scheme for link diversity increase.

The previous two references build on specific characteristics of WMN topologies to identify node clusters. We envision imposing node groupings on any WMN topology for generality. Also, our design builds on the node clustering principle to support centralized control planes for the local routing regime within a node cluster. For the wide-area routing, we assume a distributed mechanism. However, yet relying on a programmable per-cluster controller that can implement routing consuming rich metrics.

2.6 Multi-channel, multi-radio WMNs (MRMC)

We turn now to the increase in frequency diversity by applying multiple-radios and multiple channels (MRMC). A constant in the MRMC setting is the need for a routing protocol able to explore the parallelism potential introduced. Raniwala *et al.* [34] present one of the first MRMC solutions integrated with a channel-aware routing for WMNs. Results show $8x$ (eight times) increase in the WMN end-to-end throughput for a two radio two channels per WMN node setting when compared with a single radio setting. However, experiments do not vary the WMN size in nodes neither evaluate different topologies for the fixed size of 100 nodes. Therefore, we cannot infer the capacity scaling trends of this solution. Furthermore, the traffic load assumes that only 20 source-destination pairs exist in a 100 node WMN, implying that the throughput study that does not evaluate the network throughput capacity.

The results from [34] rely on a centralized architecture for scheduling the packet flows between nodes and channels. In an *extended* topology WMNs, such centralization is not practical due to the existence on long control paths with the network growth that turns the control communication unreliable. However, the solution applies to cases of diameter bounded (and size bounded) WMNs.

Draves *et al.* [35] presents a packet scheduling for the MRMC setting. They conceive the ETT metric (derived from the link loss rate and link bandwidth) to dynamically evaluate the link quality. Using per link ETT, they compute a per flow path quality metric called Weighted Cumulative ETT (WCETT). The routing protocol MR-LQSR (Multi-Radio Link-Quality Source Routing) incorporates WCETT as its metric. Results showed that WCETT multi-radio outperformed the ETX metric (accounts only packet loss on links [36]) by $2x$, and shortest-path (hops only, no link quality accounting) by $3.3x$. Compared to itself in a single radio setting, WCETT shows a two-fold improvement (authors used the median throughput of each setting). The benefits of MR-LQSR were more representative for short distance paths (see Fig. 8 - the improvement of WCETT two-radios w.r.t. itself one-radio becomes < 40% for 5 or more hops). Again, a single topology was used in experiments, containing the fixed amount of 23 nodes. No information on the WMN capacity scalability was available. Experiments used 100 different node pairs for traffic generation, however, using a single TCP flow at a time and an idle interval between flows. Such experimentation choice minimizes contention and congestion and cannot represent the WMN capacity even for the single WMN size used.

Differently from [34], the design of [35] is of practical application, assuming a distributed packet scheduling. However, while the former showed a $8x$ increase over a single

channel setting, the latter showed only $2x$ improvement. We infer that the centralized control plane paradigm, although impractical in the specific setting described, has a significant potential for improvement of the capacity scaling.

Ramachandran *et al.* [37] adds to the MRMC WMN a new dimension of the dynamic channel assignment: a larger set of channels to choose from w.r.t radios on the nodes. Authors also apply a modified version of the proactive protocol OLSR using the WCETT metric from [35]. Using a larger set of channels w.r.t. to the number of radios, the *Channel Assignment Server* (CAS) performs a centralized channel assignment for radios on each node. The authors claim that their approach goes beyond reducing the WMN self-interference because it also accounts for the interference of other external and co-located radios using the same set of channels. Experiments show an improvement of throughput of the proposed solution of $3x$ over a single-radio solution. Experiments used a single WMN topology size of 30 nodes and only four different topologies. Results varied up to 50% from the highest to the lowest performing topology. Also, the algorithm is specialized to the used topologies because it starts its scheduling from the *gateway node* of the topologies (a specific optimization considering that the gateway node will attract the WMN flows). Once more, authors do not present any evaluation of the capacity scalability.

Buddhikot *et al.* [38] present a new link quality metric to support WMN path selection in routing protocols. Authors claim that their metric is interference aware (iAWARE), providing an improvement over WCETT. iAWARE captures the interference of competing flows on the WMN. Authors applied their new iAWARE link quality metric on a reactive routing protocol derived from AODV: AODV-MR (Advanced On-Demand Distance Vector Multi-Radio). This new protocol supports WMN nodes in the MRMC setting. No evaluation is provided in this patent application.

Subramanian *et al.* [39] continue with the setting of dynamic channel selection in WMN. They evaluate both centralized and distributed channel assignment algorithms. The work does not inform which type, proactive or reactive, or specific routing protocol was used. For a two-radio setting and 12 channels to choose from, they showed a $2.5x$ improvement both for the centralized (not practical) and distributed channel assignment algorithms over a single radio setting. With eight radios per node and 12 channels to choose from, they present the highest improvement of $8x$ over a single radio, using a centralized algorithm. For the distributed algorithm, the top improvement of $5x$ occurs at five radios. Once more, a single sized topology of 50 nodes was used. No capacity scaling information is provided. Beyond channel assignment, the scheduling of packets through paths in a multi-hop setting should have been described, however, no information

was provided in this matter.

Sajjadi *et al.* [40] revisits the channel assignment problem in WMNs focused on reducing the switching delay. Their approach of switching delay has only indirect correlation with WMN capacity. Therefore, this work is less relevant to our proposal. However, Sajjadi *et al.* bring one important contribution: they turn practical in WMNs the assumption of centralized algorithms such as channel selection by applying the SDN paradigm. Furthermore, they use out-of-band SDN management and a small topology of 7 nodes which also limit the value of their work for our objectives of capacity scaling.

In the MRMC studies, the effective utilization of multiple channels implies the use of multiple radios per node. Such a requirement seems expensive for a large WMN setting. A partitioned WMN approach can use MRMC differently: to isolate partitions of nodes in the physical layer, allowing the evaluation of the WMN capacity scaling in the sub-2 per-node radios range in average - $(1, 2]$. Only nodes which interconnect partitions require two radios, constraining the increase in cost. Such an assumption seems more practical for the CWNs scenario but demands the evaluation of its capacity scaling potential. Additionally, the partitioned mode supports hierarchical routing architectures as applied in solutions which showed improved scaling [18, 20]. Furthermore, hierarchical routing supports the practical application of centralized network control to the local routing regime assuming the enforcement of a *diameter* constraint in partitions.

2.7 Autonomic Computing

We introduce to the partitioning problem in WMNs the application of autonomic computing, aiming at supporting unbounded scale WMNs. The concept of autonomic computing, or processes that are independent agents showing a set of self-* behaviors in a distributed system, was initially introduced by IBM [41, 42]. Given the lack of formal definition, different interpretations emerged. Later, a somehow standard definition for self-organizing (Self-Organizing (SOrg)) and self-healing (Self-Healing (SHeal)) autonomic behaviors developed. SOrg implies controlling (maintaining, improving, recovering) properties in the presence of an external process that adds/removes nodes on the autonomic system. SHeal consists of existing nodes adding/removing edges on the system in an attempt to recover to a previous condition [43]

Given the lack of standard definitions, Berns *et al.* [12] proposed a formal definition for forms of autonomic computing or the self-* behaviors. First, they categorize internal attributes controlled by autonomic agents as *safety* and *liveness* properties. A *safety* property is a condition that should not be violated while a *liveness* property is a condition

that should eventually be reached. Second, Berns *et al.* define types of tolerance to *internal* (from agents) or *external* (environment change or from adversaries) actions as *masking*, *non-masking*, *fail-safe*, and *graceful degradation*. *External* actions relate to changes in the environment or from external adversaries. *Internal* actions derive from the agents in the autonomic system. A *masking* tolerance implies that the safety and liveness properties are not affected by an action. *Non-masking* tolerance occurs when a set of external actions can violate safety but not liveness properties, and eventually, the safety properties are recovered. A *fail-safe* tolerance implies compromise on the liveness but not the safety properties. Finally, *graceful degradation* occurs when the action affects safety but not liveness properties, and the recovery occurs to a configuration that is a weaker form of the desired safety properties.

Furthermore, agents rely on local information: the agent's internal state and the state of its direct neighbors.

Using definitions in [12], we describe self-* behaviors relevant to this work. A Self-Stabilizing (SS) system starts in an arbitrary configuration, (*e.g.*, no nodes existing in the system), and recovers to a legal state (*e.g.*, nodes have their safety properties in valid states). The system remains in this configuration thereafter [12].

A SOrg system maintains, improves, or restores safety properties in the face of actions related to nodes entering/leaving the system [12].

A SHeal system admits a temporary violation of their safety given actions that are not related to nodes entering/leaving the autonomic system, to later return to safety by adding/removing edges to the system [12].

Our *SmartOrg* agent model is *SOrg* given its operation under the addition and removal of nodes. We claim that *SmartOrg* is also Self-Configuring by applying the Lemma 5 in [12]. Furthermore, we claim that our *SmartHeal* agents are SHeal given their actions adding/removing connections (edges on the WMN topology graph) between partitions. Finally, we can call both agents and their combined operation a form of *SS* as they converge to stable topology solutions.

Self-Stabilization is a possible representation of our experiments on which nodes enter the system on every epoch until a maximum. For every epoch, a self-stabilization occurs: nodes connect to partitions, the degree of nodes, and the diameter of partitions - safety properties - stay below the maximum accepted value. However, SS does not require that the configuration convergence must involve nodes entering or leaving the system, and we cannot guarantee convergence for initial epochs. Moreover, the control of properties concurrently to an external process of the addition of nodes is characteristic of SO. Therefore, we assume the latter.

2.8 Partitioned WMNs: reduced overhead and hierarchical routing

Hierarchical routing architectures rely on the logical or physical partitioning of WMNs [17]. The principles for hierarchical routing are a) nodes are aggregated in clusters; b) cluster head nodes are elected; c) gateway nodes interconnect clusters. The intra-cluster (local routing regime) and inter-cluster (remote routing regime) communication can assume different routing mechanisms. Intra-cluster routing can be proactive (pre-computed routing tables), and inter-cluster can be reactive. Such an approach minimizes the proactive routing protocol overhead. Such overhead can be $O(n^2)$ [44, 45], but in a hierarchical setting n is reduced to c (the number of clusters), much smaller than the WMN size n. For long-distance flows, the path set-up has higher latency. However, this approach does not penalize short distance flows (intra-cluster). On the limitations, the authors of [17] cite that the complexity of maintaining the hierarchy and electing the cluster heads can compromise the performance gains.

Our autonomic agent-based approach is a contribution to dealing with WMN partitioning, especially relevant for large-scale settings. Our design also solves the aspect of electing a cluster head, our partition origin node.

In their summary of hierarchical routing protocols, Akyildiz *et al.* do not conceive the association of clustering (WMN partitioning) and using different per cluster channels to improve scaling: *physical layer segmentation*. The authors' intention seems to be to decrease the routing protocol overhead as an indirect measure of improving the capacity of WMNs. We build on the idea of physical layer segmentation with minimal cost increase, assuming an average number of wireless interfaces per node in the range $[1, 2)$.

Ying *et al.* [46] implements an hierarchical version of the well known OLSR [47] WMN routing protocol: HOLSR. The goal is to improve the scalability of the routing protocol itself by reducing the volume of messages exchanged.

We consider out of our scope the hierarchical routing solutions that apply a combination of wireless and wired interfaces (generally Access-Points with wireless and wired interfaces) such as [48] and [49].

Li *et al.* [50] present an algorithm for partitioning the network control effort in the SDN paradigm, which improves network control scalability. There are no assumptions about the network being a WMN. However, the topologies presented suggest a meshed and wireless topology with fixed nodes. Their goal is to segment the network in partitions, each managed by a different SDN controller, to keep the controlling load within bounds. Therefore, the authors do not evaluate improvement on the network capacity by

the partitioned control. The controllers are predefined (possibly specialized nodes) and cannot be changed. It is assumed an out-of-band control network for interaction between controllers. A *Master* controller executes part of the segmentation algorithm, and the second part is delegated to the *Zone* controllers. Therefore, it is a centralized network partitioning scheme. This centralization to implement the partitioning of WMN control limits the solution to medium size WMNs.

So far, all the above approaches operate on the segmentation of the network control plane. The benefits were lower routing protocol overhead in the distributed paradigm and reduced latency for path determination and balancing the control workload on the SDN paradigm. Therefore, they aim at improving specific factors of each paradigm. None evaluated the impact of the segmentation mechanism on the capacity of the data plane.

We approach the partitioning problem in a distributed scheme in which autonomic agents embedded into the network nodes perform the partitioning at the physical and link layers. To partition the WMN, we do not assume any out-of-band control network, which would be impractical in a large-scale WMN setting. We build on the principles of autonomic computing, relying on local information to the mesh nodes: internal and neighbors.

2.9 Packet routing and scheduling techniques on WMNs

Reducing the bandwidth demanded by WMN routing protocols increases the capacity for application flows. Also, the routing control workload might be reduced. Developing *efficient* protocols w.r.t. the load they impose to the network, the routing protocol overhead, is one form of capacity improvement. Furthermore, *optimized* routing protocols that are aware of the forms of link diversity enforced and improve flow parallelism is another form of capacity improvement.

We will loosely use the terms *routing protocols* and *routing mechanisms* to describe the way nodes make decisions about the forwarding of packets through the WMN. The former comes from the classic study of distributed routing protocols, and the latter assimilate the more recent SDN paradigm.

The implementation of WMN multi-hop forwarding protocols requires observing many features [17]:

1. Routing metrics - many protocols assume the hop-count as a forwarding metric. However, implementations should also consider wireless link quality-related metrics. Such metrics include link speed, link congestion. RTT (round trip time) can be a proxy metric for the lower layer metrics mentioned. Metrics can be related to other

layers and must be provided to the routing layer (cross-layer).

2. Fault tolerance to link failures (self-healing) - encompasses how quickly can a routing mechanism identify and solve a flow path failure.
3. Load balancing - the ability to implement parallel paths in the WMN
4. Protocol scalability - how efficient will the protocol be as the WMN scales in size. Some protocols could be appropriate only for small WMN sizes. Additionally, some protocols might demand excessive bandwidth for their operation (protocol overhead) as the WMN scales up in size.

Distributed routing protocols can pre-compute routing tables (proactive) or operate on-demand (reactive). The proactive approach demands a continual exchange of information that can be in a squared relation with the number of nodes (worst case - $O(n^2)$ [44], [45]. However, routing decision has low latency because of the pre-computed routes. Furthermore, the variability of application flows (different flows regarding the source, destination, or application type) also does not affect the overhead because there exist pre-computed routing tables on each mesh node with routes for all destinations.

On-demand distributed routing protocols (reactive) exchange information when a router needs to forward a new flow. Therefore, protocol overhead is a function of both application flow variability and network size. The variability indicates how frequently will the protocol flood the WMN looking for route paths. The WMN size indicates how many nodes might be involved in such exploration by flooding. One compromise made on the on-demand routing is the higher latency for path setup, which can hamper WMN performance in a short-living flow scenario.

In addition to the higher latency, Belding-Royer *et al.* [45] claim that reactive routing protocols present *scalability difficulties* when the network has many nodes due to the flooding process and due to the need of receiving routing error messages back in the source nodes.

Santiváñez *et al.* [51] further segment the *proactive* class of distributed routing protocols. Proactive protocols are based on *link-state* or *distance vector*. Santiváñez *et al.* focus on link-state protocols, organizing them in two classes given the approach used: *efficient dissemination* and *limited dissemination* proactive protocols. Although both aim at reducing the protocol overhead, the efficient dissemination protocols distribute control messages throughout the entire network, however, using more efficient methods than the traditional flooding. In this class we find the protocols TBRPF [52], STAR [53], OLSR [47]. In the *limited dissemination* class, protocols restrict the scope of control messages updates in space and time. Examples of limited dissemination are hierarchical link-state [54], GSR, and FSR [55].

Since its introduction in [47], OLSR has become a de facto standard for WMNs routing with many improvements proposed for it. We assume that the efficient dissemination has proved a preferred technique for WMNs and their fixed-nodes characteristic.

Santivfiez *et al.* [56] provide insight on the scalability of distributed routing protocols to network size, traffic variability, node mobility, and network load. Authors provide analytical asymptotical bounds for the scaling of a set of types of routing protocols:

a) PF - Plain Flooding, reactive;
b) DSR - Dynamic Source Routing, reactive;
c) ZRP - Zone Routing Protocol, reactive and limited dissemination proactive parts;
d) SLS - Standard Link State, proactive;
e) HierLS - Hierarchical Lisk State, proactive, efficient dissemination (OLSR is example);
f) HSLS - Hazy Sighted Link State, proactive with limited dissemination;

We provide the bounds without the focus on mobility. We use the following notation: λ_t is the traffic load in the network in bits per second, λ_s is the variability of flows (the rate of new flows) in new flows per second, n is the network size. δ captures the increase in the average route path length with the increase in network size. We provide results ordered by the protocol overhead scalability w.r.t. network size from worst to best.

1 DSR: $\Omega(\lambda_s \cdot n^2 + \lambda_t \cdot n^2 \cdot \log_2 n)$
2 PF: $\Theta(\lambda_t \cdot n^2)$
3 ZRP: $\Theta(\lambda_s \cdot n^2)$
4 SLS: $\Theta(n^2)$
5 HierLS: $\Theta(\lambda_t \cdot n^{1.5+\delta})$
6 HSLS: $\Theta(\sqrt{\lambda_t} \cdot n^{1.5})$

The results explain the popularity of link-state based protocols such as the original OLSR and its improvements due to their better scalability. Although HSLS scales better than HierLS, their routing decisions are sub-optimal, especially for long-distance paths because it limits the spreading of route information in space. Therefore, better overhead scaling might not represent better capacity scaling of the WMN. The DSR protocol does not perform route caching. Therefore, its overhead can be lower with caching, assuming that long-standing flows would benefit from caching of routes. Specifically, the component dependent on the traffic load λ_t would improve.

No type of distributed routing protocol studied by Santivfiez *et al.* presented linear scaling of the protocol overhead regarding the network size as expected from the SDN paradigm. However, control overhead for routing in the SDN paradigm should have a component dependent on the flow variability, similarly to the reactive protocols DSR and ZRP. The difference is that in the SDN paradigm, such component is shown to be a linear relation w.r.t the flow variability as shown by [57].

Putta *et al.* [58] compare through simulations the performance of classic reactive (DSR, AODV) and proactive (OLSR) protocols, assuming node mobility. They evaluate packet delay, packet delivery ratio, and protocol overhead. The packet delivery ratio is a proxy for the capacity of the WMN. They varied the network size from 125 to 200 nodes in 25 steps and the load from 10 to 60 connections in 10 steps. Unfortunately, the work only provides results for the 200 nodes WMN size regarding different loads. Under this mobile setting, the authors conclude that OLSR has better performance to CBR traffic such as video and voice due to its lower route setup delay; however, imposing higher control traffic overhead. The reactive protocols (both DSR and AODV) performed better regarding packet delivery ratio for non-delay sensitive applications such as file transfer. The lower overhead facilitated a packet delivery ratio of 80% for 200 nodes, while OLSR performed below 50%.

These results seem to contradict the analysis in [56]. The explanation is that the proactive protocols suffer when mobility is representative in the network (MANET, VANET).

In [59], the authors present a multi-path routing version of OLSR: MP-OLSR. Due to the assumption of mobility, we refer to the design of the protocol and disregard results. The following research will compare results with this protocol. Uemori *et al.* [60] describe a multi-path, node-disjoint routing-ID based scheme for WMNs. The study compares the protocol overhead with MP-OLSR. MP-OLSR presented a lower overhead.

In the SDN paradigm, data forwarding resembles an on-demand (reactive) routing protocol because paths are computed when necessary. However, the WMN SDN also demands the continual exchange of information to keep status control of mesh nodes by the SDN controller. Therefore, similarly to the ZRP, the SDN based routing mechanism has both reactive and proactive characteristics. The status control traffic (proactive component) will be a linear function of the network size - O(n) - because it involves a single destination for all nodes: the SDN controller. The second component of the control traffic (reactive) is responsible for path setup. It is a function of the variability of the applications' flows (the rate of new flows). However, no flooding is required in a WMN SDN. The SDN controller keeps network topology information based on the proactive status messages received, which allows defining route paths per new flow communication

without network overhead. In turn, a computational overhead appears. Dely *et al.* [57] suggest, through limited experiments, that the reactive component of the SDN routing mechanism overhead is a linear function of the flow variability rate.

Dely *et al.* [57] are the first to present an implemented architecture to apply the SDN paradigm to WMNs. The architecture uses a hybrid approach: the SDN control channel is implemented in an out-of-band fashion using (1) a separate SSID on the wireless interface of the mesh nodes and (2) the distributed routing protocol OLSR for the routing on this control channel network. Therefore, this out-of-band control channel approach lacks the benefits of the SDN paradigm and adds complexity to the architecture when compared to a purely SDN network (control and data channels in the same network, in-band control). Besides, the topology discovery is dependent on the distributed routing protocol OLSR used in the control channel, which adds to the interdependence and complexity.

Dely *et al.* [57] compare the protocol overhead under varied flow dynamicity. They vary the number of new flows in the WMN, effectively varying the OpenFlow rule creation rate in a WMN. The control overhead of OpenFlow increases linearly with the rule creation rate. The same experiment using the proactive protocol OLSR had a constant routing overhead. However, the authors account for the control traffic as the control communication in all hops in the WMN. Therefore, the same flow is accounted for many times, given the number of hops used. This accounting of traffic is different from the capacity scaling approach, which considers only each source-destination flow. Additionally, authors do not evaluate the scaling of the overhead with the network size.

We understand that the flexibility provided by SDN-based WMNs is a crucial improvement for the WMN architecture. Our contribution to this problem setting aims at supporting the SDN paradigm into WMNs in large-scale settings, not yet portrayed in existing research.

2.10 Community Wireless Networks and their application of WMNs

This section discusses aspects of CWNs: their motivation to exist; the structure of their networks. We refer the reader to an Appendix Section A.1 with information about specific instances of CWNs.

2.10.1 The nature of CWNs

In terms of motivations to exist, CWNs have different general goals: a platform for experimentation by enthusiasts (a testbed: FunkFeuer, Austria; Freifunk, Berlin), a plat-

form for local and remote (the Internet, the public telephony system) connectivity (in Africa, the U.S.), to fulfill the desire for social integration and social community building (Athens, Greece). These characterizations, however, are not homogeneous; instead, they represent primary motivations of a CWNs.

In [61], a panel of specialists and leaders of CWNs discussed the present and future of this social organization form centered on the goal of *connecting* members. Panel members also characterize fundamental differences in CWNs. In Europe, some networks are built and operated mostly by enthusiasts interested in learning and exploring networking technology - the hacker based CWN aiming at education. Such type is not interested in subscription payment and serving regular users. Newcomers need to educate themselves to be able to install and maintain their nodes. In such areas, the CWN is not crucial for connectivity due to the existence of paid connectivity services.

In rural areas of the U.S., Nepal, and Africa, a different type of CWN exist that provide service to users as the only viable alternative to connectivity (either because of the lack service providers - rural U.S. - or lack of funds by users to consume available connectivity services).

Athens Wireless Metropolitan Network (Greece) CWN represents a third case in which the CWN was a means to support the desire for social connection and integration and community formation: "... we exist even if the Internet does not exist ... everybody creates services and provide services to the community" [62]. An expert in the field of social organization, Primavera de Filippi claims [63] that WMNs have their maximal impact on creating a social organization in communities that goes beyond the technical education and Internet access benefits.

In the past, CWNs have not spread through the U.S., or most of the organizations that started have died. The fundamental nature of European CWNs is of members dedicated to the realization of the network, whereas in the U.S., funding sources were critical for expansion [61]. Additionally, in the U.S., a turn-key solution is expected by new community members. The complexity of setting up a CWN node was a vital barrier: "... in the U.S. many people will have more money than time ... it will be easier to expand having the chance to buy an expensive device than the required time to learn the technology ..." [61]. Furthermore, in the U.S., CWNs, initially free and volunteer-based, migrated to model ISPs that behave as regular companies: hierarchical organization, leaders, and employees. In Europe, CWNs rely extensively on member contribution, diverse technical solution, and distributed decision making. We argue that such a *distributed* and *networked* organization resembles more the relationships in cities based on diversity and cooperation [6–8] than the hierarchical structure of companies [16].

Panel members in [61] agreed that the competition of Internet Service Providers (ISPs) to CWNs is mostly an issue when the latter behave similarly to ISPs. The other types of CWNs have their specific motivations to exist beyond Internet access. Furthermore, communities are diverse in their intent. However, the classification in the three types aforementioned reflects a perception of their general intent.

For the future, CWNs will continue to have, while having different reasons to exist, the critical function of being a competitor to established service providers. In some instances, the start of a CWN was followed, with a lag of years in some cases, by an expansion on the offer of network service providers. Therefore, the creation, continued existence, and improvement of CWNs go beyond being a testbed for enthusiasts: they are a competitive force to service providers and a safeguard to censorship [61].

2.10.2 The topology structure of CWNs

The distributed and inherently redundant nature of WMNs, in contrast with the hierarchical form of other wireless networks such as cellular wireless, facilitates the construction and operation of a network by communities. However, CWNs use the WMN architecture as one of many connectivity mechanisms on their topology implementation. In general, CWNs designers restrict WMNs to *islands* of nodes which in turn use other mechanisms to interconnect to the CWN at large [62, 64] given the capacity and geographical scalability limitations of standard single-channel WMNs. Other wireless link-layer methods used in CWNs are PtP[1] and PtM[2]. Larger CWNs apply fiber-based backbones to cope with high-capacity, long-distance connectivity [65].

The use of PtP and PtM links imply manual configuration (the choice of which extension point to use) and specialize which type of nodes serve as extension points to the network. In the PtP and PtM, one of the nodes (the first P side) allows extending the network while the other is a leaf node. Furthermore, capacity benefits will not come from the link-layer isolation provided by the PtP and PtM links. An increase in capacity requires the use of orthogonal frequencies in the links, adding to the configuration complexity.

Given a node placement, deciding how to *induce* a network topology using the mentioned types of linkage is a hard problem. Finding a graph partitioning scheme (a set of PtP, PtM combinations) that maximizes an objective (connectivity, capacity, redundancy - assuming there is an objective way to determine these high-level properties) is an NP-Hard problem (Graph Partitioning) [66, 67]. Furthermore, the problem of attributing frequen-

[1] Point-to-point: one node on WiFi AP mode (Infrastructure BSS) and a single remote node on WiFi station mode.

[2] Point-to-multipoint: one node on WiFi AP mode and a set of remote nodes on WiFi station mode.

cies to minimize interference (Graph Coloring problem) is an NP-Complete problem [68]. Finally, it is essential to account for dynamicity: as nodes fail, the current design might no longer be the best solution, requiring a new design.

The manual configuration in CWNs is, in fact, opportunistic: the network starts small (or even minimal) and grows on-demand. As more nodes want to join, they connect to the network at the closest point. In effect, no optimization plays a vital role in the link design level [65].

In this research, we operate on the WMN's multipoint link connectivity[3], comprising short-distance links (when compared to PtP and PtM). In a multipoint link setting, any node is an extension point. An *autonomic agents* approach operating in a distributed manner avoids the mentioned complexities if agent design optimizes the objectives such as capacity and supporting centralized control. Also, such an automated approach supports dynamicity and can dramatically reduce efforts of manual configuration.

2.11 Experimentation platforms and challenges

The modern design of communication networks blurs the line between applications and networks. Modern networks rely on complex software on its operation, such as in the SDN paradigm (Software-Defined switches and controllers). Another example is the LTE standard for wireless cellular networks with its MME (Mobility Management Entity), Packet Gateways at the network core, network functions (NF) for network packets processing chains such as firewall processing, address translation, accounting. Moreover, the direction for 5G wireless networks is to turn software even more prominent on its architecture: Cloud-RAN, SDN-based back-haul networks, distributed packet gateways based on SDN-like architectures with programmable switching data-planes and NFs as software in commodity servers. The motivation is to increase efficiency, and flexibility to dramatically accelerate innovation in networks, a movement modeled after the successful evolution of large-scale applications based on cloud computing. Finally, an increased pace of innovation in networks requires more frequent evaluation, turning strategies for the efficient evaluation of modern networks a critical factor.

Furthermore, the reverse direction also holds: large scale applications behave as networks. Disaggregation induces the *microservices* design (e.g., Netflix [69]), which resembles a network of specialized functions. This design promotes horizontal scaling, flexible evolution (concurrent execution of different versions of software), inexpensive redundancy. Also, NetChain [70] is a state-of-the-art, distributed consensus solution that provides a

[3] All nodes configured as Independent Basic Service Set (IBSS) WiFi, forming a mesh-like topology.

sub-RTT agreement. NetChain turns a chain of Software-Defined switches into its computation environment to implement its proposed consensus protocol. Therefore, testing and debugging applications in these new settings is not dissimilar from emulating large-scale networks.

Our specific focus involves studying the capacity scaling characteristics of modernized WMNs, whose topologies are autonomically manipulated by mesh node agents. In this setting, we focus our interest in wireless networks, and we add the new constraints of experiment sets that must vary the size of WMNs from a small to a large number of nodes (*e.g.*, 2 - 1000s), on each experiment, nodes operate at their maximum possible throughput. Moreover, topologies must support dynamic change according to the decisions of agents. Aiming at WMN modernization, we assume the SDN paradigm on WMNs design, involving the integration of pre-existing complex software into the network control plane (SDN controller and protocol) and data plane (SDN switches, programmable data planes). Such software applications are not available in network simulators or simple to re-implement as simulated models [71, 72].

Existing alternatives for the experimental evaluation of modern (software infused) wireless networks based on real testbeds, simulation, or emulation impose tradeoffs between the desired objectives. Following, we elaborate on these methods.

Real-time testbeds provide precise reproduction of wireless systems. Also, testbeds admit the reuse of existing software. However, scaling testbeds to support the study of large networks is cost-prohibitive. Also, testbeds are not flexible regarding emulating different distances of wireless links and variations of topologies. Finally, the shared use nature of testbeds might become an issue regarding extensive experimentation, such as in the capacity scaling studies.

Simulations also offer a precise representation of wireless networks, supporting the accurate study of capacity and high flexibility regarding physical distances and topologies (ns-3 [73], others). Simulations are also inexpensive in terms of CAPEX (capital needed to build topologies) on scaling to large network sizes. However, the reuse of existing software is severely limited. Any software must be transformed into models in the simulation environment, which is error-prone and cost/time demanding. Moreover, simulations tend to consume a significant time for computation of the wireless models.

Fall *et al.* [74] proposed as a solution an experimentation framework based on *emulation*: network simulators simulate wireless stacks; real software runs in virtual machines implementing the processing capacity of nodes. We call the method in [74] real-time emulation. However, the approach of the solutions in this category for modeling the wireless environment varies from minimal (Core [75] - on-off link connectivity) to only emulating

the characteristics of individual links (Mininet-WiFi [72]) without accounting the interaction between links such as contention and interference. In this tradeoff space, the study of capacity is not accurate. Moreover, the split of the node's *wireless system* and *computing system* in two different environments introduces the problem of time synchronization: each environment has its independent timing system [76–78].

Mechanisms for time synchronization amongst networked systems exist such as Network Time Protocol (NTP) [79] or other higher precision approaches [80–82]. The fundamental difference between those approaches and the time synchronization on emulation is that the former assumes systems that should operate at the same clock but subject to small clock drifts. In emulations on our settings, the simulated clock is fundamentally different from the real-time clock, and such difference is highly variable, depending on the resulting computational load of simulation and systems emulation.

Some network simulators such as ns-3 [73] and its real-time scheduler attempt to support emulation by synchronizing its internal simulated time with the external computing system's real-time. However, such synchronization works in a single direction in practical terms: the real-time event scheduler of the simulator does not allow events to be processed *before* the corresponding real-time: *no anticipated event execution*. Events might be processed with a delay concerning the real-time: *possible delayed event execution*. Such delay is due to the computation complexity of network models and a large number of events in the simulated network models required for the capacity scaling evaluation. Following we detail our analysis of ns-3 and its real-time scheduler since we applied it our our experiments and it is also used as part of other solutions [83–85].

In practical terms, any reasonably complex wireless topology will imply delays to packets due to model computation efforts and not inherent to the models themselves. In our preliminary experiments using ns-3 in emulation mode (ns-3's real-time scheduler), we confirmed such expectation.

Such a scenario becomes critical when emulating WMN topologies because packets can transition from real-time computing containers to the network simulator many times, given the path length in the number of nodes in a WMN. Furthermore, the number of simulation events in a WMN network tends to scale with the square of the number of nodes: $O(n^2)$. The quadratic scaling occurs since every transmission in an abstract spectrum channel in the wireless simulation models should be inspected against all other nodes in the spectrum for reachability determination. If all nodes transmit in a given time frame (unbounded demand), the quadratic relation appears in the time frame. In the study of the scaling of WMNs, nodes attempt to communicate at their maximum (all nodes, at all times, transmitting) to identify the capacity limits. Figure 2.1 shows our

practical confirmation of the quadratic scaling of events for WMNs.

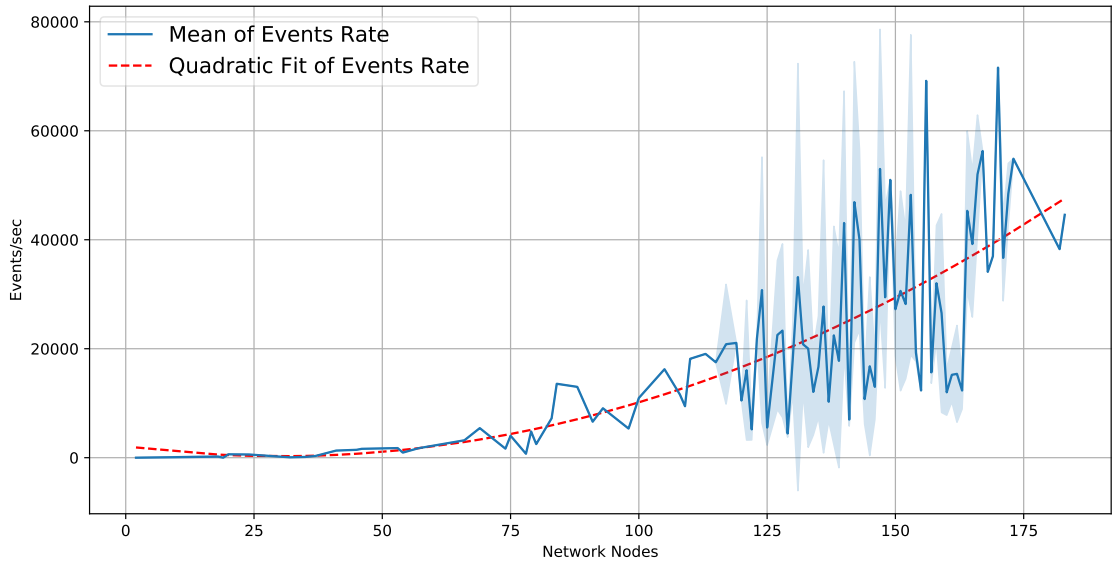


Figure 2.1: Scaling of the rate of simulation events w.r.t. the number of wireless nodes. Mean values of the rate of events for three runs with seeds 10, 20, 30.

Fontes *et al.* [86] attempt to reduce the simulation overhead and allow the simulation to run within the real-time constraints. Their approach consists of moving the processing of the network data plane outside the simulation system and executing the control plane in the simulation environment. However, if the evaluation intention goes beyond protocol design and evaluation (as it occurs in the case of a throughput capacity scaling study), the approach is insufficient.

Another attempts intend to increase the event processing capacity scalability implementing *parallelism* into simulators. However, improvements will represent at most a linear function on the number of parallel processes p of the simulator - $O(p)$. Moreover, in practical terms, the number of parallel processes p tend to be much smaller than the number of nodes: $p << n$. Furthermore, the literature on network simulation describes that linear scaling with parallel processing is hard to achieve. The best-known solutions for parallelism, such as using *event time lookahead*, only provide scaling for specific topologies which have significant propagation delays on the transmission medium. The lookahead-based approach achieves very low scalability for *wireless* and *bus* topologies in which the propagation delay is minimum, close to zero [78]. The conclusion is that parallel processing in network simulators is not a definitive solution to avoid introducing delays when operating in emulation mode.

Other attempts exist in the other direction: expanding the real-time (*time dilation*) perceived by the emulated environments (VMs, containers). We call this method *ex-*

tended time emulation [83, 85, 87–89] attempts to reproduce - not simply synchronize - the time perceived in the emulated computing environment by sourcing it from the simulated environment, reducing the effort and risk on porting existing software applications to a simulated environment. The time dilation mechanism also increases the relative computation power of the simulation system w.r.t. its nodes.

Gupta *et al.* [87] proposes *fixed time dilation*. It relies on full virtualization to isolate the time references in the duplicated kernel stacks of the VMs. This approach also requires anticipating a fixed time dilation factor that either will render the simulation inefficient (conservative choice) or require an iterative approach for tuning the dilation factor. To improve the latter, SliceTime [83] implements a dynamic adjustment of the dilation factor. These two last examples rely on full virtualization to isolate clocks, characterizing a high overhead for emulation of a large number of nodes.

To scale to medium-size networks, [83] assumes that a small subset of nodes requires virtual environments, given that its time extension scales linearly with the number of VMs. Furthermore, authors assume a precision of hundreds of microseconds to 1 millisecond, which could yet interfere in the timing of the TCP protocol algorithms, turning SliceTime inappropriate for LAN scenarios with RTTs in the same order of SliceTime's time precision. SliceTime splits time evolution into slices and makes each element wait for the others at the end of the time slice (a barrier-like mechanism). Furthermore, no guarantees of time convergence exist within a time slice, such as a time interpolation, taking into account the simulated time speed. A positive aspect of SliceTime is its complete transparency to the guest OS in the VM due to manipulating time on the hypervisor (Xen) layer. Limiting the nodes with VMs is not a valid assumption for our setting. We need a SDN software switch on each mesh node.

ns-3's DCE (Direct Code Execution) [84] and the *Timekeeper* project [85, 88] address the problem mentioned above of time synchronization in a network emulation setting for large-scale wireless networks. None rely on full virtualization: Timekeeper uses lightweight virtualization (containers), and DCE turns all application code and kernel stacks into libraries running alongside the simulator process. We identify shortcomings as follows.

DCE uses the concept of a *library-based operating system*. DCE allows the use of user-space applications and a limited set of kernel-space modules in experiments. However, all application code needs to execute as libraries, requiring code recompilation for compatibility with DCE's specialized loader. Also, DCE manipulates the scheduling of the apps (execution inside the simulator), apps must restrict their interaction with the OS to the POSIX API, and DCE implements significant modification of the library OS kernel. The approach of running applications and kernel code as libraries, instead of enabling

an emulation (real software execution integrated with a simulated network stack), brings the real software to a simulated environment: app code and kernel stack become part of the simulation process. Such an approach also limits computation power used on the emulation due to the incorporation of the node's computation resources into the simulation process. On the upside, DCE developers claim that a broad set of time system calls are available for user-space apps. In [90], the authors claim that DCE has not yet been evaluated under large-scale wireless settings and adjust DCE to explore multiple cores of the host emulation system.

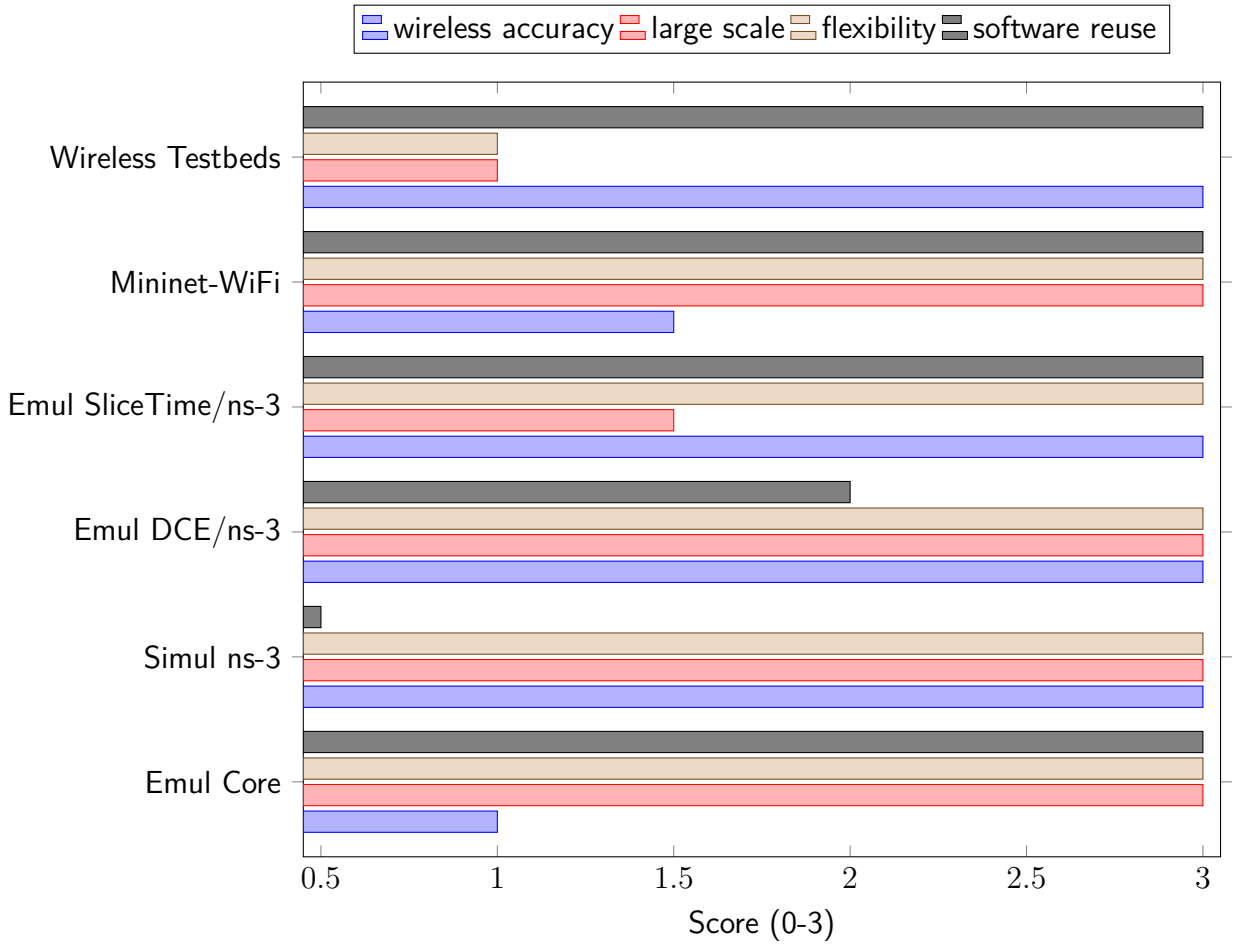


Figure 2.2: Comparing solutions to the requirements for the evaluation of modern wireless networks. Scores for the support of a requirement: 0-nonexistent, 1-minimal, 2-partial, 3-full.

Timekeeper limits the integration into an emulation to user-space applications. Furthermore, it provides an insufficient set of time-related system calls. Therefore, one needs, at least, to investigate each application's usage of time-related system calls to verify the feasibility of the intended emulation. An additional downside potential, the pro-

cesses of the user-space applications are not scheduled by the Linux scheduler, rather by Timekeeper's kernel modules. On the upside, Timekeeper allows unmodified user-space applications if they commit to the resources provided by Timekeeper.

VT-Mininet [89] integrates time dilation (a Virtual Time) into the well known and popular Mininet-HiFi [91, 92] emulation platform. Their approach for creating a virtual time to the lightweight containers implemented by Mininet-HiFi builds on the principles defined by the Timekeeper project [85, 88]. However, VT-Mininet does not support the emulation of wireless networks. Furthermore, although improving on Timekeeper, it yet does not expose the virtual time to all kernel functions such as the Linux *tc* (Traffic Control) kernel packet scheduler used for controlling the emulated link capacities.

In summary, reusing pre-existing software requires low to no modification on the target software, and requires a computation environment such as full or lightweight virtualization (VMs or containers). While VMs support time isolation from real-time, they impose prohibitively high overhead for large-scale experiments. Furthermore, time synchronization to simulated time is challenging at large scales due to the hard isolation of hypervisors. On the other extreme, containers share the kernel environment of systems, implying an nonexistent standard mechanism for time isolation from real-time (especially for in-kernel functions).

Below we provide guidelines to the design of an emulation framework involving exploring the upsides of both Timekeeper and ns-3's DCE while solving shortcomings to our evaluation setting. One key differentiation regards the support of the emulated time inside the kernel task (Item 2). Figure 2.3 provides an overview of the different types of tasks executed inside the Linux kernel and their entry/exit points.

1. Supporting kernel modules (kernel-space) and user-space applications. No modifications must be required to user-space applications and a large set of time-related system calls must be supported.
2. The code of kernel modules can demand modification. However, a guideline for adapting kernel code will simplify this task. The kernel must also be able to discriminate between kernel asynchronous tasks (Hard-IRQs, Soft-IRQs, Tasklets, Workqueues) and user-space processes in kernel synchronous execution, providing the appropriate perception of time. See Figure 2.3;
3. Applications must use the regular Linux scheduler;
4. The isolation of the different network stacks of the emulated nodes can explore lightweight mechanisms such as Linux network namespaces for higher scalability;
5. A common *Emulated time* can be injected into the Linux kernel through a new system call. Interpolation of the emulated time on the kernel can avoid the need

of frequent simulated time updates, keeping the time update overhead low. A time dilation factor (similar to the approach in TimeKeeper [85] and VT-Mininet [89]) allows virtualizing the CPU capacity and keeping-up with the required extended time of simulation platforms;

6. Control Groups (specifically, freeze CGroups) control the computing power of the emulated nodes. Using a single system call, the framework can freeze/unfreeze all processes under execution in the network nodes. Improvement to the CPU CGroups are required to enforce a maximum equivalent CPU capacity even without CPU contention.
7. Dynamic adjustment of the time dilation factors given the actual load on the containers (similar to VT-Mininet [89]). Will reduce the maximum time extension overhead as possible.

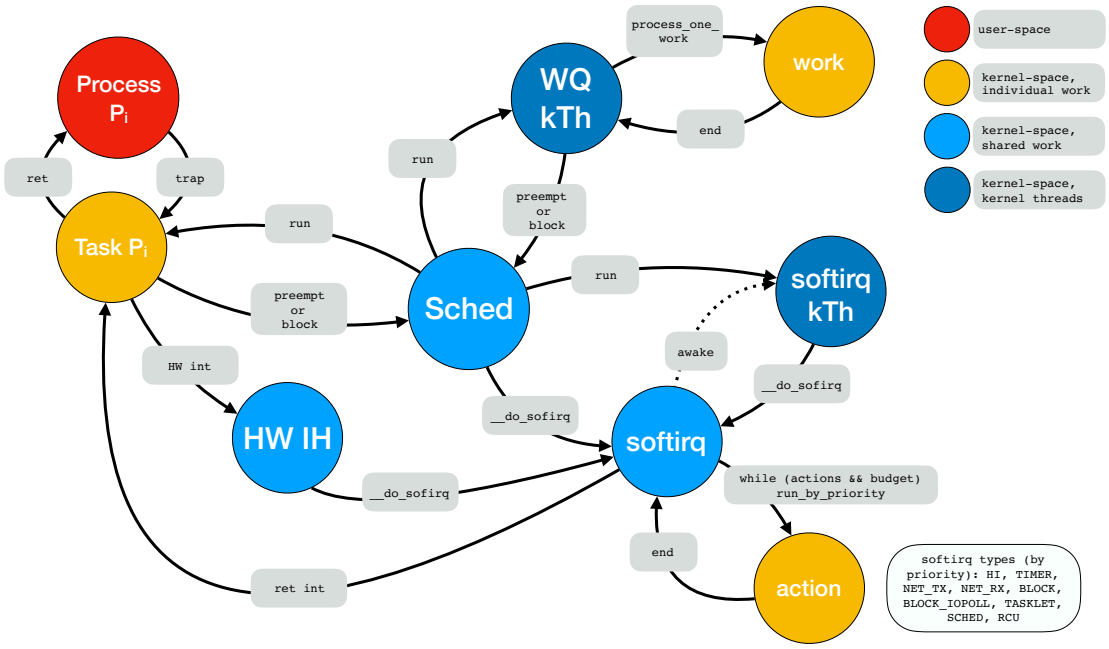


Figure 2.3: State machine for the Linux kernel execution, identifying the different forms of kernel tasks inside the Linux kernel and their entry/exit points. Linux kernel version 4.4.

Chapter 3 An experimentation platform for the evaluation of autonomic agents

This chapter introduces a tailored experimentation platform to support the decision-making of autonomic agents while realistically evaluating wireless mesh networks metrics (throughput capacity, latency, and others) under load through accurate simulation. We named it *PANE - Platform for Autonomic Networks Evaluation*. *PANE* is composed of independent modules to support essential requirements for autonomic agent evaluation, which operate on communication networks.

Our motivation for the design of this platform consists of fulfilling three main requirements. The first and more significant requirement was the ability to *fast prototype autonomic agent models that consumed wireless network structural information*. Differently from network simulators also relying on nanoseconds resolutions, the time scale of events in this agent environment is of sub-seconds. Moreover, network simulators are difficult to extend and modify, having strict software engineering design constraints (*e.g.*, ns-3 [73, 93]). We designed the agent simulation component of this platform from scratch (clean-slate approach), using an interpreted programming language (Python): *ASim*. *ASim* supports the integration of different network simulators as necessary. It relies on standardized formatting of messages for integration with network simulators.

The second requirement was the *support of dynamic behavior regarding all layers of the network topology definition*. The idea of agents that operate at the physical and link-layer levels requires allowing them to make decisions on the network topology during experiments dynamically. This characteristic is not common to network simulators that typically assume an a priori design of network topologies (network nodes and their connectivity) before the execution of experiments. The component *NetSim* integrates *ASim* to a well known network simulation toolset.

The final requirement is the *precise wireless network simulation* integrated with the autonomic agents' environment. We applied the well-known ns-3 network simulator [73, 93] to implement this requirement, integrating to it Linux Namespaces-based [94] containers for the implementation of nodes functions based on existing software.

Figures 3.1 and 3.2 present the architectural view of the experimentation platform and the relationship between the layers of the protocol stack and their operating environments. This platform has approximately 16800 lines of code between Python, C/C++, and Unix

shell script. Documentation relies heavily on *code comments*, representing an addition of 40% in lines (approximately 6800).

Following, we introduce the modules of this platform.

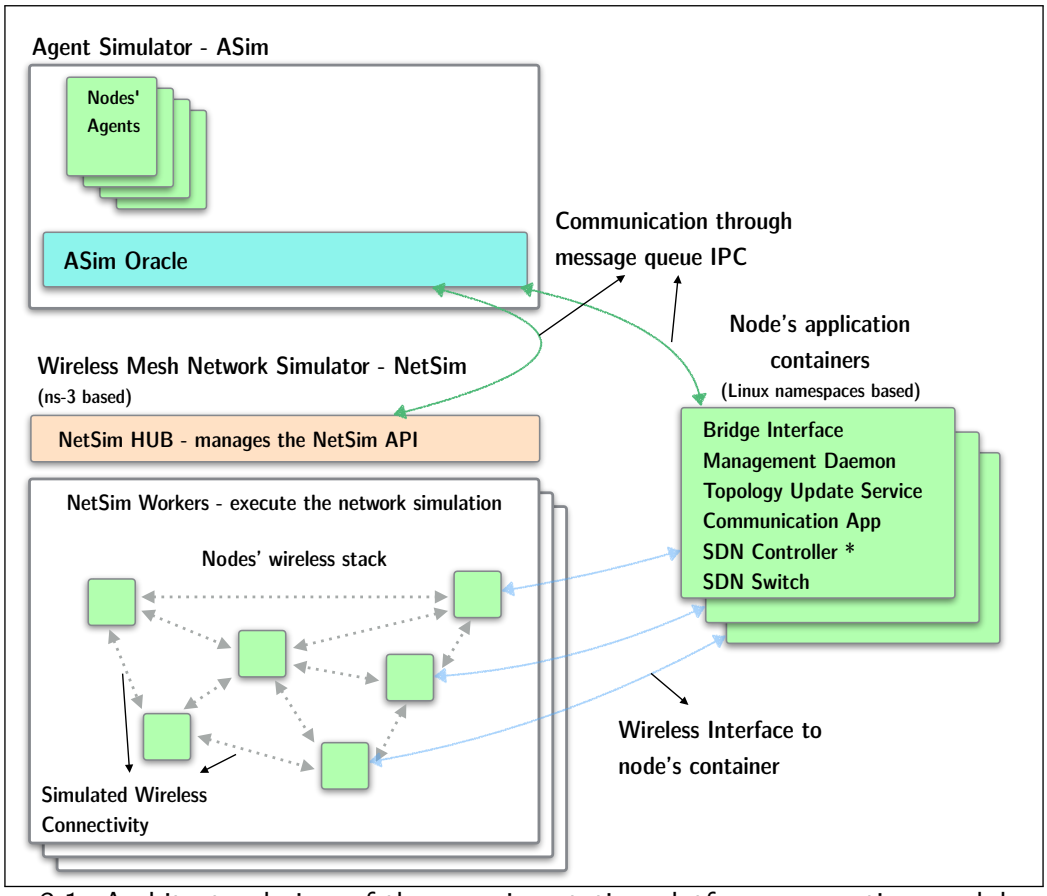


Figure 3.1: Architectural view of the experimentation platform, presenting modules, and their integration through message queues IPCs (inter-process communication). NetSim uses MPI [95] for multiplying its instances and scale up its processing capacity.

3.1 Agent Simulator - *ASim*

ASim is a discrete event simulator written in Python using multi-threading and multi-processing to allow for fast prototyping of agent models and independence of specific network simulators and node container environments while coping with large-scale experiments. It implements the high-level decision making of agents in the nodes: read the environment, decide based on autonomic properties, act: send commands to the *NetSim* component to enforce decisions. Also, it implements the general control of experiments: node placement creation, epochs, the collection of the resulting data, among other functions.

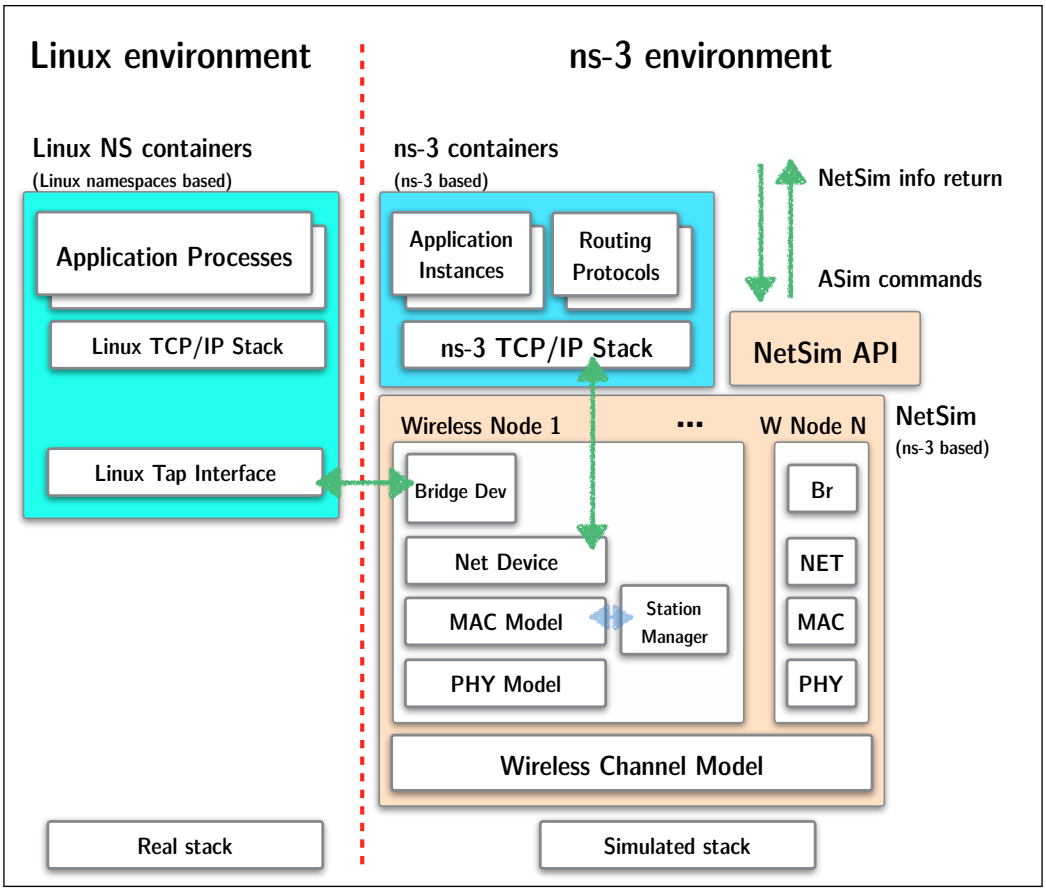


Figure 3.2: Protocol stacks and executing environments, showing the flexibility of the implementation of network nodes' functions: real software inside Linux Namespaces containers, internally simulated as ns-3 containers.

ASim has a sub module named *Interactive Console* (*IntCons*), which provides a graphical visualization of the topologies produced by the agents operating in *ASim*. The *IntCons* module relies on the Matplotlib [96, 97] Python library for graphical visualization and supports real-time and offline visualizations. In offline mode, *IntCons* can reproduce an atomic state of the WMN topology saved during a previous execution or can re-execute a sequence of commands performed by the agents to show the WMN formation process. *ASim* saves the sequence of commands sent to *IntCons* for its future standalone execution. Figures 4.2, 4.3, 4.4, 5.1, 5.2, 6.1, 6.2, 6.3 are examples of images produced by *IntCons*. Figure 3.3 shows an experiment using SDN switches and controllers with SDN paths displayed in the *IntCons*.

Both *ASim* and *IntCons* use geographical data such as the Microsoft US Buildings Data Set [98] and the Global Administrative Areas (GADM) [99] through the GeoPandas [100] Python library. MS US Buildings provides the location of buildings in all 50 states of the United States used by *ASim* to determine realistic locations for the simulated mesh

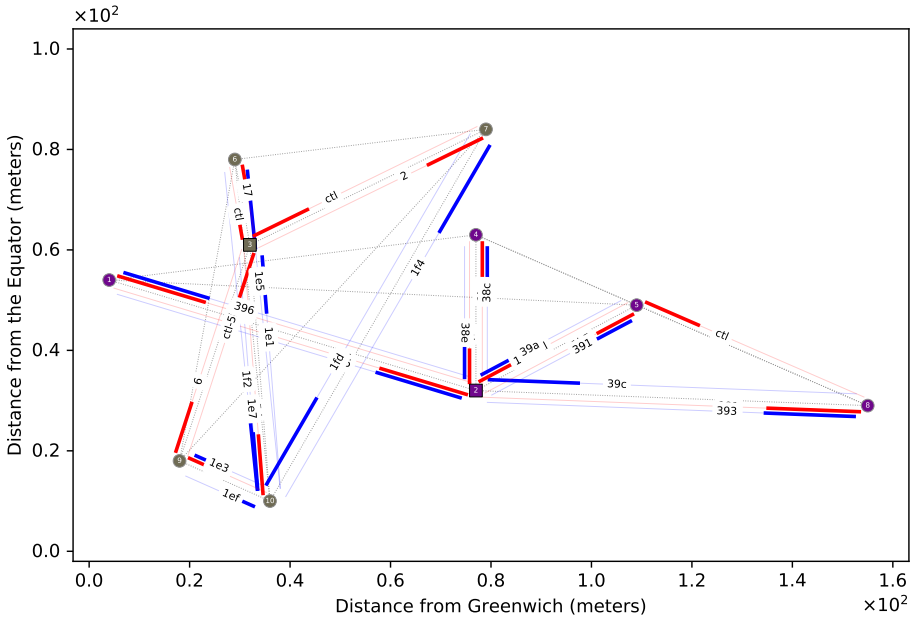


Figure 3.3: Interactive Console (IC) depicting a small topology of 10 nodes using Linux containers in SDN mode. Red lines represent the path of control channel of nodes. Blue lines represent data paths used by nodes.

nodes. GADM provides political boundaries, which in the US include Country, State, County, and City limits. Also, ASim can produce synthetic node placements based on random positions and controlling the node density over the experimentation area.

ASim and *IntCons* also rely on the NetworkX [101, 102] Python library for storing graph data representing the neighboring relationship between agents and for performing graph processing functions.

3.1.1 *ASim* parameters

The tables below present *ASim* parameters on their categories. When the type is "bool", the "Deafult" represents the action performed on the attribute.

3.2 Network Simulator - *NetSim*

NetSim is a layer built on top of the well known and validated ns-3 network simulator [73, 93] to command the configuration of network nodes and their network stack, which are implemented using ns-3 models. *NetSim* admits commands from the autonomic agents, translating them into topological behaviors by the ns-3 simulated nodes such as *new network node*, *node creates a mesh partition*, *node joins a partition*, *node*

ASim Generic Parameters				
Syn	Keyword	Description	Type	Default
-t	–max-time	Experimentation time	int	100
-m	–mpi-size	Number of MPI processes on NetSim. Minimum 2	int	2
	–sysvq-base	Initial SYS V queue index value	int	100
	–cs-m	Channel frequency attribution method: 1=Least Nodes Rank, 2=Least Used Nearby, 3=Random, 4=Round Robin, 5=Random attribution/active channel change	int	4
	–mail-to	The experiment's completion notification e-mail	str	"
	–epoch-len	The epoch duration in seconds and number of epoch splits in format '90,10'	str	'90,10'
	–channel-sets	Restricts the sub-bands of channels for a given bandwidth. Example: '20,1,2;40,1'	str	"

Table 3.1: ASim Generic Parameters.

ASim Display Consoles				
Syn	Keyword	Description	Type	Default
-e	–node-evol-graph	Node evolution plots: 0-> do not create, 1->save plots only, 2->save plots and display	int	0
-c	–top-cons	IntCons: 0-> do not create, 1->save plots only, 2->save plots and display	int	1
–ic-chm	–ic-ch-mode	Interactive Console convex-hull mode: simple, convex-hull, freq-reuse, all	str	None
	–ic-track-groups	IntCons will track the list of groups by their IDs (comma separated list)	str	None
	–ic-track-nodes'	IntCons will track the list of nodes by their IDs (comma separated list)	str	None
	–ic-cmds-save	IntCons save commands to file provided (no extension needed)	str	None
	–ic-no-toolbar	Disables toolbar in interactive mode of IntCons	bool	store_true
–ic-cbc	–ic-color-by-channel	IntCons will use the channel number for coloring groups	bool	store_true
–tc-um	–top-cons-under-mesh	IntCons draws the underlying WMN's max connectivity topology	bool	store_true

Table 3.2: *ASim* Display Consoles Parameters.

leaves a partition, *node changes its radio frequency*. *NetSim* is built in C++ using the programming guidelines of ns-3.

NetSim uses MPI [95] for the parallel execution of its instances, increasing experimen-

ASim Node Placement Parameters				
Syn	Keyword	Description	Type	Default
-s	–seed	Node placement randomization seed	int	1230
	–npr-goal	Goal for the # of nodes per NetSim MPI rank. Limited by the nodes' density (hence goal)	int	60
–n-max	–nodes-max-goal	Goal for the total # of nodes. Limited by the nodes' density (hence goal), disables '–npr-goal'	int	0
–n-dens	–nodes-density	Node placement average density inverse in m^2/node eg. 1200, 800. Disables '–npr-goal', '–nodes-max-goal'	int	0
–sp-sizes	–search-part-sizes	Search partition of specific sizes in the current seed and node density. Different seeds should be tried externally (external list provided here one by one)	str	None
	–spawn-mode	Node spawning mode: 'synthetic', 'geo', 'geoM', 'replay'	str	synthetic
	–spawn-data	Location/name of node spawning data for 'geo' and 'replay' modes	str	None

Table 3.3: *ASim* Node Placement Parameters.

tation scalability. Two main types of functions exist on the MPI processes of *NetSim*: the *NetSim-Hub* and *NetSim-Workers*.

The *NetSim-Hub* is responsible for time synchronization between *NetSim-Hub*, *NetSim-Workers*, and *ASim*. Also, it receives commands from *ASim* and either execute them directly or transfer them to *NetSim-Workers*. *NetSim-Hub* also receives informational and status messages from *NetSim-Workers* and forward them to *ASim*.

NetSim-Workers execute the network simulation workload according to the topologies defined by topology manipulation and network communication commands sent by the autonomic agents on *ASim*. *NetSim-Workers* also create and control the lifecycle of nodes' containers, and perform bi-directional communication to containers.

3.3 Nodes' Containers

Nodes' containers implement a computing environment for the execution of nodes' applications such as data generation and consumption, the real TCP/IP network stack, mesh routing protocol, SDN controllers and SDN software switches when applying the SDN paradigm. The platform currently supports ns-3-based containers (applications, IP stack, routing protocol are internal models to ns-3) and Linux Namespaces-based

ASim Agent Models Parameters				
Syn	Keyword	Description	Type	Default
-i	–node-ai-model	Node agent model: 'standard', 'smart'	choice	smart
	–max-degree	Max node degree in a group. Used by the 'smart' agent	int	-1
	–sh-max-degree	Max node degree when bridging to a group. Used by the self-haling agent. Default: 0, use the same value as the smart agent	int	0
–lvth	–liveness-threshold	Format: '10,20,10'. Threshold in % used by a so-agent, so-agent-leader, sh-agent to determine a better option than the agent's current one.	str	10,20,10
–svo	–safety-violation-opt	Format: 'rnd,brd,sft' or 'no'. Active options for safety violation solution of the smart agent. Any option can be provided independently or with others. 'no' implies no option active	str	rnd
–sipct	–sec-intf-pct	Percentage of nodes with a second interface: 10, 20, 25 ... 100	int	0
–sipol	–sec-intf-policy	The type of Self-Healing agent using the secondary interface (ss, gp)	str	ss
–no-atomic	–no-atomic-env-read	Disables reading the environment atomically	bool	store_true

Table 3.4: *ASim* Agent Models Parameters.

containers (existing software runs unmodified inside Linux Namespaces consuming the Linux TCP/IP stack and routing protocols). We use the *clone()* Linux system call to create namespaces that isolate processes, networking resources, hostname. We leave the IPC mechanisms shared to implement an inter-module API, as described in Section 3.4.

The current implementation of Linux containers is restricted to small scales (a few 10ths of nodes) due to the limitation on the synchronization between simulated time inside ns-3 and the real-time perceived by application and kernel functions on Linux containers.

The alternative of implementing applications inside *NetSim* allows experimentation at larger scales without incurring in synchronization issues. However, we cannot reuse existing software such as SDN controllers and switches.

ASim Nodes containers attributes				
Syn	Keyword	Description	Type	Default
	–container-type	Type of NetSim containers for nodes computing resources: 0=ns-3, 1=LinuxNS	int	1
	–no-sdn-mode	Disable SDN mode on nodes' containers	bool	store_true
	–qos-queues	Queues based QoS on nodes' SDN switches	bool	store_true
	–ovs	Use OpenVSwitch as the SDN switch in nodes' containers	bool	store_true
	–fwd	SDN ctrl forwarding algorithm: HC=0, HLRB=1, HLRB-SHC=2, CA=3	int	0
	–phy-std	PHY Standard for wireless interfaces: 11a (default), 11afixed6, 11n5GHz, 11ac	str	11a
	–phy-bw	PHY bandwidth for wireless interfaces (MHz)	int	20
	–force-non-promisc	Force non-promisc wireless operation regardless of the SDN mode	bool	store_true

Table 3.5: *ASim* Types of nodes containers and associated attributes.

ASim Debugging Modes				
Syn	Keyword	Description	Type	Default
	–shortcut	Shortcuts NetSim, making ASim operate isolated. ASim sends response messages to simulate NetSim's behavior	bool	store_true
-d	–debug	Use ns-3 debug libraries	bool	store_true
	–use-gdb	Run NetSim under GDB'	bool	store_true

Table 3.6: *ASim* Debugging Modes.

ASim type of nodes' applications				
Syn	Keyword	Description	Type	Default
	–app-type	app type on ns-3 containers: UDP=0, TCP=1	int	0
	–app-udp-no-rate-control'	If using ns-3 containers app type UDP, disable Rate Control (use CBR traffic)	bool	store_true'
	–comm-dist	Maximum distance to dst nodes on communication	int	1000
–app-off	–app-disable	Disable any node communication	bool	store_true

Table 3.7: *ASim* type of nodes' applications.

3.4 Inter-modules messaging API

For simplicity, reliability, and general applicability, the messaging API uses string encoding. This design decision simplified the data exchange between platforms developed in different languages such as C, C++, and Python. The Inter-Process Communication (IPC) mechanism used was System V Message Queues [103].

3.5 Time synchronization

The modules that compose this platform operate in tandem regarding their time evolution. However, as the relevant time scales on each module differ, we implement different tolerances for time synchronization. *ASim* and the *NetSim-Hub* use nano-seconds precision for time comparison and perform synchronization every 0.1 seconds. In other words, after executing independently for 0.1 seconds, both *ASim* and the *NetSim-Hub* wait for each other to confirm that they are ready to move to the next 0.1 seconds epoch. The *NetSim-Hub* and *NetSim-Workers* use the same precision for time comparison, but they synchronization epoch duration is 0.001 seconds.

Multiple *NetSim-Workers* should only be used when experiments do not allow communication between nodes at different frequencies. This setting was useful for the capacity scaling experiments described in Chapter 4 when nodes communicated only internally to their partitions.

In Linux Namespaces-based Nodes' Containers, currently, there is no external time reference. Any applications and the Linux Kernel consume real-time. When using these containers, the *NetSim* modules operate with the ns-3 real-time scheduler, which limits the simulated time advancement to the system's real-time.

Chapter 4 WMN capacity scaling under autonomic topology manipulation

In this chapter, we experimentally evaluate the impact of the controlled formation of WMN topologies by autonomic agents with previous research, which assumed random topologies. Also, we contrast our experimental results with analytical bounds on the throughput capacity of WMNs. We use this study to select autonomic agent models that better fit our objectives. A common characteristic expected from all our agents is the ability to split large WMN topologies into partitions.

We start describing a generic operational cycle followed by the agents. In this chapter, the agents' single function is to control the connectivity of their assigned mesh nodes' wireless interface. We follow presenting the design and behavior of the different agents we envisioned, presenting the different topological outcomes advent from the behavior of our agents, and, presenting the evaluation of the WMN capacity scaling obtained by the agents' induced topologies.

4.1 Operational cycle of autonomic agents

The general design of the agents involves cycles of operation with the following four phases:

1. Read the environment: internal states, neighbors' states.
2. Evaluate states: on autonomic functions, deciding to maintain, improve, recover safety and liveness properties.
3. Act: send commands that enforce decisions.
4. Idle: wait until the next cycle (nodes use the network).

The second phase is the main differentiating element (properties and behavior), which is specific to each agent as described in the subsection 4.2.2. The commands available to use in the third phase are: *node creates a mesh partition*, *node joins a partition*, *node leaves a partition*. Also, a possible decision is *maintain membership*, implying a null action.

In the results of this chapter, our agents operate in atomic information gathering mode: when obtaining their local information, they read a consistent state of the network. This mode is equivalent to reading information of the multiple available frequencies in parallel, and in an infinitesimal time. Although not a realistic expectation, we apply this operation

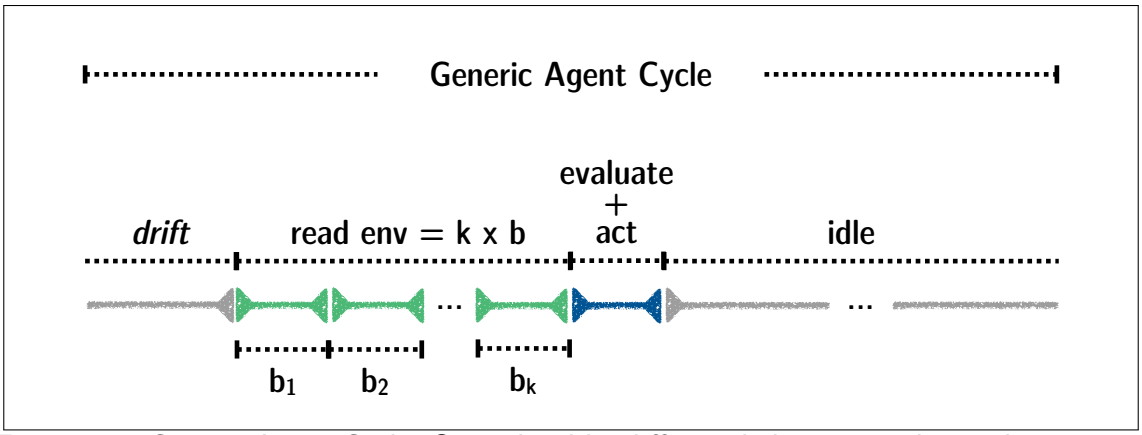


Figure 4.1: Generic Agent Cycle. Specialized by different behaviors in the *evaluate + act* phase.

mode to isolate the study of the capacity scaling from the study of the convergence likelihood of our agent models. All our agent models obtain fast convergence, in up to two epochs. They start from all mesh nodes inactive to a final state in which all nodes do not have a better outcome than their current connectivity decision (a setting equivalent to a Nash Equilibrium).

In Chapters 5 and 6, we evaluate the convergence likelihood of our selected agent models, enforcing their operation in concurrent mode and their frequency spectrum scanning in their *Read Environment* phase a single frequency at a time.

4.2 Autonomic behavior of agents

4.2.1 Manual node agent design

The *Manual* agent design joins the closest partition using signal strength as a proxy to proximity, which represents its liveness property. It holds as safety property an upper bound on the size of a partition, limited to 80 nodes. It does not change its partition membership after joining a partition. No review of properties occurs since the generic agent cycle runs only once for *Manual* when it starts its operation. These simple and static decisions resemble a manual configuration that is not reviewed to adapt to changed conditions.

Manual determines the partition proximity through the perceived signal strength from any single reachable node in the partition. *Manual* does not average the signal of reachable nodes in a partition to reflect its proximity. Moreover, the partition size upper enforced by *Manual* is critical to induce a partitioned topology outcome. However, this upper bound is only applied when deciding which partition to join, not after joined a partition. In

atomic experimentation settings, *Manual* properly bounds the sizes of partitions, inducing balanced partitioning settings. The same cannot be guaranteed for concurrent operation.

More formally: let m be a *Manual* agent analysing its properties. Let P be the set of all partitions while $P_m \subseteq P$ is the set of all nearby partitions to m; let $k \in P_m$ be a nearby partition. Let $F_{sig\_s}(k, m)$ be a function that returns the highest signal strength of any reachable node $i, \forall i \in k$ as perceived by the node m, let $F_{SS} \rightarrow P_m$ be a function that sorts P_m by $F_{sig\_s}(k, m), \forall k \in P_m$ into the ordered list $P_m^{'}$. If $P_m^{'} \neq \varnothing$, the first item $P_m^{'}[0] = k_b$ is the best partition membership option regarding proximity.

Let $F_{size}(k)$ be a function that returns the size in the number of nodes of a partition k. Let $F_{MPS} \rightarrow P_m$ be a function that *i)* eliminates partitions $k \in P_m$ for which $F_{size}(k) > 80$.

On its second phase of the generic agent cycle, *Manual* executes the functions $F_{MPS} \rightarrow P_m$ to eliminate partitions above limit and $P_m^{'} = F_{SS} \rightarrow P_m$ to determine the closest nearby partition.

Let A_M be the decision of *Manual* as a set of commands:

$$A_M = \begin{cases} \text{If } P_m^{'} = \varnothing & : \{create\_partition\} \\ \text{Otherwise} & : \{join\_partition(k_b)\} \end{cases} \tag{4.1}$$

4.2.2 Smart node agent design

The *Smart* agent design type holds *partition diameter* and *maximum node degree* as safety properties, and a membership to the *largest nearby partition* as a liveness property. Following we provide a textual description followed by a formal definition.

The *Smart* agent joins the largest nearby partition for which it finds the shortest path of hop-distance at most h to the partition origin. It is trivial to verify that the diameter of the partition is, at most, $d = 2 \times h$: the origin node is a member of any shortest path from border to the origin, and the diameter unites two shortest paths from origin to the border.

The *is origin* attribute is true for an instance of *Smart* which creates a partition, and false otherwise.

Optionally, *Smart* nodes assume a node degree dg upper bound constraint to decide on their partition membership. If no membership option is valid, *Smart* nodes create a new partition. They continually review their membership decision following the generic agent cycle described in Section 4.1.

Smart attempts to change its partition membership in every new cycle to the largest *valid* neighboring partition which is, at least, sp percent larger (a threshold) than the

Smart's current partition.

A *valid* neighboring partition has *i)* at least one node at communication reach of the deciding node (hence, neighboring), *ii)* all nodes not violating safety properties, *iii)* all nodes will not violate safety properties after the addition of the deciding node.

More formally, let m be a Smart agent node reviewing its properties. Let P be the set of all partitions while $P_m \subseteq P$ is the set of all nearby partitions to m; let $k \in P_m$ be a nearby partition, o_k is the origin node of partition k. Let k_c be the current partition of m if is already connected, and $k_c \in P_m$.

Let $F_{distance\_sp}(i, j)$ be a function that returns the shortest-path distance between two nodes. Let $F_{DT} \rightarrow P_m$ be a function that eliminates nearby partitions k for which $F_{distance\_sp}(m, o_k) > h, \forall k \in P_m$.

Let $F_{degree}(i)$ be a function that returns the degree of a node i; let $F_{DG} \rightarrow P_m$ be a function that removes partitions $k \in P_m$ if there exists any node $i \in k$ such that $F_{degree}(i) \geq dg, \forall i \in k, \forall k \in P_m$.

Let $F_{neighbors}(k, i)$ be a function that returns the number of future neighbors of a node i if it joins a neighboring partition $k \in P_m$; let $F_{NEIGH} \rightarrow P_m$ be a function that removes partitions $k \in P_m$ if $F_{neighbors}(k, m) > dg, \forall k \in P_m$.

More formally, let $F_{size}(k)$ be a function that returns the size in number of nodes of a partition k, let s_m be the size of the current partition k_c of the agent m, let $F_{ES} \rightarrow P_m$ be a function that *i)* eliminates partitions $k \in P_m$ for which $F_{size}(k) < (1 + sp) \times s_m$, and *ii)* sorts P_m by $F_{size}(k), \forall k \in P_m$ into the ordered list $P_m^{'}$.

If $P_m^{'} \neq \emptyset$, the first item $P_m^{'}[0] = k_b$ is the best partition membership option regarding size. If $P_m^{'} = \emptyset$ and $k_c \in P_m$ (current partition k_c checked as *valid* regarding safety properties), we add back k_c into $P_m^{'}$ as an option (stay on current partition).

On its second phase of the generic agent cycle, *Smart* executes: $F_{DT} \rightarrow P_m$, $F_{DG} \rightarrow P_m$, $F_{NEIGH} \rightarrow P_m$, and finally $P_m^{'} = F_{ES} \rightarrow P_m$.

Let A_S be the decision of *Smart* as a set of actions:

$$A_S = \begin{cases} \text{If } P_m^{'} = \emptyset & : \{create\_partition\} \\ \text{Else if } P_m^{'} = \{k_c\} & : \emptyset \quad \text{(a null action set)} \\ \text{Otherwise} & : \{leave\_partition(k_c), \\ & \quad join\_partition(k_b)\} \end{cases} \tag{4.2}$$

We evaluated the capacity scaling of versions of *Smart* holding different values for the safety property *maximum node degree* from 4 to 10, and no degree constraint.

4.3 Visual outcome of the behavior of agents in atomic settings

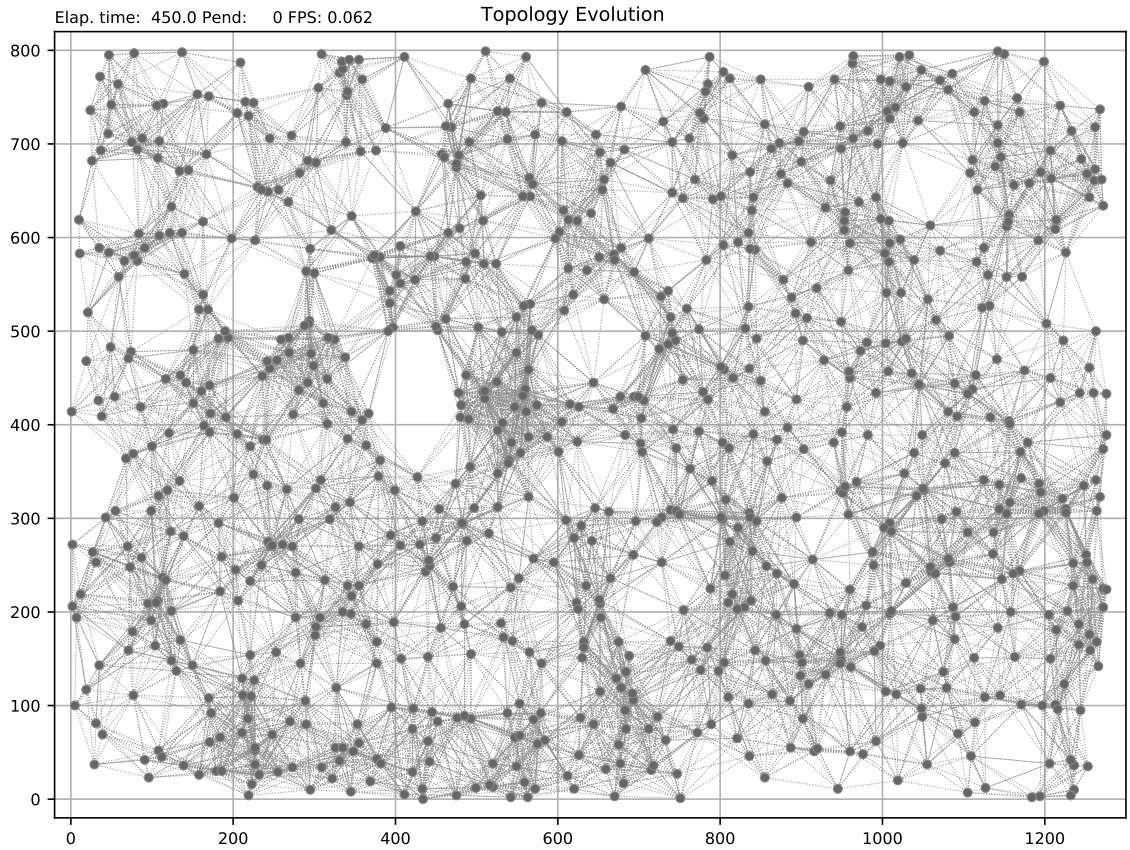


Figure 4.2: Underlying maximum possible connectivity of the random node placement from seed 5 if a single partition exists. Shows the maximum set of neighboring options of any node when deciding partition membership. Area of $\approx 1\ Km^2$ (1280 by 800 meters) using the IEEE 802.11a wireless standard.

Figure 4.2 represents the maximum *underlying* connectivity if all nodes operate on the same frequency in the physical layer, the same logical network (SSID, BSSID) at the link-layer, using their nominal transmit power. Therefore, no additional connectivity can exist.

More formally: let $G(V, E, P)$ be a graph representing the connectivity topology of a WMN where V is the set of nodes, E is the set of edges, P the set of positions of the nodes in V, n is a node such that $n \in V$, f_n, t_n are frequency, network id of n; E is maximal if $\forall i, j \in V, f_i = f_j, t_i = t_j$.

The choices of the self-organizing agent on the WMN nodes define the partitioned WMN topology (Figures 4.3, 4.4). The same node placement (position of nodes) is the

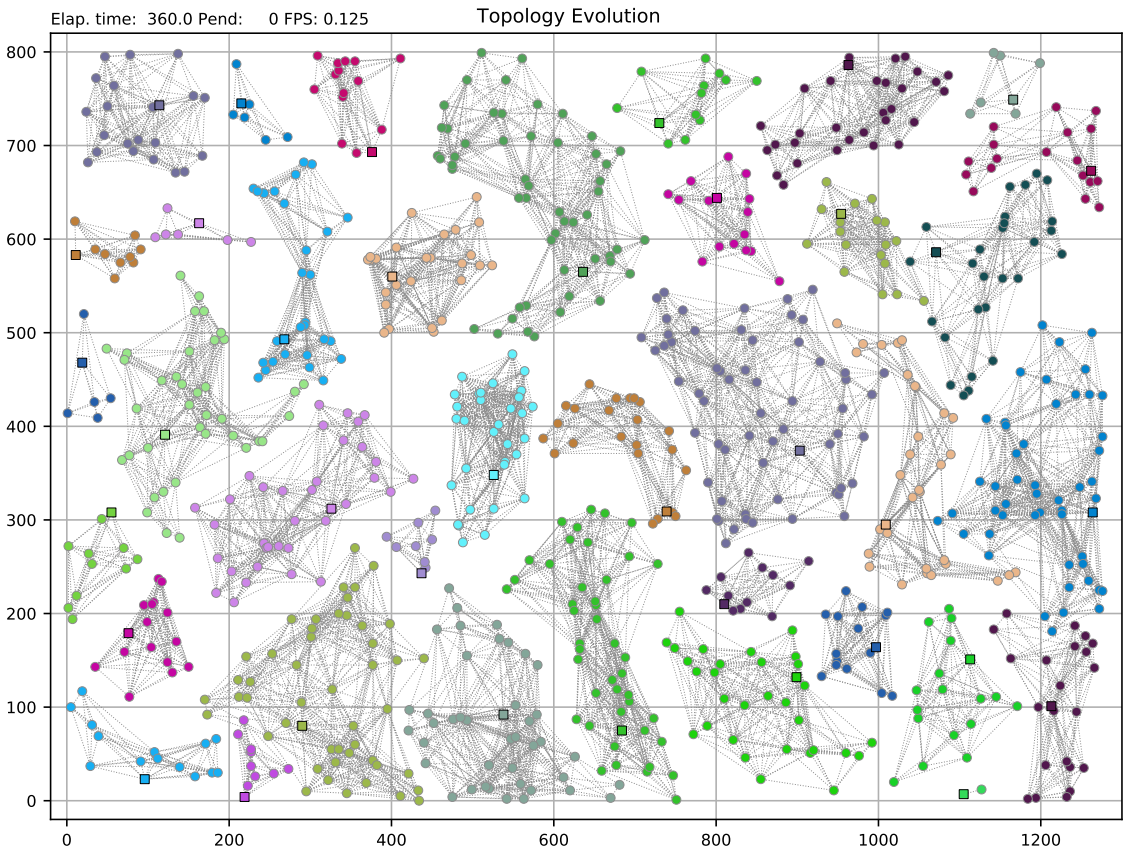


Figure 4.3: Evolved WMN topology partitioned by the *Manual* self-organizing agent. Same area, wireless standard and node placement of Figure 4.2. A subset of the maximum set of edges exist.

basis for the three figures. Partitions in Figure 4.3 are not constrained in their diameter. Partitions in Figure 4.4 are diameter constrained. More formally: let $G_{4.2}(V_{4.2}, E_{4.2}, P_{4.2})$, $G_{4.3}(V_{4.3}, E_{4.3}, P_{4.3})$, $G_{4.4}(V_{4.4}, E_{4.4}, P_{4.4})$ be graphs representing the topologies of Figures 4.2, 4.3, 4.4, respectively; $P_{4.2} = P_{4.3} = P_{4.4}$, $E_{4.2} \supseteq E_{4.3}$, $E_{4.2} \supseteq E_{4.4}$.

4.4 Scaling results

This section presents the experimental settings used in our evaluation and results for the capacity scaling of the partitions created by our agent models.

4.4.1 Experimentation settings for scaling results

Here we provide the experimental settings used to produce the results presented in Figures 4.5 - 4.8. The following list describes experiments and how the node placements (positioning of nodes) were produced.

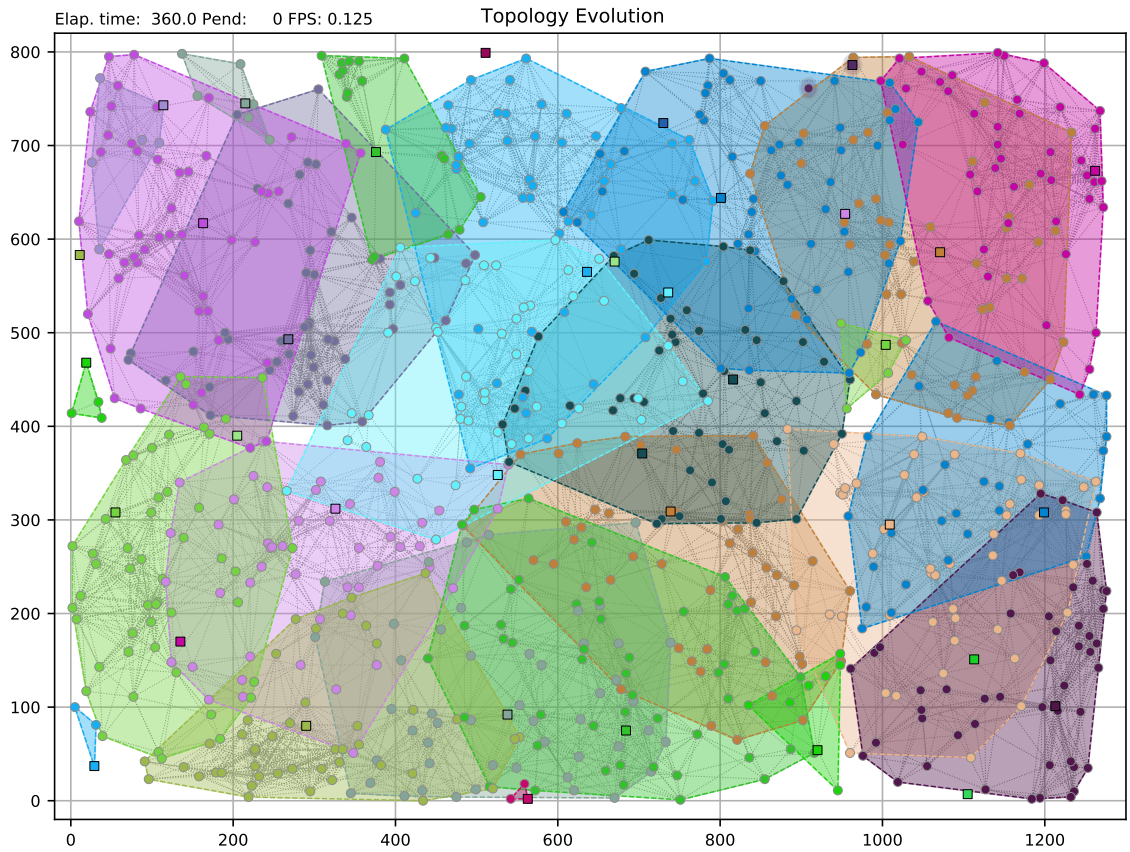


Figure 4.4: Evolved WMN topology partitioned by the *Smart* self-organizing agent, producing overlapping partitions, made evident by the partitions' convex-hull. Same area, wireless standard and node placement of Figure 4.2.

1. Nodes have randomized positions controlled by a *node placement random variable* (uniformly random) and its seed value. We used a common set of 15 *seeds* to all agent models, creating 15 different node placements (each one represents a different experimental evaluation). Also, we added to the common seeds set of each agent model-specific seeds to guarantee that any scaling data point (a given partition size on scaling plots) derives from, at least, 10 samples. A capacity sample derives from the partition operation during a given epoch. Moreover, we also enforce that those samples came from at least 3 distinct partitions.
2. We control the average node placement density over the experimentation area. Our initial analysis uses the density of 1 node per $1260\ m^2$. We also evaluate the agents' robustness to different densities: 1 node per $400, 800, 1600\ m^2$.
3. Each experiment consists of a set of 7 epochs of 90 simulated seconds each. Therefore, an experiment resembles the formation of WMN partitions: initially, all nodes are disconnected; at the end, all nodes as members of partitions and actively com-

municate. All experiments converge to stable topologies given the use of atomic operation as described in Section 4.1.

The following list describes the communication traffic for capacity determination, followed by a formal definition.

4. The communication traffic between nodes assumes unlimited demand. A pair source-destination within a partition defines a flow. Each flow consists of multiple messages of size 20 KB (kilobytes), limited by the flow duration of 3 sec. We store the number of bytes received by the destination nodes, accumulating those per partition on each epoch to derive the data points of partition's throughput capacity (received bytes over the total communication time on the epoch - a communication rate).
5. Source nodes choose destinations at random controlled by a *partition communication random variable*. The destination is a member of the same partition as the source node. Many intermediary (forwarding) nodes might be involved in a given flow.
6. We used a unidirectional data flow TCP app: source sends data to a destination. The TCP protocol will attempt to consume all available bandwidth. The ns-3 TCP congestion control model used was *TCPNewReno* [104].
7. The communication volume and rate are free of routing protocol overhead (end-to-end). We used the ns-3's implementation of the routing protocol OLSR [47] and its standard hop-count routing metric.

Following we provide a formal description for the communication traffic.

T_T is the *total* communication rate in a mesh network (one partition). T_N is the *net* rate: the end-to-end communication rate free of routing and forwarding overhead. T_N is the metric we used on the determination of a mesh network throughput capacity. The forwarding overhead O_F and routing overhead O_R are extra traffic carried on the network necessary to implement the end-to-end traffic.

Let R be a routing protocol which sends control messages of average length r bytes, $|R|$ be the total number of messages in a given interval t, and A be the application used for peer-to-peer communication between a source node n_s and a destination node n_d. As $\{n_s, n_d\}$ are not necessarily neighbors, a multi-hop communication on path $P = \{n_s, f_1, ..., f_{|P|-2}, n_d\}$ is implemented by the forwarding routers f, comprising the set $F = [P] \setminus \{n_c, n_s\}$. Let $L = \{1, ..., |L|\}$ be the set of all communication flows of the application A over a given time interval t. Let $f_{i,u}$ represent the forwarding node u in the path of a flow $i \in L$. Finally, the function $\omega()$ returns the data received by a node.

We now define T_T based on the application traffic transported by all nodes T_A and

the routing overhead O_R:

$$
\begin{aligned}
T_A &= \sum_{i=1}^{|L|} \left[\omega(n_{s_i}) + \sum_{u=1}^{|F|} \omega(f_{i,u}) + \omega(n_{d_i}) \right] \\
O_R &= |R| \times r \\
T_T &= \frac{T_A + O_R}{t}
\end{aligned}
$$

We define the *net* traffic T_N as the source-destination data reception of flows. Given the use of a unidirectional app, traffic reception concentrates on the destination nodes:

$$\omega(n_{s_i}) \ll \omega(n_{d_i})$$

Therefore:

$$T_N = \frac{\sum_{i=1}^{|L|} [\, \omega(n_{s_i}) + \omega(n_{d_i}) \,]}{t} \approx \frac{\sum_{i=1}^{|L|} [\, \omega(n_{d_i}) \,]}{t}$$

It follows that the forwarding overhead O_F is:

$$O_F = \sum_{i=1}^{|L|} \sum_{u=1}^{|F|} \omega(f_{i,u})$$

The following list describes settings of the wireless subsystem.

8. Nodes use the IEEE 802.11a wireless standard, which supports transmission rates from 6 to 54 Mbps. We use the well known ns-3 simulator [73] to model the wireless stack of mesh nodes. In these experiments, we also implemented the communication applications inside ns-3. Therefore, experiments do not suffer from poor time synchronization due to the simultaneous use of simulated time wireless stacks and real-time applications.
9. The mesh nodes' IEEE 802.11a physical layer operates on the 5 GHz band with a 20 MHz bandwidth. The transmission power is 16 dBm (default). The gain of the transmission and reception antennas is 1 dBi (also defaults). The CCA (Clear Channel Assessment[1]) threshold is -99 dBm. The Energy Detection Threshold[2] is -96 dBm. The last two also default values.

[1] Identifies the channel as free for transmission.

[2] Triggers the start of packet reception.

10. The link-layer of the mesh nodes operates in the IBSS (Independent Basic Service Set) mode, supporting the creation of wireless mesh networks through the multi-point association of nodes at the link-layer.

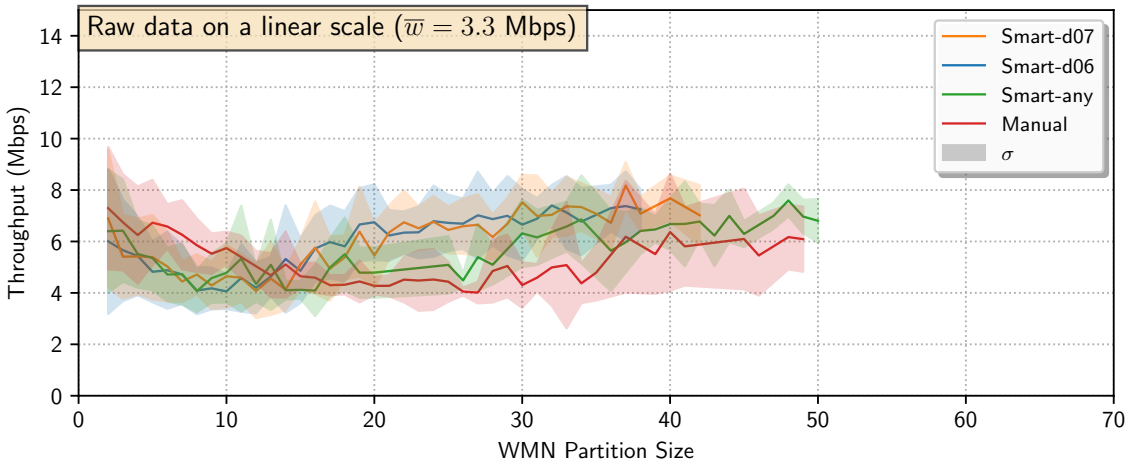


Figure 4.5: Capacity scaling of the *Manual* and *Smart* agents. Density $1/1260\ nodes/m^2$. Three variations of *Smart*: node degrees 6, 7, and no node degree constraint (*any*). Transparent regions represent the standard deviation. Agents enforcing degree control outperformed the other agents.

4.4.2 Capacity scaling results

This sub-section presents the experimental capacity scaling results achieved by the baseline agent (*Manual*) and the *Smart* agent. For *Smart*, we provide individual results for each node degree evaluated: $dg = \{any, 4, ..., 10\}$.

Figure 4.5 presents the graph of the capacity in $Mbps$ achieved by partitions of a given size in their number of nodes. For any size, the graph presents the mean capacity (lines) and the standard deviation (transparent areas). The curves represent scaling results of the *Manual*, *Smart* (no node degree constraint - *any*), *Smart* with maximum node degrees 6 and 7.

Our *first* observation is that the capacity scaling up to the size ten is negative, which is consistent with previous reporting [105]. However, extending the observation of the scaling to larger network sizes than previously reported shows that the negative trend changes to positive. With larger WMN partitions, an increase in spatial diversity takes place, allowing different data flows to co-exist when they are spatially distant, not competing in the same collision domain.

The *second* observation about Figure 4.5 concerns the *region of minimum capacity: RMC*. The *RMC* for the node degree constrained curves occurs early at sizes $\approx 9 - 11$.

Also, the width of *RMC* is of $2-3$. The *Manual* agent's curve has *RMC* centered at size ≈ 18 with width of $\approx 25-30$, representing a much larger range of sizes with reduced capacity.

Finally, we explain the format of the capacity scaling curves. The curve starts from a single collision domain (SCD) at size 2. At this point, all nodes can sense the transmission of all other, and they share the capacity of the SCD fairly. With the increase in the size of partitions, phenomena such as *the hidden terminal*, *the exposed terminal*, *flow in the middle* occur [106, 107], causing packet corruption and/or excessive contention which decrease the capacity of the partitions. At the end of the minimum capacity range, the increasing spatial area of the partitions allows for increased concurrency of flows, yielding a recovery to the throughput capacity.

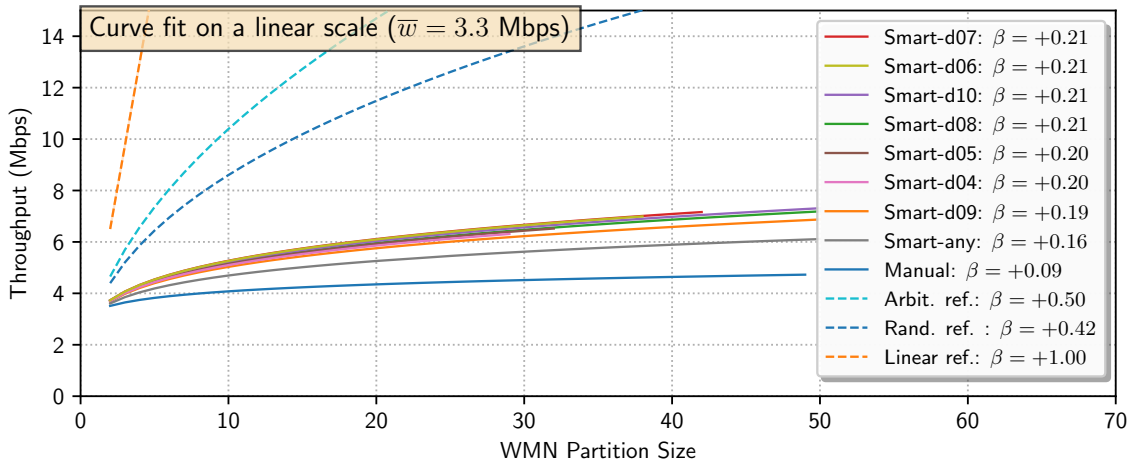


Figure 4.6: Curve fit of the raw capacity scaling curves to the scaling model of Equation 4.3. Density $1/1260\ nodes/m^2$. The legend presents the β scaling factor for the respective agent configurations. Agents enforcing degree control outperform the other agents.

Figure 4.6 presents graphs of curves derived from fitting the raw capacity scaling data to a generic scaling model (Equation 4.3). More formally, let $C(n)$ be the throughput capacity as a function of the WMN partition size n in the number of nodes, w a constant representing the mesh nodes' wireless interface link throughput, and β the scaling factor.

$$C(n) = w \cdot n^{\beta} \tag{4.3}$$

We apply the algorithm *curve_fit* which performs a non-linear least squares method to fit a non-linear function to data [108]. The generic scaling model of Equation 4.3 is applied in other scaling studies such as [6, 7, 15, 16].

Figure 4.6 shows that all node degree constrained versions of *Smart* presented a better

scaling factor β than the two non-constrained models *Smart-any* and *Manual*. Figure 4.6 also provides a perspective of the arbitrary and random asymptotic references from [23] (dashed lines) under the w coefficient and size range of this study, allowing an *external* comparison to those analytical studies.

We refrain from drawing any *internal* comparisons between the degree constrained agent models given that the generic scaling model (Equation 4.3) is not a good representation of our data.

$$C(n) = a \cdot n^4 + b \cdot n^3 + c \cdot n^2 + d \cdot n + e \tag{4.4}$$

We return to the concept of the *RMC* to internally compare the scaling curves, relying on a fitting model based on a polynomial function of order four (Equation 4.4). Figure 4.7 presents the fitted curves using the polynomial model, showing a closer representation of the capacity scaling data. Assume the width of the *RMC* as the difference of the two first real roots r_1, r_2 of the fitted polynomials (fitted curves of Figure 4.7 are offset by -5.0 Mbps to produce real roots). Also, assume the scaling factor $\gamma = 100/(r_2 - r_1)$, which turns larger values into better scaling results (similarly to the metric observed on the exponential model).

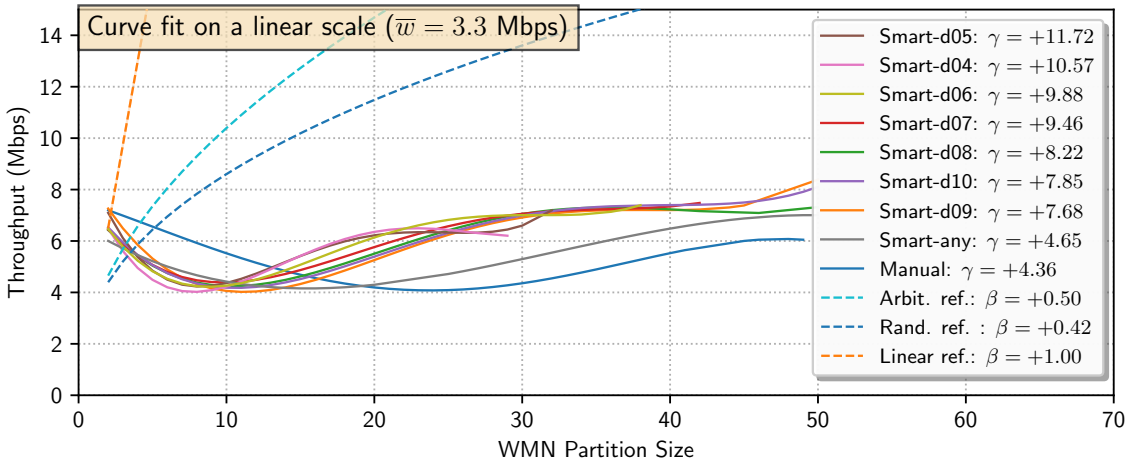


Figure 4.7: Capacity scaling curves fitted by a polynomial function of order four. Agents enforcing degree control have reduces RMC. Density $1/1260\ nodes/m^2$.

The *Smart* models with maximum node degree around 5 produced the best γ. Extending this analysis to other node placement densities (Figure 4.8), we find that the models with degree constraints closer to the range $5 - 6$ more frequently produced the best γ. This result is consistent with previous studies on the ideal node degree in a multi-hop network [26], [27]. Additionally, the non-constrained version of *Smart* and the *Manual* agent (for which a node degree constraint is not an option) present the weakest γ, supporting

the application of the degree control in the self-organizing models we proposed.

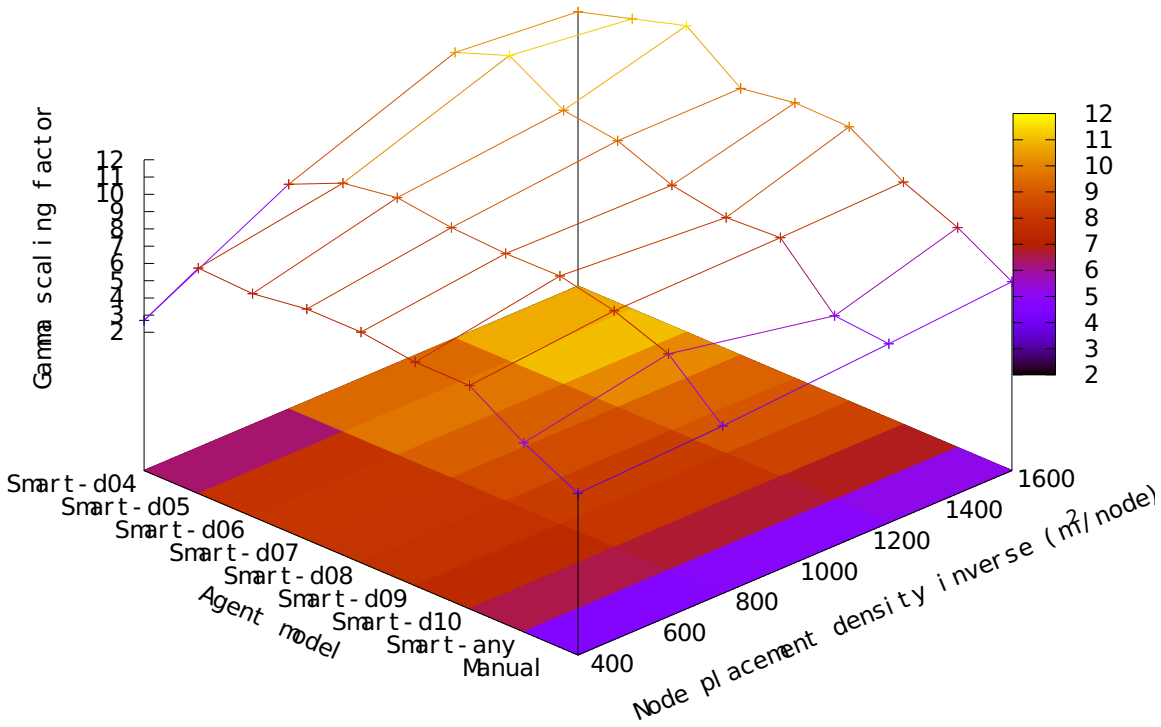


Figure 4.8: The robustness of the agents enforcing maximum degree control under different node placement densities. Scaling model: fourth order polynomial (Equation 4.4).

Chapter 5 Self-Organizing WMN nodes

This chapter describes the idea, design, and evaluation of self-organizing agents for WMNs. Previous research explored existing topological properties of the network to increase WMN throughput capacity [20]. Our goal is to explicitly enforce a set of invariants to the WMN topology by its dynamic formation at the link and physical layers using autonomic agents. On this agent design, we expect no preliminary network planning or manual configuration, characterizing a self-configuring network solution [12]. Under the addition and removal of nodes, our agent-based network formation solution must guarantee a set of properties critical to the applicability of the SDN paradigm into WMNs: a self-organizing design [12].

A critical question in dynamic and autonomic network formation is the convergence to stable configurations and how the parameters of the self-configuring, self-organizing agents affect such convergence likelihood. To evaluate the agents' ability to converge, they operate in concurrency mode, reading one frequency at a time on their *Read Environment* phase described in Section 4.1.

In this chapter, we motivate the reasoning behind the design of our self-organizing WMN agent and why they can support the application of SDN at large scale WMNs. Also, we present their detailed design, discuss how they gather information, we present conditions for convergence, and, finally, we present experimental results about the convergence, effort to convergence, and resulting topology structure. I published part of the content of this chapter in [109].

5.1 Design of Self-Organizing WMN nodes

Our self-organizing agents evolve a large node placement (a set of geographical positions) of wireless mesh nodes into partitions holding the properties of bounded diameter and node degree. The network formation is the result of a distributed algorithm executed by the agents in the WMN nodes (software components) which independently make decisions that collectively lead to the final topology. Such decisions are based on the agent's design that enforces its safety and liveness properties. The organization occurs both at the physical and link layers. The physical isolation allows using orthogonal frequencies on different partitions. At the link-layer, the use of different network IDs (such as the BSSID - Basic Service Set ID) creates logical isolation to provide robustness to the case of equal

frequency in neighboring partitions.

We call this design the *Smart* agent. The WMN partition diameter property bounds the communication latency, a critical requirement for intra-group SDN-based network control planes. The autonomic aspect is fundamental to this solution, considering that the graph partitioning is an NP-Hard problem [66] or NP-Complete under specific assumptions [67].

The second safety property of *Smart* is the control of the mesh node connectivity degree. In the SDN paradigm, this design bounds the number of events per new data flow handled by an SDN controller of partitions. Transmissions of a node n_i in a WMN are received by all its neighbors given an inherently broadcast nature of wireless communication. In a path of distance h for an average WMN node degree dg, $ev \approx dg \cdot h$ events arrive at the controller for every new flow initiated given the on-demand control nature of SDN. Therefore, the bounded node degree limits the per-flow workload regarding network control events.

Moreover, the combined bounding on diameter and node degree limits the number of nodes per partition (the Degree/Diameter Graph problem [14]) regardless of any underlying node placement density to support the precise workload control in WMN SDN controllers. Finally, our self-organization design also solves the question of electing a mesh node to act as an SDN controller. Similarly to the leader election concept in distributed consensus protocols [110], partitions evolve from a unique origin node, a clear candidate to act as the partition controller.

5.2 Autonomic behavior of agents

5.2.1 Smart node agent design

The *Smart* agent joins the largest nearby partition for which it finds the shortest path of hop-distance at most h to the partition origin. It is trivial to verify that the diameter of the partition is, at most, $d = 2 \times h$: the origin node is a member of any shortest path from border to the origin, and the diameter unites two shortest paths from origin to the border.

The *is origin* attribute is true for an instance of *Smart* which creates a partition, and false otherwise.

Optionally, *Smart* nodes assume a node degree dg upper bound constraint to decide on their partition membership. If no membership option is valid, *Smart* nodes create a new partition. They continually review their membership decision following the generic agent cycle described in Section 5.1.

Smart attempts to change its partition in every new cycle to the largest *valid* neigh-

boring partition which is, at least, sp percent larger (a threshold) than the agent's current partition.

A *valid* neighboring partition has *i)* at least one node at comm. reach of the deciding node, *ii)* all nodes not violating safety properties, *iii)* all nodes will not violate safety properties after the addition of the deciding node.

More formally, let m be a Smart agent node reviewing its properties. Let P be the set of all partitions while $P_m \subseteq P$ is the set of all nearby partitions to m; let $k \in P_m$ be a nearby partition, o_k is the origin node of partition k. Let k_c be the current partition of m if it is already connected, and $k_c \in P_m$.

Let $F_{distance\_sp}(i, j)$ be a function that returns the shortest-path distance between two nodes. Let $F_{DT} \rightarrow P_m$ be a function that eliminates nearby partitions k for which $F_{distance\_sp}(m, o_k) > h, \forall k \in P_m$.

Let $F_{degree}(i)$ be a function that returns the degree of a node i; let $F_{DG} \rightarrow P_m$ be a function that removes partitions $k \in P_m$ if there exists any node $i \in k$ such that i will be/is a neighbor to m and $F_{degree}(i) \geq dg, \forall i \in k, \forall k \in P_m$.

Let $F_{neighbors}(k, i)$ be a function that returns the number of future neighbors of a node i if it joins a neighboring partition $k \in P_m$; let $F_{NEIGH} \rightarrow P_m$ be a function that removes partitions $k \in P_m$ if $F_{neighbors}(k, m) > dg, \forall k \in P_m$.

More formally, let $F_{size}(k)$ be a function that returns the size in number of nodes of a partition k, let s_m be the size of the current partition k_c of the agent m, let $F_{ES} \rightarrow P_m$ be a function that *i)* eliminates partitions $k \in P_m$ for which $F_{size}(k) < (1 + sp) \times s_m$, and *ii)* sorts P_m by $F_{size}(k), \forall k \in P_m$ into the ordered list $P_m^{'}$.

If $P_m^{'} \neq \varnothing$, the first item $P_m^{'}[0] = k_b$ is the best partition membership option regarding size. If $P_m^{'} = \varnothing$ and $k_c \in P_m$ (current partition k_c checked as *valid* regarding safety properties), we add back k_c into $P_m^{'}$ as an option (stay on current partition).

On its second phase of the generic agent cycle, *Smart* executes: $F_{DT} \rightarrow P_m$, $F_{DG} \rightarrow P_m$, $F_{NEIGH} \rightarrow P_m$, and finally $P_m^{'} = F_{ES} \rightarrow P_m$.

Let A_S be the decision of *Smart* as a set of actions:

$$A_S = \begin{cases} \text{If } P_m^{'} = \varnothing & : \{create\_partition\} \\ \text{Else if } P_m^{'} = \{k_c\} & : \varnothing \quad \text{(a null action set)} \\ \text{Otherwise} & : \{leave\_partition(k_c), \\ & \quad join\_partition(k_b)\} \end{cases} \tag{5.1}$$

We evaluated the operation of versions of *Smart* holding different values for the safety property *maximum node degree*: 5, 10, and no degree constraint.

5.3 Agent information

Agents base their decisions on local information (internal and neighbors), independent of network connectivity. They read the environment (wireless scan) at the link-layer, which also include physical layer attributes (*e.g.*, operating frequency). If connected to a network, nodes can obtain attributes from their routing layer (*e.g.*, partition size in nodes derived from the node's routing table, or an internal state of an SDN controller propagated to its managed SDN nodes). Nodes do not rely on synchronous communication to each other at the time of decision making (*e.g.*, a network layer send-receive operation) neither information forwarding (such as in a distributed protocol). Such a design is robust to connectivity failures, supporting self-organization in the absence of any initial input. It resembles a form of autonomic bootstrapping similar to the concept of self-stabilization [12].

Nodes share (broadcast) their local properties at frequent intervals as part of the wireless link-layer control protocol; nodes read the environment (receive broadcasts) also frequently. The broadcast interval b is much smaller than the maximum observation interval: $b \ll o$, and $o = b \times k$ (k an integer constant that supports observing the number of frequencies c in use: $k \geq c$). Therefore, an observation action captures complete shared information from neighbors with high probability. In the WiFi standard, the default broadcast period b is 0.1 sec for the link-layer control protocol (beacon interval) [111]. Our experiments have epochs (the generic agent cycle) of $e = 90$ sec to accommodate *a) observation and decision making* of agents integrated to the *b) synchronous network communication* that determines the capacity of partitions. Precisely, *a)* takes place in the initial 10% of the epoch while in the remaining 90% the agent is idle, moment reserved for *b)*.

Wireless standards define management packets that share internal network attributes such as The *Information Elements* field on a set of management packets of the IEEE 802.11 standards [111] exemplify a data structure for attribute sharing in a real execution setting. Through the scan of multiple frequencies, a node can *listen to* management messages of other wireless nodes (on any nearby partition) to collect needed information.

In this work, the autonomic agents rely on a simulated agent platform (Section 3), using the module ASim to obtain their information. Section 5.4 presents a detailed analysis of the convergence of *Smart* under concurrent operation supported by ASim.

5.4 Convergence of the Smart agent

In this section, we analyze conditions for the consensus to a stable partition membership solution given the autonomic properties of the *Smart* agent. Convergence is an essential outcome of the network organization; a non-converging solution stresses the network control plane on routing information update in the distributed routing schemes or on managing a large number of network control events on a SDN controller. First, we comment on the behaviors of *Smart* and how they impact convergence. Next, we introduce two optimization alternatives. Finally, we derive an analytical expression of divergence indication P_D given the agent properties, the agent operating cycle, node placement density, and wireless communication reach. The wireless reach, in turn, is the result of the other parameters in the wireless system. P_D holds a correlation with the convergence time and the effort to convergence; therefore, it allows comparing the impact of different choices of configuration sets. P_D is not a probabilistic indication of divergence.

5.4.1 Triggers for slow convergence

We recall that the liveness property motivates a node to review its current settings in favor of improved ones when comparing to the current option. Safety properties are hard limits not to be violated, restricting the set of possible partition membership options.

Each autonomic property will lead to violations in different conditions regarding joining or leaving a partition. The versions of *Smart* comprise different combinations of safety and liveness properties, thus leading to different expectations regarding their impact to a stable partitioning.

Solving a violation in the agent's current partition membership requires leaving the current partition in favor of another (perceivably valid) option. In attempting to maintain its liveness by always choosing its best option, a node can change its partition membership, also implying a sequence of leave, join partitions.

When leaving a partition, a node can lead other nodes to violations. The lemma that follows describes these violations.

Lemma 5.1 (Leaving partitions)**.** *The action of leaving partitions can cause safety violations of the diameter bound property.*

Proof. Assume a partition P with its origin node $o_P \in P$. We define disconnected nodes to a partition P as nodes in any segment S_i of P so that $o_P \notin S_i$ (not in the segment of the origin node). A node $n_i \in P$ leaving its partition can create segments in P thus disconnecting nodes $n_j \in P$ if $n_j \in S_i$ and $o_P \notin S_i$. A disconnected node is in a safety

violation condition given its infinite distance to the origin node.

Moreover, n_i can induce increased distance to the origin node $o_P \in P$ due to the elimination of the shortest path of a node $n_j \in P$ to the origin. If the distance of n_j to the origin goes above the limit h, the node n_j is in safety violation. □

When joining partitions, the concurrent execution of the agents' *read + evaluate + act (REA)* phases could lead to decisions based on outdated information which could lead to safety property violations. We analyze *Smart*'s two safety properties, partition diameter bound, and node degree bound regarding the *join* event in the two lemmas that follow.

Lemma 5.2 (Addition of nodes on the diameter property)**.** *The concurrent addition of nodes will not induce violations of the diameter bound property.*

Proof. Assume the concurrent addition of two new nodes n_1, n_2 to a partition P. The concurrent *REA* does not change the condition that n_1, n_2 used, say, the existence of a node $n_0 \in P$ with distance to the partition origin at most $h - 1$. Therefore, although both n_1 and n_2 were not aware of each other as future neighbors when they made their *join* decision, they end up with a valid distance of h based on their connectivity to n_0. □

Lemma 5.3 (Addition of nodes on the node degree property)**.** *The concurrent addition of nodes can induce violations of the maximum degree bound property.*

Proof. Assume a partition P and the existence of a node $n_0 \in P$ with degree $dg - 1$ (dg **:** the maximum node degree bound). If the nodes $n_1 \notin P, n_2 \notin P$ concurrently join the partition P using an outdated perception of n_0's degree below the limit, they unintentionally induce a degree $dg + 1$ onto n_0, causing a safety violation. □

In summary, the version *Smart-any* of the agent: *1)* only induces a violation of the maximum distance to origin property when *leaving a partition* to maintain its liveness or to solve a violation induced to it by other nodes that left its partition.

The version *Smart-dg* can induce violations of the maximum distance to origin property *1)* when *leaving a partition 1a)* to maintain its liveness or *1b)* to solve a safety violation. Also, *Smart-dg* can induce violations of the maximum node degree property *2)* when *joining a partition* due to outdated information given concurrent *REA*.

Finally, the following theorem illustrates a divergence scenario by *Smart-dg*.

Theorem 5.1 (*Smart-dg* divergence)**.** Smart-dg *can diverge to obtain a stable partitioning.*

Proof. Assume two nearby nodes n_1, n_2 that both can join partitions P_i, P_j. Also, assume the interval l comprising the *REA* phases of the generic agent cycle. If, say, n_2 starts reading the environment after n_1, there exists the chance that n_2's evaluation will not reflect the effects of n_1's action into the environment. As a consequence, n_1 and n_2 could decide to join the same partition, say, P_i, and cause a violation of the maximum node degree safety property. This is the violation described in the Lemma 5.3. If we assume no change to the timing of the agents' cycles, the same concurrent *REA* effect could occur later and lead both to decide on joining the other partition, say, P_j. Again, it could lead to a safety property violation of the Lemma 5.3. Therefore, it is possible that n_1, n_2 enter a continued change of partition memberships, characterizing a divergence to a stable partitioning. □

5.4.2 Optimizations to improve convergence

The key challenge to reduce convergence time is letting the agents achieve an ordering of execution of their *REA* such that it minimizes the impact of concurrency on corrupting the environment information.

A first solution for achieving such a *minimal corruption ordering* is the randomization of the agent execution on the interval l of the agent cycle after detecting a safety property violation. We named it *RAND*. Once detecting itself in an invalid state, the agent introduces a random delay on the start of its next *REA*, attempting to induce a separation to the *REA* of other nearby agents. Other problems related to concurrent operation also applied randomized delays such as accelerating the convergence of the leader election on the Raft consensus protocol [110] and providing fairness to the channel access mechanism of the DCF (distributed coordination function) used in WiFi [111].

A second optimization for improving convergence consists in defining an arbitrary ordering for solving violations amongst nodes of a partition. However, distributed agents agreeing on a unique sequence is a complex problem [13, 70, 110, 112–115]. We apply a relaxed ordering based on the distance of nodes from the partition origin: nodes closer to the origin wait for their neighbors which are away from the origin and also in violation. This simple design creates a partial ordering that resembles prioritizing nodes closer to the border of partitions for violation solution. This optimization is named *PORD*.

5.4.3 Modeling divergence

Here we model a divergence indication for the case of agents applying no optimizations for violation solution. We split the membership divergence scenario of *Smart-dg* in three conditions: *i)* the proximity of nodes letting them to interfere on each other's environment,

ii) the probability of some concurrency level on nodes' *REA* phases (P_C), *iii)* the probability of impact given the concurrency level (P_I). While *i)* is a function of node density and probability distribution for nodes' positions; the other two conditions are functions of the timing parameters of the generic agent cycle. First, we delve into conditions *ii), iii)* to derive probability expressions based on the parameters of the generic agent cycle. Later, we combine *i)* to derive the divergence indicator P_D.

As described in Section 5.3, let b be the agent's broadcast interval, c the number of frequencies to observe, $k \geq c$ a constant that allows reading all frequencies, and the combined *REA* interval $l = b \times (k+1)$, assuming the duration of *evaluate* + *act* as b for simplicity. Finally, e the duration of the total agent cycle $Y = \{b_1, b_2, ..., b_d\}$, $e \geq l$, and $d = e/b = |Y|$ the number of broadcast intervals d in the epoch e.

We model P_I as a linear function of the level of concurrency of two agents, assuming that all frequencies are equally likely used by any given partition. Therefore, the largest the concurrency level, the higher the likelihood that the given partition frequency will be read by two agents before an action, inducing corrupt environment information. Let i be the number of concurrent broadcast intervals b of two nodes:

$$P_I = \frac{i \cdot b}{l} = \frac{i \cdot b}{b \cdot (k+1)} = \frac{i}{(k+1)} \tag{5.2}$$

Assuming that the starting times of the agents' cycles are independent, the probability of an agent starting its cycle at any interval b is b/e. Therefore, the probability of concurrency P_C between any two nodes n_1, n_2 is:

$$P_C = \left(\frac{b}{e}\right)^2 \tag{5.3}$$

Combining the probabilities of concurrency P_C and impact P_I for any starting interval $b \in Y$:

$$P_C \cdot P_I, \ \forall b \in Y = P_{CI} = \frac{e}{b} \cdot \left(\frac{b}{e}\right)^2 \cdot \sum_{j=1}^{k+1} \frac{j}{k+1} \tag{5.4}$$

$$P_{CI} = \left(\frac{b}{e}\right) \cdot \sum_{j=1}^{k+1} \frac{j}{k+1} \tag{5.5}$$

Replacing the summation in Equation 5.5 by the expression for an arithmetic series:

$$P_{CI} = \left(\frac{b}{e}\right) \cdot \frac{k+2}{2} \tag{5.6}$$

Finally, we combine parameters of the node placement density and wireless range

to derive the divergence indicator P_D. We recall that a divergence scenario requires at least one node continually changing decisions over time. For that, a node n_i needs to be affected by, at least, concurrent decisions of a neighboring node, or by concurrent decisions of neighbors of its neighbors. The former directly affects attributes of the node n_i while the latter can induce changes to neighbors ($neigh(n_i)$) that are observed by n_i. Therefore, nodes in an area $A = f(2r)$ of a reference node n_i can directly induce safety property violations. Given our node placement density N_D, the expected value E_n of nodes in a circular area A is $E_n = N_D \cdot A$ that we express below:

$$E_n = \pi(2r)^2 \cdot N_D = 4\pi r^2 \cdot N_D \tag{5.7}$$

Combining Equations 5.6 and 5.7 derives the divergence indicator P_D as a function of the parameters in the agent's cycle, the density of nodes, and the wireless connectivity range r:

$$P_D = P_{CI} \cdot (E_n - 1) = \left(\frac{b}{e}\right) \cdot \frac{k+2}{2} \cdot (4\pi r^2 \cdot N_D - 1) \tag{5.8}$$

$$P_D = \left(\frac{b}{e}\right) \cdot \frac{k+2}{2} \cdot (4\pi r^2 \cdot N_D - 1) \tag{5.9}$$

From Equation 5.9, reducing the indication P_D implies *a)* decreasing b, k; *b)* decreasing the number of nodes involved in E_n; or *c)* increasing the epoch e. Any of these alternatives create undesired collateral effects; however, the optimizations described in sub-section 5.4.2 achieves *b)* given its ordering approach without enforcing restrictions in the communication range r or node density N_D. Furthermore, the increase of e has diminishing returns, given its inverse relation to P_D. We comment on the validity of Equation 5.9 when evaluating experimental results in Section 5.6.2.

5.5 Visual outcome of Self-Organizing agents

Figure 5.1 represents the maximum *underlying* connectivity if all nodes operate on the same frequency in the physical layer, the same logical network (SSID, BSSID) at the link-layer, using their nominal transmit power. Therefore, no additional connectivity can exist. Node placement based on the U.S. buildings data set from [98].

More formally: let $G(V, E, P)$ be a graph representing the connectivity topology of a WMN where V is the set of nodes, E is the set of edges, P the set of positions of the nodes in V, n is a node such that $n \in V$, f_n, t_n are frequency and network id of n; E is maximal if $\forall i, j \in V, f_i = f_j, t_i = t_j$.

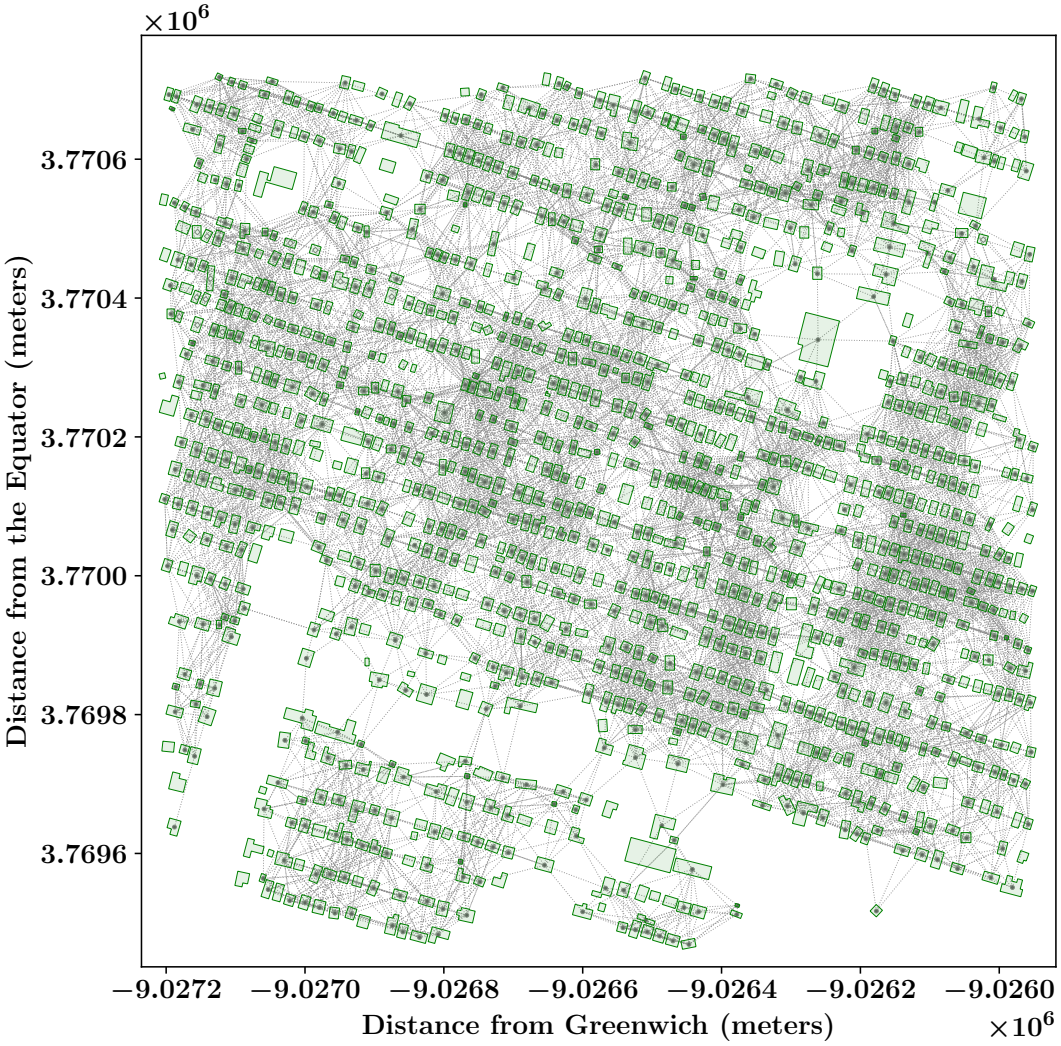


Figure 5.1: Underlying maximum possible connectivity of 1000 nodes on a $\approx 1.44\ Km^2$ region in Chatham county, GA (nearby Savannah, GA). Shows the maximum set of neighboring options (or a single partition solution). Wireless standard IEEE 802.11a. Density of $1/1600\ node/m^2$.

The choices of the self-organizing agent on the WMN nodes define the partitioned WMN topology (Figure 5.2). The same node placement is the basis of the two topologies. Partitions in Figure 5.2 are degree/diameter constrained. More formally: let $G_{5.1}(V_{5.1}, E_{5.1}, P_{5.1})$, $G_{5.2}(V_{5.2}, E_{5.2}, P_{5.2})$ be graphs representing the topologies of Figures 5.1, 5.2, respectively; $V_{5.1} = V_{5.2}$, $P_{5.1} = P_{5.2}$, $E_{5.1} \supseteq E_{5.2}$.

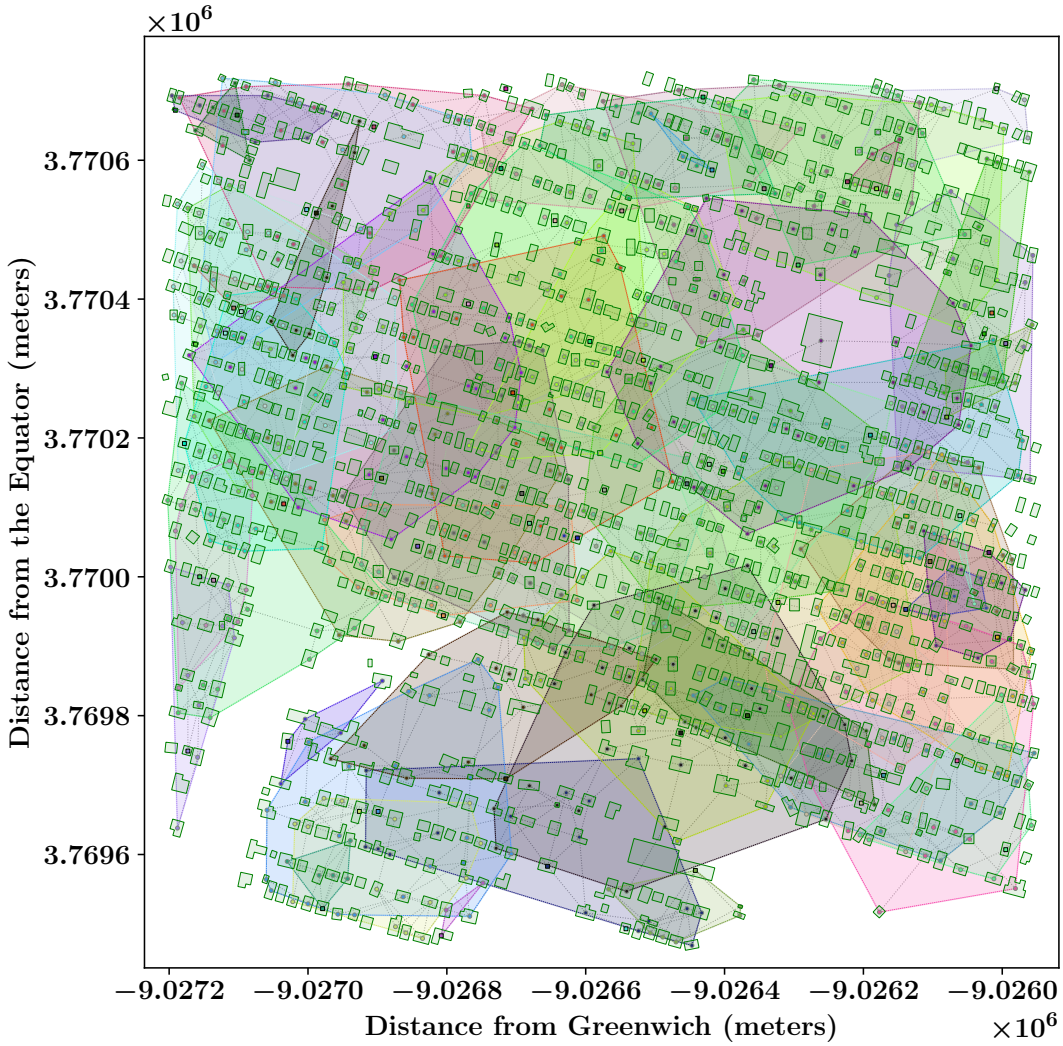


Figure 5.2: Self-organized WMN topology partitioned by the *Smart* agent, producing overlapping partitions, made evident by the partitions' convex-hull. Same wireless standard and node placement of Figure 5.1. Max. node degree 5, max. partition diameter 6.

5.6 Results

This section presents the experimentation settings, and results of agents convergence. Results derive from experiments conducted using the experimentation platform described in Chapter 3.

5.6.1 Experimentation settings

The following list describes node placements and settings of the wireless subsystem.

1. Nodes have randomized positions controlled by a *node placement random variable*

(uniformly random). We used a common set of 30 different *seeds* to all agent models, creating 30 different node placements.

2. We control the average node placement density over the experimentation area as described in Section 5.6.2.
3. Nodes use the IEEE 802.11a wireless standard. We use the well known ns-3 simulator [73] to model the wireless stack of mesh nodes.
4. The mesh nodes' IEEE 802.11a physical layer (5 GHz band) with a 20 MHz bandwidth. The tx power is 16 dBm (default). The gain of the tx/rx antennas is 1 dBi (also defaults). The CCA (Clear Channel Assessment[1]) threshold is −99 dBm. The Energy Detection Threshold[2] is −96 dBm. The last two also default values.
5. The link-layer of the mesh nodes uses the IBSS (Independent Basic Service Set) mode, creating WMNs through multi-point association.

5.6.2 Experiments evaluating convergence

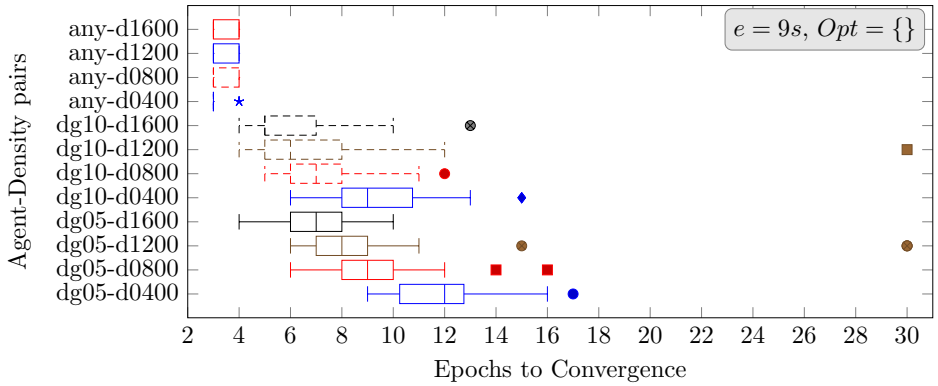


Figure 5.3: Convergence without optimizations for small epochs ($e = 9s$): the slowest conv. of up to 16 epochs, and 30 for outlier cases (diverged).

We simulated the convergence of *Smart* enabling *REA* concurrency, a real scenario in which agents can decide based on outdated information.

For the results in this section, assume $b = 0.1, k = 8$, $r = 100$ (IEEE 802.11a). The plots show results for 12 agent-density pairs of *Smart* which are a combination of one degree constraint in $dg = \{any, 10, 5\}$ with one node density inverse in $\{1600, 1200, 800, 400\}$ $m^2/nodes$. Each agent-density result combines experiments using 30 different node placements (NP), and each NP with a total of 400 nodes. Each experiment had a total time of 30 epochs. At least 90% of nodes are spawned in the first epoch to represent the extreme case of 90% node churn on the return of a power outage. The sp parameter

[1] Identifies the channel as free for transmission.

[2] Triggers the start of packet reception.

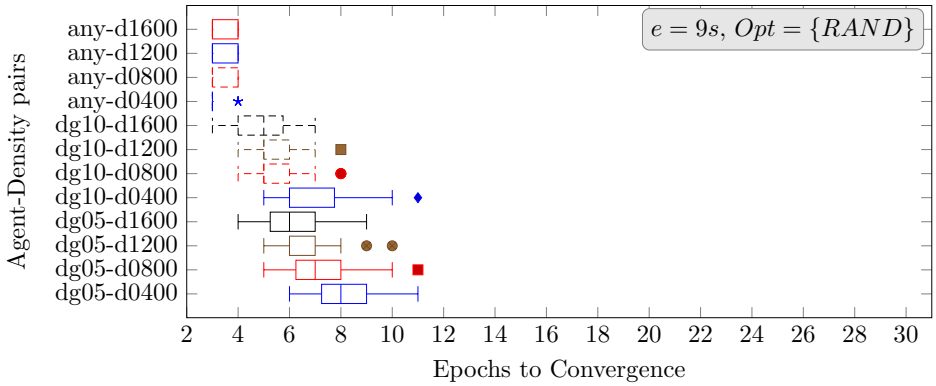


Figure 5.4: Convergence with randomization for small epochs (e = $9s$): improved convergence up to ≈ 11 epochs.

of the *partition size* liveness property is 0.2 for the origin node and 0.1 for any other node. Small values of sp induce the formation of the largest possible partitions given the limitation imposed by the combination of the diameter and maximum node degree safety properties. However, small values of sp might impose longer convergence in the number of epochs. The evaluation of the appropriate ranges of values for sp will be the subject of future work.

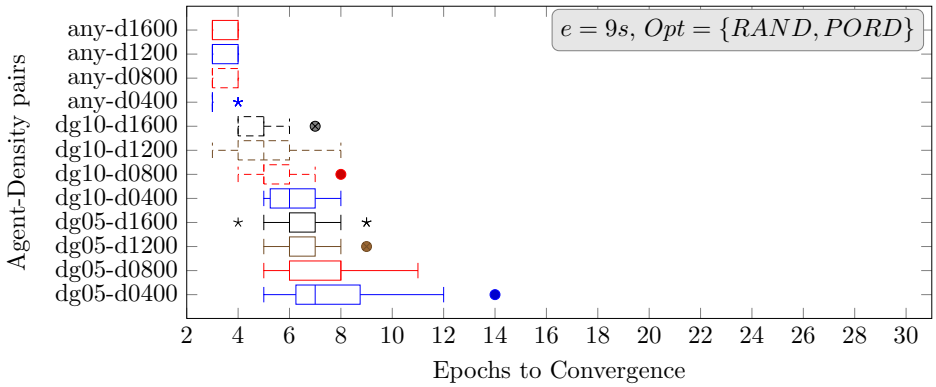


Figure 5.5: Convergence combining randomization and sequencing (e = $9s$): a small improvement over randomization, up to ≈ 8 epochs except for $dg05$ on node densities $1/800, 1/400$ $nodes/m^2$.

Using Figure 5.3, in which agents apply no optimizations for solving safety violations, we confirm assumptions of the analytical model for divergence of Section 5.4: *a)* the *Smart-any* agent has much faster convergence; *b)* increased density of nodes increases the time to convergence. We add that a more constrained degree (*e.g.*, dg = 5) increases convergence time. Although the P_D expression does not capture the aspect of degree constraint as a parameter, the intuition of its impact is that a more constrained degree (smaller dg value) will increase the likelihood of more nodes in violation which explains

the increase in the convergence time.

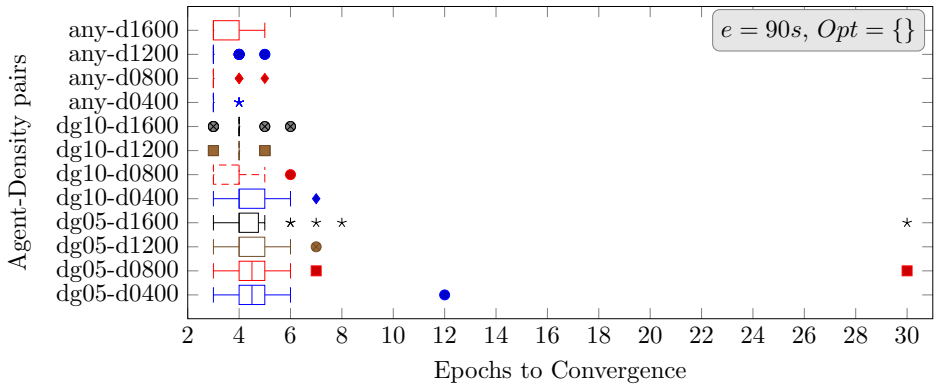


Figure 5.6: Convergence without optimizations for a large epoch (e = $90s$): better results than a small epoch and the combined optimizations of Figure 5.5; however, not by a large margin. Divergence cases found.

When comparing the slow convergence shown on Figure 5.3 to the faster convergence of Figures 5.4 and 5.5, we verify the importance of optimizations to cope with small epochs and highly constrained maximum degrees (*e.g.*, dg = 5). Small epochs allow for faster absolute convergence while the highly constrained degree improves capacity and reduces the number of control events in an SDN setting (see discussions on Section 5.1). Moreover, Figure 5.3 presents cases of divergence: the outlier points with convergence time of 30 epochs did not converge until the end of the experiment.

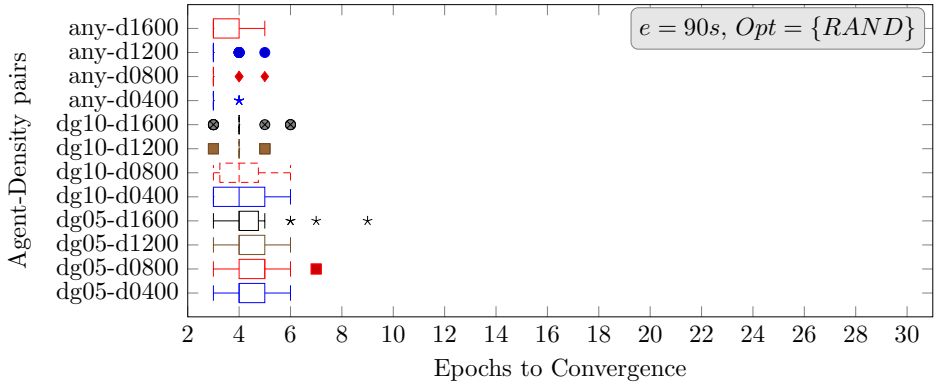


Figure 5.7: Convergence with randomization for a large epoch (e = $90s$): a small improvement in number of epochs over the setting of large epochs without optimization.

Finally, we confirm the positive impact on convergence (faster convergence in the number of epochs) of increasing the epoch time. Figures 5.6 and 5.7 present the fastest convergence in the number of epochs. However, in terms of absolute time they represent a slower total convergence. For epochs e = 9 sec, assuming an upper bound of U_b = 9 epochs (Figure 5.5), the convergence time is $C \approx 9 \times 9 = 81$ sec. For epochs e = 90 sec,

an upper bound of U_b = 6 epochs (Figure 5.7) renders a convergence time of $C \approx 90 \times 6 = 540$ sec.

Furthermore, we verify a tendency of diminishing returns on the increment of the epoch length for the $10x$ experimented; this effect is captured in P_D when analyzing it as a function of the epoch: $P_D(e) = 1/e$.

5.6.3 Effort to convergence

Figures 5.8, 5.9, 5.10 present the number of node partition membership changes during the convergence process. Figure 5.10 shows one additional benefit of the $PORD$ optimization: a significant reduction on the number of node partition membership changes. This reduction relieves the stress on the adaptations of upper layers (layers 2-3) to the topological changes induced by the mesh node agents.

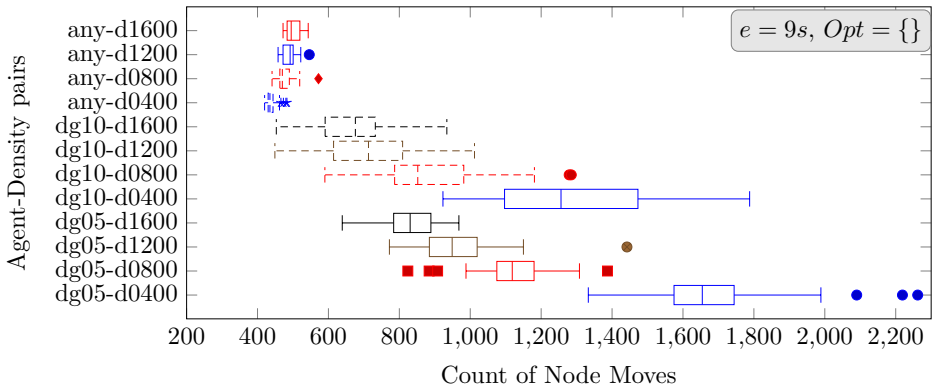


Figure 5.8: Total of *move* events when a node changes partition memberships (e = $9s$). No optimizations.

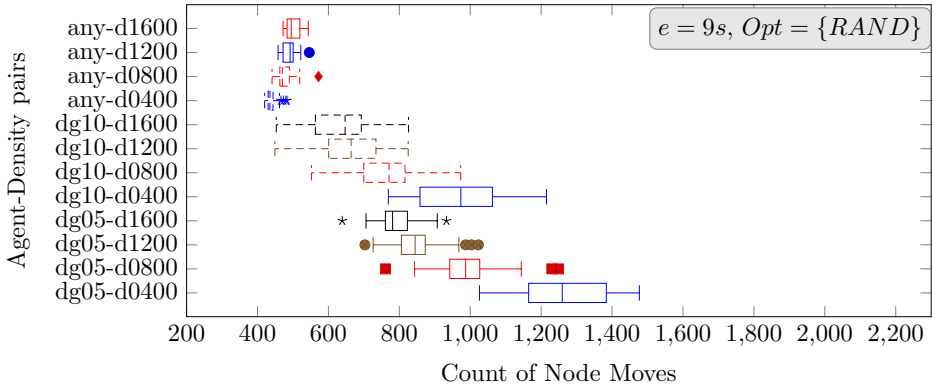


Figure 5.9: Total of *move* events when a node changes partition memberships (e = $9s$). Optimization $RAND$.

Figures 5.11, 5.12, 5.13 present the number of nodes detecting safety violations during the process to convergence. Figure 5.13 shows a significant reduction of the number of

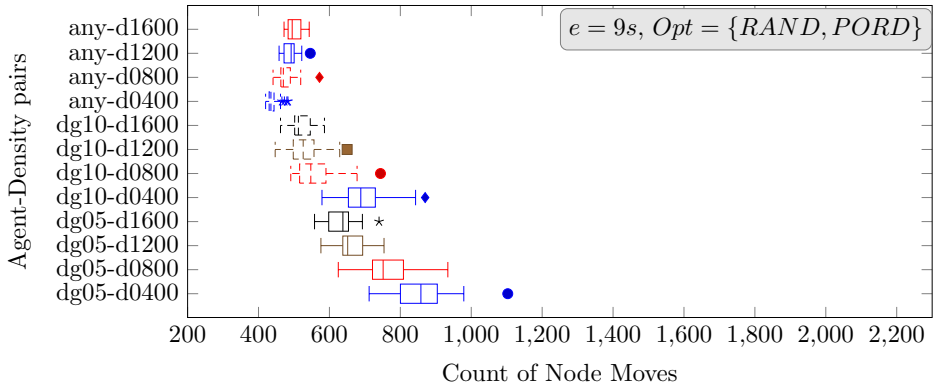


Figure 5.10: Total of *move* events when a node changes partition memberships (e = $9s$). Optimizations $RAND, PORD$. A significant reduction on the number of node movements due to the partial ordering optimization ($PORD$).

nodes detecting safety violations, indicating a smoother evolution to a stable partition set. Again, a contribution from $PORD$.

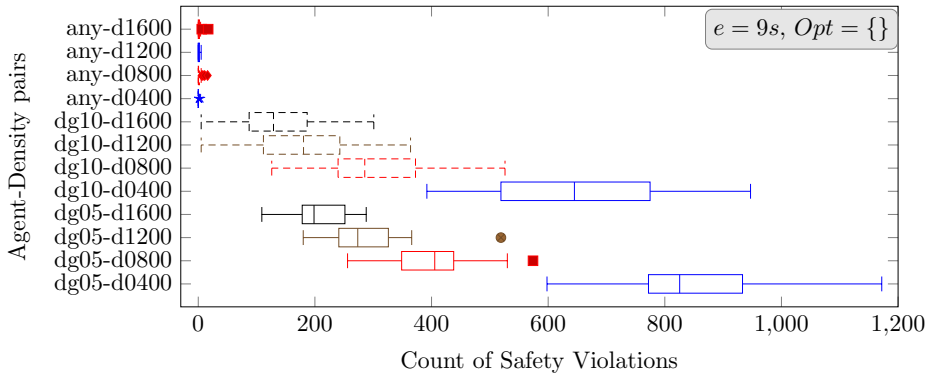


Figure 5.11: Total of *safety violation* events when a node identifies itself in an invalid state regarding its autonomic properties (e = $9s$). No optimizations.

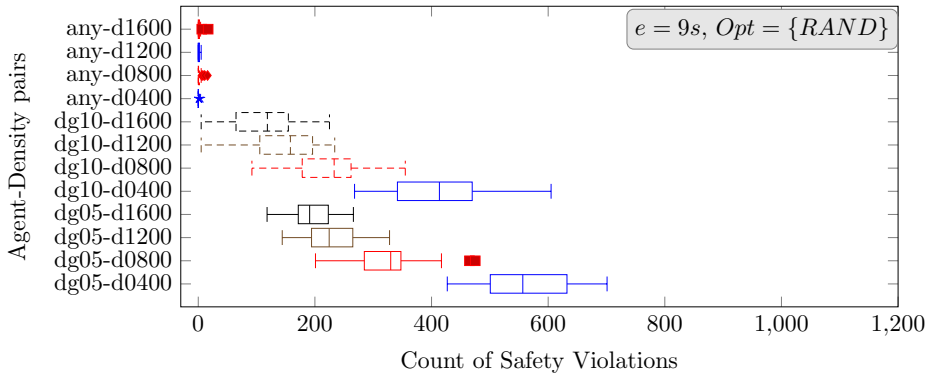


Figure 5.12: Total of *safety violation* events when a node identifies itself in an invalid state regarding its autonomic properties (e = $9s$). Optimization $RAND$.

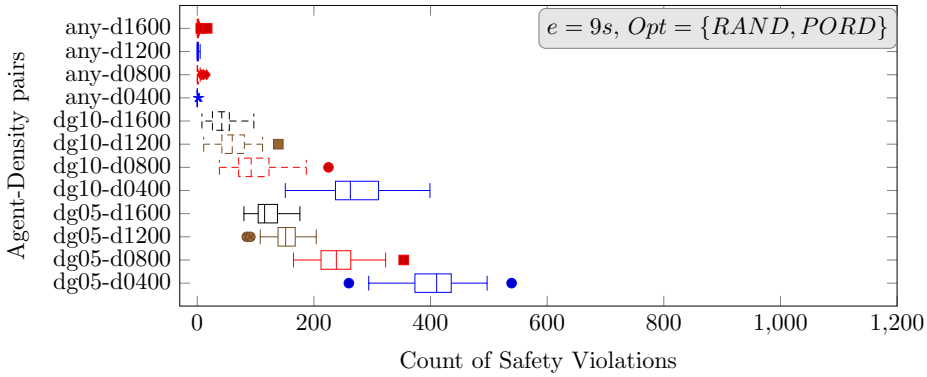


Figure 5.13: Total of *safety violation* events when a node identifies itself in an invalid state regarding its autonomic properties (e = $9s$). Optimization $RAND, PORD$.

Figures 5.14, 5.15, 5.16 provide insights on the form of violations occurred while evolving to a stable partition set. Each figure represents the maximum value observed for the safety property over all 30 experiments (each a different node placement) conducted for the given configuration of: agent degree, node density.

Figures 5.14, 5.15 present the evolution of the maximum node degree in the experiment set over time (epochs) while Figures 5.16, 5.17 present the evolution of the maximum partition diameter in the experiment set over time (epochs). The initially erratic control of the safety properties later converged to the defined objectives.

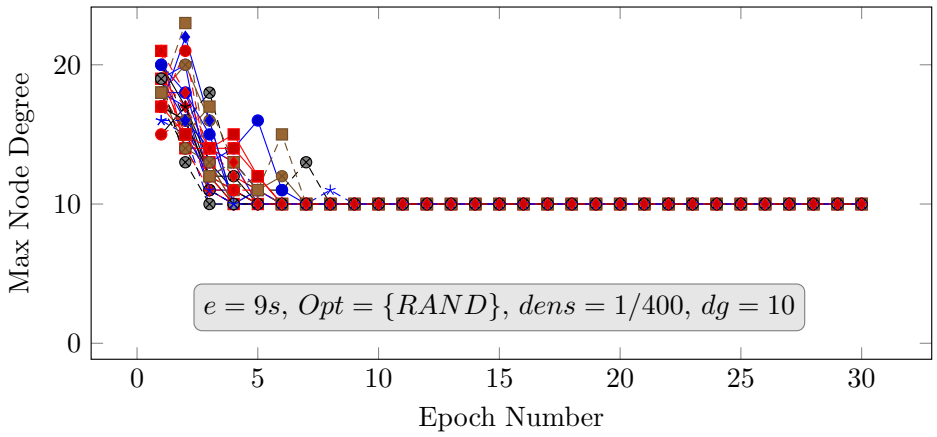


Figure 5.14: The maximum node degree of a mesh node for the agent smart $dg10$ on density $1/400$ $nodes/m^2$ (e = $9s$). Optimization $RAND$. An ineffective control of the maximum node degree that converges to the objective of a maximum node degree of 10.

5.6.4 Resulting WMN partitioning structure

This section analyzes the resulting partitioned WMN topology regarding the size of the partitions, recalling that the combined node degree and partition diameter constraints

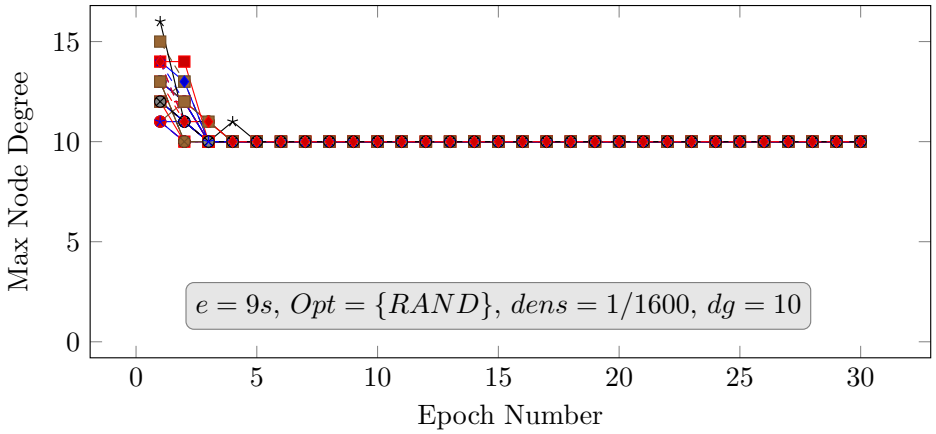


Figure 5.15: The maximum node degree of a mesh node for the agent smart $dg10$ on density $1/1600\ nodes/m^2$ (e = $9s$). Optimization $RAND$. Again, ineffective control of the maximum node degree initially that later converges to the objective of a maximum node degree of 10. A reduction on the error of the maximum degree when assuming a reduced density.

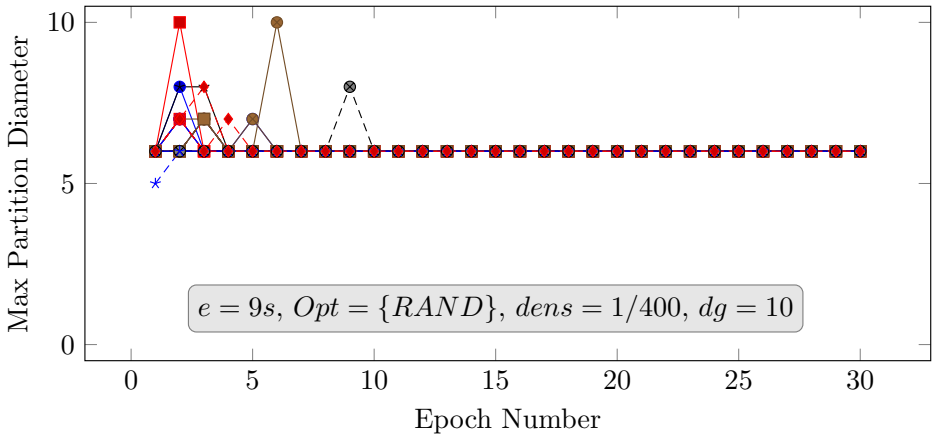


Figure 5.16: The maximum WMN partition diameter for the agent smart $dg10$ on density $1/400\ nodes/m^2$ (e = $9s$). Optimization $RAND$. An initially ineffective control of the partition diameter that converges to the objective of a maximum diameter of 6. The controlling error is higher for this higher density of $1/400\ nodes/m^2$.

result in a bounded size for the partitions: the Degree/Diameter Graph problem [14].

The resulting partitioning structure is a relevant outcome regarding recovering the global connectivity of the WMN, a function performed by self-healing agents in the mesh nodes. A highly fragmented partition set with mostly small partitions (below the maximum size possible given the degree/diameter constraint) implies the need of a larger number of nodes with a second wireless interface to interconnect partitions. Moreover, a large number of partitions when compared to the available set of frequencies might render frequency diversity inefficient due to the reuse of frequencies by nearby WMN partitions.

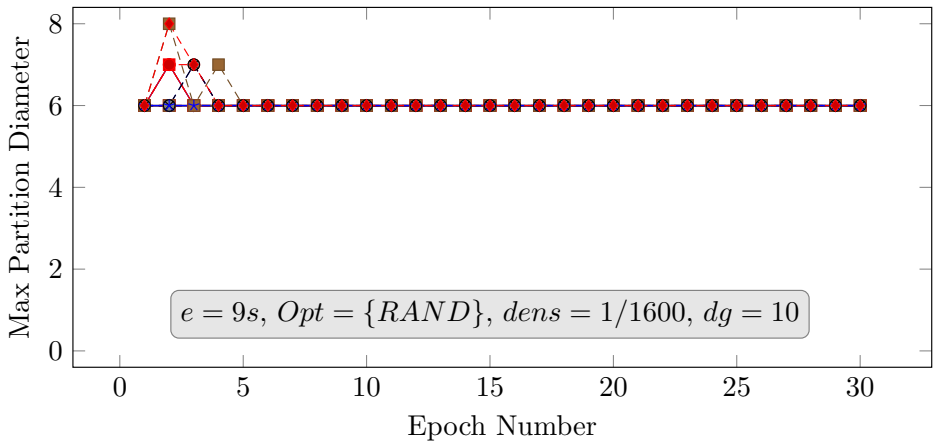


Figure 5.17: The maximum node degree of a mesh node for the agent smart $dg10$ on density $1/1600$ $nodes/m^2$ (e = $9s$). Optimization $RAND$. Again, ineffective control of the partition diameter initially that later converges to the objective of a maximum diameter of 6.

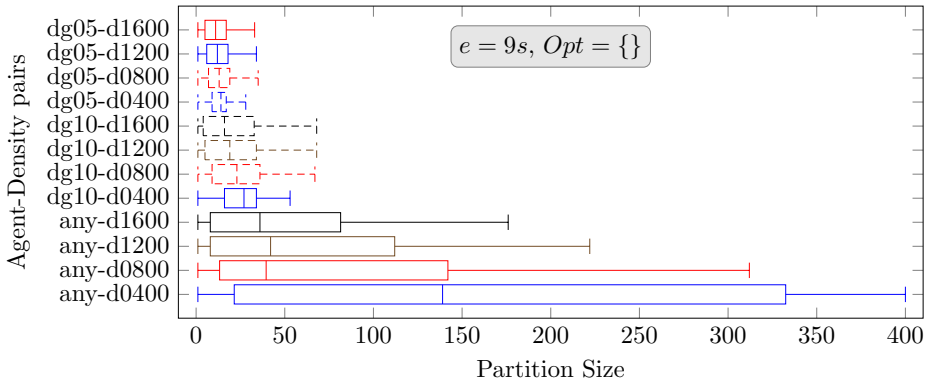


Figure 5.18: Typical WMN partition sizes. No optimization.

Therefore, the ideal outcome a partition set with the largest possible size to minimize the requirement of mesh nodes with a second wireless interface for inter-partition connectivity and efficient frequency diversity.

Figures 5.18,5.19, and 5.20 confirm the bounded sizes as outcomes of the partitioning determined by *Smart-dg*. For *Smart-any*, we verify that the undesirable outcome of a single partition containing all 400 nodes is possible for high node densities. Moreover, smaller degrees induce smaller maximum partition sizes. Furthermore, the different optimizations applied did not change the outcome regarding partition sizes.

Under the highest node density of $1/400$ $nodes/m^2$, we verify the tendency of not forming the highest possible partitions regarding size, which is an undesired outcome.

Figures 5.21 and 5.22 present the frequency of partition sizes over all 30 experimented node placements. The bin width for the sizes was 20. Each plot shows results of two

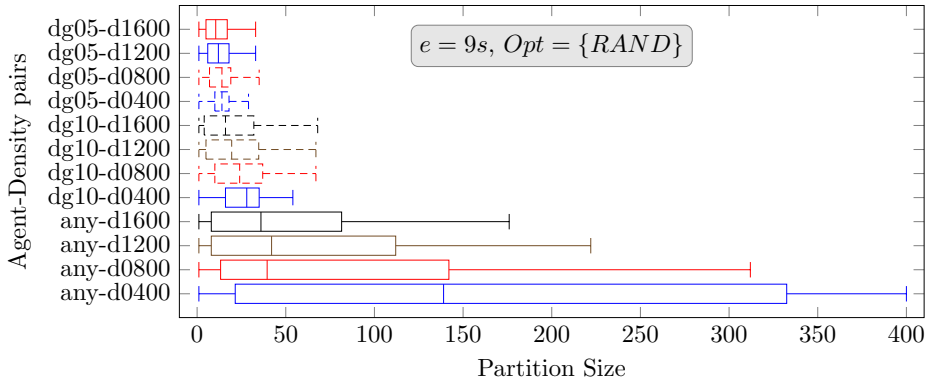


Figure 5.19: Typical WMN partition sizes. Optimization $RAND$.

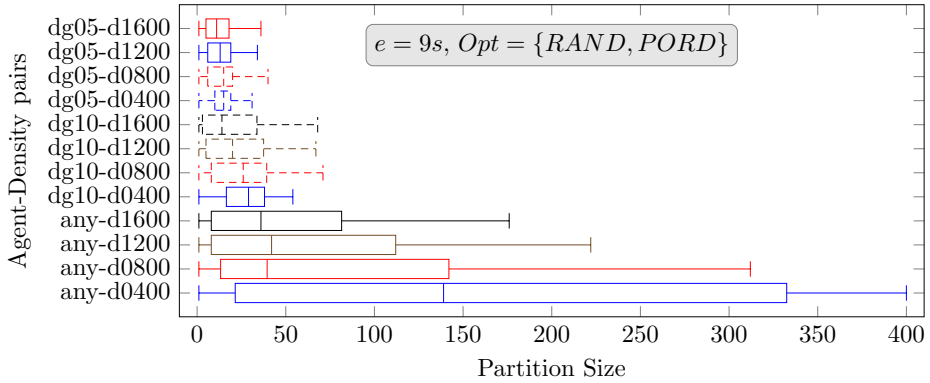


Figure 5.20: Typical WMN partition sizes. Optimization $RAND, PORD$.

versions of *Smart-dg* ($dg05, dg10$) and *Smart-any*. Results confirm that *Smart-any* cannot enforce a bounded partition size. We find a non-partitioned topology outcome (node density of $1/400$ $nodes/m^2$, size 400), as previously shown in Figures 5.18, 5.19, and 5.20.

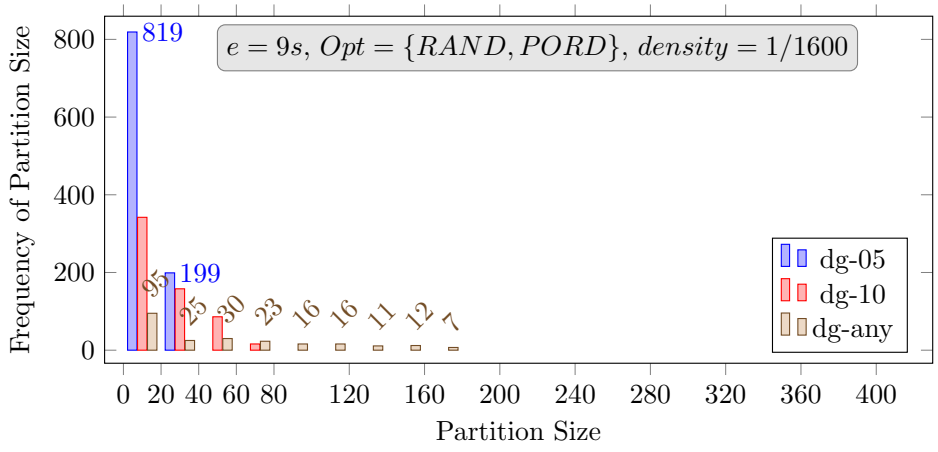


Figure 5.21: Frequency of WMN partition sizes. Bin width 20. Optimizations $RAND, PORD$, density $1/1600$ $nodes/m^2$.

Comparing the versions of *Smart-dg* ($dg05$, $dg10$), we realize that a more relaxed degree constraint (*e.g.*, dg = 10) resulted in fewer small partitions. We hypothesize that this result will facilitate the global WMN connectivity by the self-healing function, given the existence of fewer partitions.

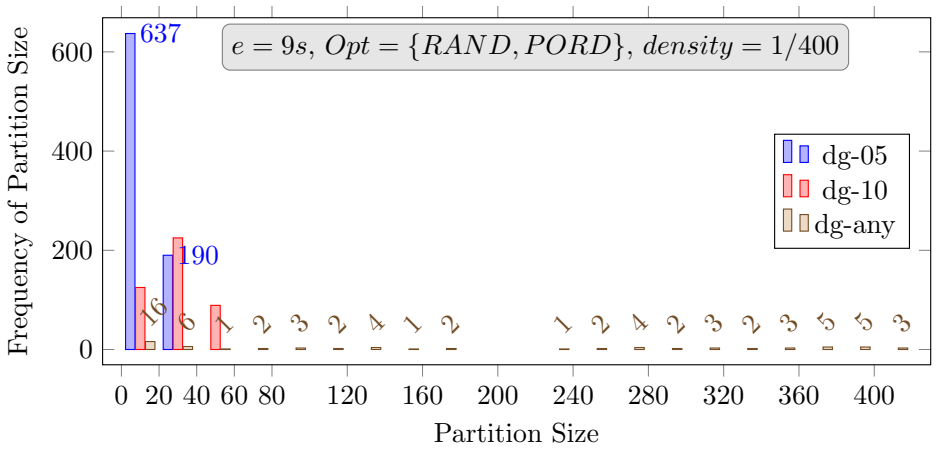


Figure 5.22: Frequency of WMN partition sizes. Bin width 20. Optimizations $RAND, PORD$, density $1/400$ $nodes/m^2$.

Therefore, there is a trade-off regarding the selection of max. degree for *Smart-dg*. A more constrained max. degree (*e.g.*, dg = 5) induces a lower control workload in the SDN paradigm (less nodes, reduced number of control events per flow). However, the topology structure with a larger number of small partitions requires a higher number of nodes with dual wireless interfaces for recovering global connectivity.

Chapter 6 Integrated Self-Organizing, Self-Healing WMN nodes

This chapter presents the design of agents that evolve a large node placement (a set of geographical positions) of wireless mesh nodes into a topology structure based on a set of interconnected partitions. The two main invariants of partitions are maintained, namely the bounding on partition diameter and bounding on the nodes' wireless interfaces degrees. The network formation is the result of a distributed algorithm executed by the agents in the WMN nodes, which independently make decisions that collectively lead to the final topology. Decisions are based on the agents' design composed of safety and liveness properties. The following sections present design principles for integrated self-organization and self-healing, the detailed design of our agents, and the outcomes of our experimental evaluations. Part of the content of this chapter was accepted for publication at NetSoft2020 [116].

6.1 Design principles for the integrated Self-Organizing, Self-Healing WMN nodes

The network formation occurs both at the physical and link layers. The physical isolation allows using orthogonal frequencies on different partitions. At the link-layer, the use of different network IDs (such as the BSSID - Basic Service Set ID) creates logical isolation to provide robustness to the case of equal frequency in neighboring partitions.

We expect no preliminary network design or manual configuration, characterizing a self-configuring network solution [12]. Under the addition and removal of nodes, this design must guarantee a set of properties critical to the applicability of SDN into WMNs: a self-organizing design [12]. Under broken reachability introduced by the partitioning actions, self-healing [12] agents must recover the global connectivity. A critical question in dynamic and autonomic network formation is the convergence to stable topology configurations and how the parameters of the self-organizing, self-healing agents, also self-configuring in nature, affect such convergence likelihood while guaranteeing the set of desired properties.

Agents materialize as software processes into mesh nodes, each agent controlling one wireless interface. It follows that two interfaces are required to run both organizing and healing functions on a node. However, the recovery of global connectivity demands only

a % of the total nodes running healing functions. We evaluate different healing designs, aiming at maximum connectivity with minimum healing agents.

We repurpose a minimum set of safety properties from Chapter 5 into a reference design: the *Smart-based* behavior. The first property is the WMN partition diameter property, which bounds the communication latency in a partition, a critical requirement for intra-group SDN-based network control planes. The autonomic aspect is fundamental to this solution, considering that the balanced graph partitioning is an NP-Hard problem [66] or NP-Complete under specific assumptions [67].

The second property of *Smart-based* is the control of the mesh node's wireless interface connectivity degree. This property bounds the number of events per new data flow handled by an SDN controller of partitions in the SDN paradigm. Transmissions of a node n_i in a WMN are received by all its neighbors, given the inherently broadcast nature of wireless communication. In a path of distance h for an average WMN node's interface degree dg, $ev \approx dg \cdot h$ events arrive at the controller for every new flow initiated. Therefore, the bounded interface degree limits the per-flow workload regarding network control events. Moreover, the degree control enforces density limitation on partitions that limits the contention and self-interference on WMNs [26, 27].

The combined bounding on diameter and interface degree limits the number of nodes per partition (the Degree/Diameter Graph problem [14]) regardless of any underlying node placement density, supporting the precise workload control in WMN SDN controllers. Finally, the self-organization design also solves the question of electing a mesh node to act as an SDN controller. Similarly to the leader election in distributed consensus protocols [110], partitions evolve from a unique origin node, a candidate to act as the partition controller.

The general operation of the agents involves cycles as described in the Section 4.1. The commands available to use in phase *Act* continue to be: *agent creates a mesh partition*, *agent joins a partition*, *agent leaves a partition*. Also, a possible decision is *maintain membership*, implying a null action. We comment on the phase *Read the Environment* of the agent operating cycle in the Sections 5.3 and 6.3.

6.2 Integrated autonomic behavior of agents

The following sections present the common behavior of our autonomic agents first to describe each agent's specificity later.

6.2.1 *Smart-based*: reference design of agents

The *Smart-based* reference agent design consists in the enforcement of two safety properties: the *partition diameter* bound and *maximum node interface degree*. Following, we provide preliminary definitions to define the two properties later formally.

Definition 6.1 (Valid nearby partitions)**.** *Valid nearby partitions: a* valid *neighboring partition has* i) *at least one node at communication reach of the deciding node,* ii) *all nodes not violating safety properties,* iii) *all nodes will not violate safety properties after the addition of the deciding node.*

Definition 6.2 (Partition origin nodes)**.** *Partition origin nodes: the* 'is origin' *attribute is* 'true' *on a* Smart-based *instance which creates a partition, and* 'false' *otherwise.*

Definition 6.3 (Potential neighbor node)**.** *Potential neighbor node: a node which will be a neighbor of a deciding node 'm' in a neighbor partition 'k'.*

Definition 6.4 (Partition diameter bound)**.** *Autonomic proxy for the partition diameter bound: to enforce the bound on the* partition diameter *using local information, a* Smart-based *agent enforces a distance to the origin node of a partition.*

Therefore, a nearby partition is valid to a *Smart-based* agent regarding the diameter bound property if the agent's hop-distance shortest path is at most h to the partition origin. It is trivial to verify that the diameter of the partition is, at most, $d = 2 \times h$: the origin node is a member of any shortest path from border to the origin, and the diameter unites two shortest paths from origin to the border.

More formally, let m be a *Smart-based* agent node reviewing its properties. Let P be the set of all partitions while $P_m \subseteq P$ is the set of all nearby partitions to m; let $k \in P_m$ be a nearby partition, o_k is the origin node of partition k. Let k_c be m's current partition if it is already connected, and $k_c \in P_m$.

Let $F_{distance\_sp}(i, j)$ be a function that returns the shortest-path distance between two nodes. Let $F_{DT} \rightarrow P_m$ be a function that eliminates nearby partitions k for which $F_{distance\_sp}(m, o_k) > h, \forall k \in P_m$. In practice, a *Smart-based* agent applies $F_{distance\_sp}()$ using local information by adding one hop to the distance of a potential neighbor on a different partition (adding 0 if $k = k_c$) and assuming the shortest-path to a partition k as the minimum distance amongst all potential neighbors in k. The origin node of any k has $h = 0$.

Definition 6.5 (Enforcing a node degree bound)**.** *Enforcing a node degree bound 'dg':* Smart-based *agents limit the set of valid nearby partitions in two steps: a) controlling*

their degree to at most 'dg', b) not inducing an invalid degree above 'dg' to their potential/actual neighbors.

More formally, let $F_{degree}(i)$ be a function that returns the degree of an wireless interface managed by an agent i; let $F_{DG} \rightarrow P_m$ be a function that removes partitions $k \in P_m$ if there exists any agent $i \in k$ such that i is in a potential neighbor to m and $F_{degree}(i) \geq dg, \forall i \in k, \forall k \in P_m$.

Let $F_{neighbors}(k, i)$ be a function that returns the number of future neighbors of i if it joins a neighboring partition $k \in P_m$; Finally, let $F_{NEIGH} \rightarrow P_m$ be a function that removes partitions $k \in P_m$ if $F_{neighbors}(k, m) > dg, \forall k \in P_m$.

6.2.2 *SmartOrg*: Self-Organizing agent design

The goal of *SmartOrg* agents is to produce a balanced partitioning, enforcing diameter and node degree bounds. *SmartOrg* agents inherit the safety properties of the *Smart-based* reference, and use the *partition size* liveness property to induce a balanced partitioning.

Definition 6.6 (Partition size liveness)**.** *Partition size liveness: agents aim at becoming a member of the largest* valid *neighboring partition, which is, at least, 'sp' percent larger (a threshold) than the agent's current partition.*

More formally, let $F_{size}(k)$ be a function that returns the size in number of nodes of a partition k, let s_m be the size of the current partition k_c of the deciding agent m, let $F_{ES} \rightarrow P_m$ be a function that *i)* eliminates partitions $k \in P_m$ for which $F_{size}(k) < (1 + sp) \times s_m$, and *ii)* sorts P_m by $F_{size}(k), \forall k \in P_m$ into the ordered list $P_m^{'}$.

If $P_m^{'} \neq \varnothing$, the first item $P_m^{'}[0] = k_b$ is the best partition membership option regarding size. If $P_m^{'} = \varnothing$ and $k_c \in P_m$ (current partition k_c checked as *valid* regarding safety properties), we add back k_c into $P_m^{'}$ as an option (stay on current partition).

SmartOrg agents only admit other *SmartOrg* agents when assessing the state of their safety properties. Therefore, *SmartHeal* agents do not induce degree increase or partition diameter increase in the autonomic functions of *SmartOrg*. Section 6.4 provides support for this design decision.

Finally, on the evaluation phase of its agent cycle, SOrg executes its inherited autonomic functions: $F_{DT} \rightarrow P_m$, $F_{DG} \rightarrow P_m$, $F_{NEIGH} \rightarrow P_m$, and its specific function $P_m^{'} = F_{ES} \rightarrow P_m$.

Let A_S be the decision of *SmartOrg* as a set of actions:

$$A_S = \begin{cases} \text{If } P_m^{'} = \varnothing & : \{create\_partition\} \\ \text{Else if } P_m^{'} = \{k_c\} & : \varnothing \quad \text{(a null action set)} \\ \text{Otherwise} & : \{leave\_partition(k_c), \\ & \quad join\_partition(k_b)\} \end{cases} \tag{6.1}$$

6.2.3 *SmartHeal*: Self-Healing agent design

The goal of *SmartHeal* agents is to interconnect partitions in order to recover maximum possible global connectivity while respecting the bounds on partition diameter and node degree. *SmartHeal* agents inherit the safety properties of *Smart-based* reference; however, we admit different degree bounds dgh as variations for *SmartHeal*. *SmartHeal* agents also admit two different forms of liveness: *partition size* and *signal strength*. Moreover, *SmartHeal* agents have an additional safety property to induce increased connectivity: *different membership of companion* SmartOrg *agents*.

Definition 6.7 (Companion agents)**.** *Companion agents' partitions: the partition set of agents residing at the same mesh node.*

More formally, let $C(m)$ be a function returning the set of current partitions of agents in the deciding node m. Let $F_{CO} \rightarrow P_m$ be a function that removes partitions $k \in P_m$ if $k \in C(m)$.

Definition 6.8 (Signal strength liveness)**.** *Signal strength liveness: agents aim at becoming a member of the closest* valid *neighboring partition by finding partitions with a potential node showing signal strength 'st' percent higher (a threshold) than the signal of the closest neighbor in the agent's current partition.*

More formally, let $ST(k)$ be a function the returns the maximum signal strength of all potential neighbors of a partition k, let $st_m = ST(k_c)$ (m's current max signal strength), let $F_{ST} \rightarrow P_m$ be a function that *i)* eliminates partitions $k \in P_m$ for which $ST(k) < (1 + st) \times st_m$, and *ii)* sorts P_m by $ST(k), \forall k \in P_m$ into the ordered list $P_m^{'}$.

SmartHeal agents only admit *SmartOrg* agents when accounting their diameter bound safety property; however, *SmartHeal* account both other *SmartHeal* and any *SmartOrg* agents on their node degree bound property. Therefore, *SmartHeal* agents never extend a partition from another *SmartHeal* and respect node degree bounds. Section 6.4 provides support for these design decisions.

Finally, on the *eval* phase of its agent cycle, *SmartHeal* executes its inherited autonomic functions: $F_{DT} \rightarrow P_m$, $F_{DG} \rightarrow P_m$, $F_{NEIGH} \rightarrow P_m$, and its specific functions $F_{CO} \rightarrow P_m$, $P_m^{'} = F_{ES} \rightarrow P_m$ or $P_m^{'} = F_{ST} \rightarrow P_m$ (the active liveness).

Let A_S be the decision of *SmartHeal* as a set of actions:

$$A_S = \begin{cases} \text{If } P_m^{'} = \varnothing & : \varnothing \quad \text{(a null action set)} \\ \text{Else if } P_m^{'} = \{k_c\} & : \varnothing \quad \text{(a null action set)} \\ \text{Otherwise} & : \{leave\_partition(k_c), \\ & \quad join\_partition(k_b)\} \end{cases} \tag{6.2}$$

6.3 Agent information

We assume the same principles of Chapter 5 regarding information gathering and sharing: the collection of local (internal and neighbors) information, sharing based on broadcasts at a much smaller period then collection for complete information reading with high probability. Reading information based on wireless scans that do not require network connectivity or synchronous communication, configuring a mechanism robust to failures, and highly scalable. Agents share the bounds on their current partition - SmartOrg *interface degree*, SmartHeal *interface degree*, *distance to the partition origin bound* - and other dynamic or agent-specific attributes - *agent type*, *distance to the partition origin*, *interface degree*, *partition size*. Other data exist as shared attributes of wireless standards such as *link-layer network ID* (SSID, BSSID in IEEE 802.11), *physical layer's signal strength*. The per-frequency scanning process provides the *wireless frequency* attribute.

Table 6.1 consolidates the information that characterizes the *state* of an agent, obtained in the phase *Read Environment* of the agent operational cycle described in Section 4.1. The items flagged in the "SP" column are the system parameters.

The list below provides additional information about the attributes that compose the *state* of an agent. Any system parameter (SP) attribute is an initial input configured by the designer of the autonomic WMN. They are stored and retrieved from non-volatile memory. The other attributes are volatile.

1. *SmartOrg* interface degree bound: retrieved from the system parameters by agents on the origin node, shared by the *SmartOrg* agent on the origin node to the other agents in the partition. All agents on the partition re-share the per partition attributes. Defines the maximum number of direct neighbors (graph degree) that the interface controlled by a *SmartOrg* agent can have.

Agents' *State*			
SP	Attribute	Source	Type
X	*SmartOrg* interface degree bound	Set by origin node, re-shared by agents	Per partition
X	*SmartHeal* interface degree bound	Set by origin node, re-shared by agents	Per partition
X	Distance to partition origin bound	Set by origin node, re-shared by agents	Per partition
X	*SmartHeal* liveness	Set by origin node, re-shared by agents	Per partition
	Partition size in nodes	Routing layer, shared by agents	Per partition
	Agent type	Agent shared attributed	Per agent
	Agent distance to the partition origin	Agent shared attributed	Per agent, dynamic
	Agent interface degree	Agent shared attributed	Per agent, dynamic
	Link-layer network ID	Wireless standard attribute	Per partition
	Signal level strength (RSSI)	Wireless standard attribute	Per neighbor, dynamic
	Wireless frequency (channel number)	Scanning process	Per partition

Table 6.1: The *state* of autonomic agents.

2. *SmartHeal* interface degree bound: same source and sharing as the *SmartOrg* interface degree bound (SP), applied to limit the numbers of neighbors of the *SmartHeal* agents in the partition.
3. Distance to partition origin bound: same source and sharing as the *SmartOrg* interface degree bound (SP). Limits the maximum number of hops between an agent (*SmartOrg* and *SmartHeal*) and the agents in the origin node.
4. *SmartHeal* liveness: same source and sharing as the *SmartOrg* interface degree bound (SP). Defines the type of liveness used by the partition: partition size (ps) or signal strength (ss).
5. Partition size (in the number of nodes in the partition): dynamic information extracted by the agent from the routing layer, and shared as part of its state. Agent type: a fixed attribute of the agent and shared as part of its state. Defines the agent as *SmartOrg* or *SmartHeal*.
6. Agent distance to the partition origin: dynamic information describing the number

of hops from the agent to the origin node. Attribute shared by the agent as part of its state.

7. Agent interface degree: the agent's perception of its number of direct neighbors. Attribute shared as part of its state.
8. Link-layer network ID: identification of the link-layer mesh network that represents the partition. Attribute shared by the wireless technology as part of its link-layer information set (in IEEE 802.11 - WiFi, this is SSID, BSSID).
9. Signal strength: the signal level that a given agent perceives of each of its direct neighbors. Information extracted from the wireless interface physical layer (wireless radio system). In IEEE 802.11, this is known as the RSSI. No sharing.
10. Wireless frequency (or channel number): the operational frequency used by the agent or any of its neighbors. For the neighbors, the scanning process determines this information. The agent knows its operating frequency inside a partition by reading information from its physical layer. No sharing.

When reading the environment, agents scan the frequency of their current partition first to improve the efficiency of the *PORD* optimization, discussed on Section 6.4.2.

To obtain the experimentation results presented in this chapter, the autonomic agents rely on a simulated agent platform (Chapter 3), using the module ASim to obtain their information and operate under concurrent settings. Section 6.4 presents an extensive analysis of the convergence of the integrated operation of *SmartOrg* and *SmartHeal* under concurrent settings supported by ASim.

6.4 Convergence of the *Smart-based* agents

Chapter 5 describes conditions that lead agents to safety violation states either when joining or leaving partitions under concurrent operation. Our integrated design of SOrg and SHeal agents stem from two common safety properties, which are the same that based the analysis Chapter 5: bounds on the partition diameter and wireless interface degree. In Chapter 5, nodes had a single interface, thus the "node degree." Here we assume multiple interfaces on nodes, hence the need for a per-interface degree control. Therefore, the conditions previously described are yet valid here. However, the integrated operation of agents with distinct objectives brought new challenges to convergence.

In this section, we comment on new divergence scenarios on the integrated operation of *Smart-based* agents and the application of optimizations on the timing of the operational cycle of agents for improved convergence in time and effort.

6.4.1 New divergence scenarios

The cases described below represent conditions that occurred in experiments, later solved and verified experimentally.

The first case stems from the hypothesis that differing degree bounds on *SmartOrg* (dg) and *SmartHeal* (dgh) could improve the efficiency of connectivity recovery by *SmartHeal* (details on the hypothesis and its evaluation on Section 6.6.3). In doing so, we found the following divergence case.

Lemma 6.1 (Agents with different interface degree)**.** *Agents with different interface degree bounds can lead to divergence.*

Proof. Assume a partition p_1 on its dg bound limit. Assume a disconnected *SmartHeal* agent sh_1 with its degree bound $dgh = dg + 1$. Assume a *SmartOrg* agent $so_2 \in p_1$ which is at the reach of sh_1, turning p_1 a nearby partition to sh_1. Given the degree bounds of sh_1, it bridges to p_1, inducing a degree violation in so_2. When so_2 leaves p_1 to solve its violation, it turns p_1 out of reach of sh_1, causing sh_1 to be in distance violation, and also leaving p_1. Assuming that so_2 had p_1 as its best membership option, so_2 joins p_1 again after its degree reduction when sh_1 left it. Therefore, so_2 joins p_1, configuring the starting condition that will lead to divergence. □

To solve the above divergence scenario, *SmartOrg* agents do not consider *SmartHeal* agents to determine valid nearby partitions: they do not extend a partition from *SmartHeal* agents, nor assume that *SmartHeal* agents induce an increase on their degree. Therefore, *SmartHeal* becomes a secondary agent type that reacts to the changes induced by *SmartOrg*.

Lemma 6.2 (Divergence by signal strength liveness)**.** *The signal strength liveness on the* SmartHeal *agents can lead to divergence.*

Proof. Assume two nearby *SmartHeal* agents sh_1 and sh_2, respectively in partitions p_1 and p_2. sh_1 detects p_2 as its best liveness given the proximity of sh_2 which induces its highest perceived signal strength. sh_1 changes its membership to p_2. Concurrently, sh_2 evaluates p_1 as its best liveness given the proximity of sh_1, changing its membership to p_1. These cases represented no violation, the *RAND* optimization was not activated. Therefore, the scenario repeats in next epochs. □

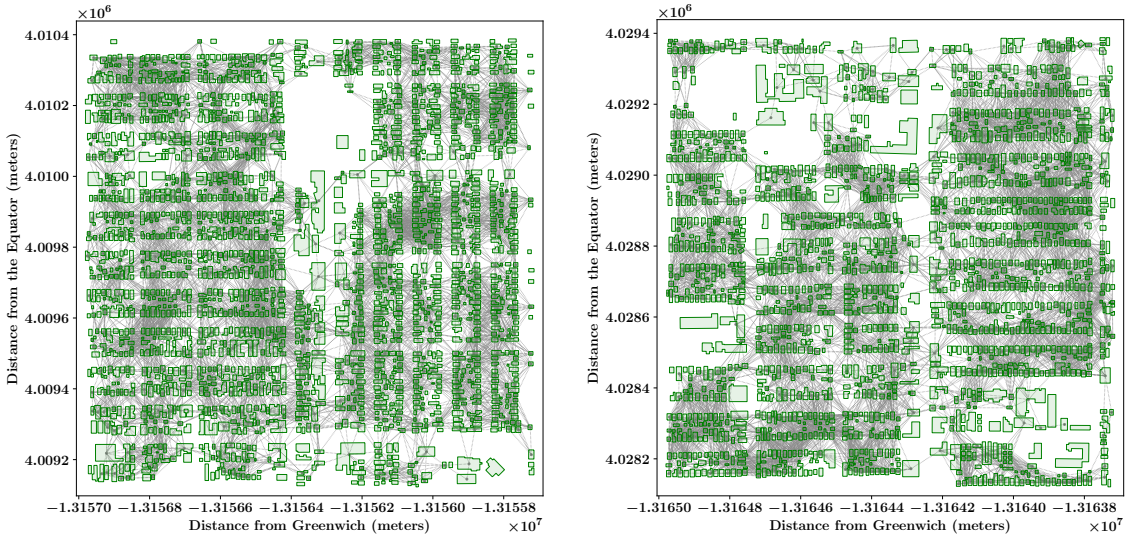
The specific case described in Lemma 6.2 generalizes to any liveness that uses abruptly varying information such as the *signal strength*. The *partition size* liveness has a smooth

variation, not triggering this condition. The chosen solution was to activate *RAND* anytime an agent leaves a partition, let it be by a safety violation, liveness improvement, or the segmentation on a partition that forces an agent out.

6.4.2 Pseudo-orderings for convergence

We maintain the two optimizations proposed in Chapter 5, which enforce convergence (*RAND*) and reduce the effort to convergence regarding the number of partition changes (*PORD*). *RAND* changes the concurrency set of an agent (neighbor agents in their *REA* phases) by introducing random delays to the start of the next *REA*. *PORD* enforces a partial ordering in which agents closer to the border of partitions act earlier to solve violations.

6.5 Visualizing the outcome of the behavior of the integrated agents

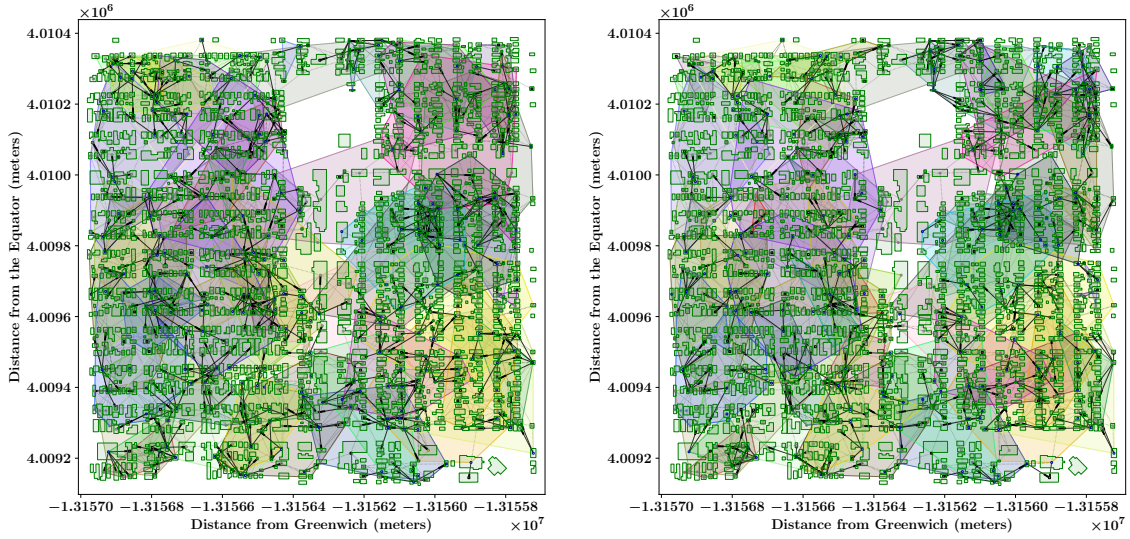


(a) Long Beach, CA, U.S. (b) Los Angeles, CA, U.S.

Figure 6.1: Underlying maximum possible connectivity of 1000 nodes on a $\approx 1.44\ Km^2$ region in LA County, CA. Shows the maximum set of neighboring options (or a single partition solution). Wireless standard IEEE 802.11a. Density of $1/1600\ node/m^2$.

Figure 6.1 represents the maximum *underlying* connectivity if all nodes operate on the same frequency in the physical layer, the same logical network (SSID, BSSID) at the link-layer, using their nominal transmit power. Therefore, no additional connectivity can

exist. The node placements are based on the Microsoft U.S. buildings data set [98].



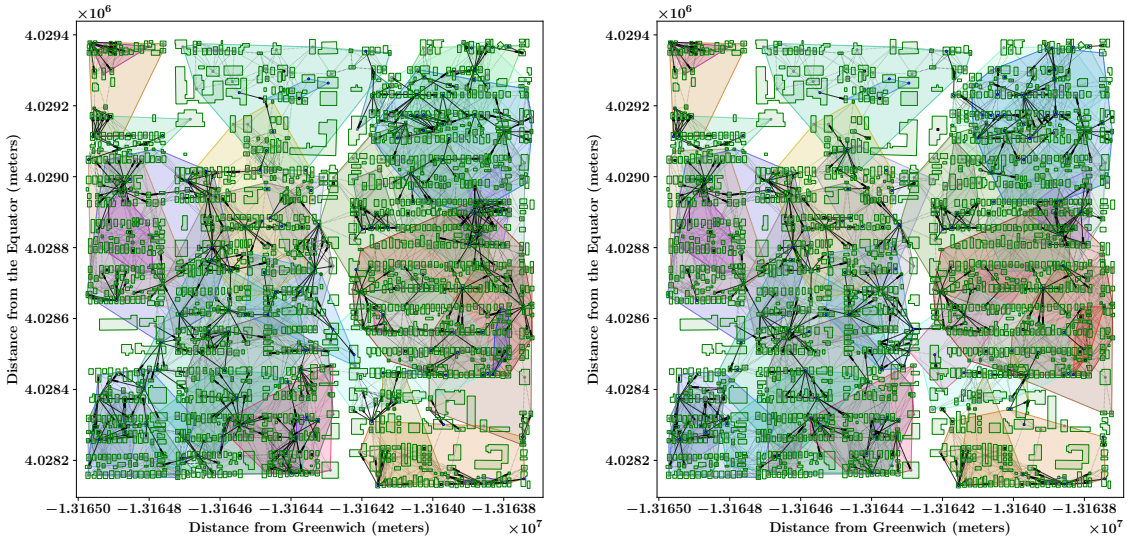
(a) Liveness *partition size*, 100% connected (b) Liveness *signal strength*, 99.8% connected

Figure 6.2: Autonomically created WMN topology partitioned by *SmartOrg* and connected by *SmartHeal* agents. Overlapping partitions evidenced by the partitions' convex-hull. Dense edges show inter-partition connectivity by *SmartHeal* agents on 20% of nodes. Same wireless standard and node placement of Figure 6.1. Maximum interface degree $\{5, 6\}$, maximum partition diameter 6. Region of Long Beach in LA County.

More formally: let $G(V, E, P)$ be a graph representing the connectivity topology of a WMN where V is the set of nodes, E is the set of edges, P the set of positions of the nodes in V, n is a node such that $n \in V$, f_n, t_n are frequency and network id of n; E is maximal if $\forall i, j \in V, f_i = f_j, t_i = t_j$.

The choices of the autonomic agents on the WMN nodes define the partitioned WMN topology and inter-connections (Figures 6.2 and 6.3). The node placement of Figure 6.1a is the same basis for the topologies obtained in the Figures in 6.2 while of the topologies in Figures 6.3 shared the same node placement with Figure 6.1b. Partitions in Figure 6.2 and 6.3 are degree/diameter constrained.

More formally: let $G_{6.1}(V_{6.1}, E_{6.1}, P_{6.1})$, $G_{6.2}(V_{6.2}, E_{6.2}, P_{6.2})$ be graphs representing the topologies of Figures 6.1, 6.2, respectively; $V_{6.1} = V_{6.2}$, $P_{6.1} = P_{6.2}$, $E_{6.1} \supseteq E_{6.2}$. Finally, topologies in $E_{6.1}$ are maximal.



(a) Liveness *partition size*, 100% connected (b) Liveness *signal strength*, 100% connected

Figure 6.3: Autonomically created WMN topology partitioned by *SmartOrg* and connected by *SmartHeal* agents. Overlapping partitions evidenced by the partitions' convex-hull. Dense edges show inter-partition connectivity by *SmartHeal* agents on 15% of nodes. Same wireless standard and node placement of Figure 6.1. Maximum interface degree $\{10, 11\}$, maximum partition diameter 6. Region of Long Beach in LA County.

6.6 Results

This section presents the experimentation settings, and the results of the integrated operations of agents. We apply the experimentation platform used in 3 to support agent information and decision-making.

6.6.1 Experimentation settings

The following list describes node placements and settings of the wireless subsystem.

1. Nodes have positions based on the centroids of buildings from [98]. We used a common set of 30 different *regions* within LA County, CA, creating 30 node placements.
2. We control the average node placement density over the experimentation area to enforce the node densities pointed in the results.
3. The nodes' wireless interfaces use IEEE 802.11a physical layer (5 GHz band) with a 20 MHz bandwidth. The tx power is 16 dBm, the gain of the tx/rx antennas is 1 dBi (all defaults). The CCA (Clear Channel Assessment[1]) threshold is -99 dBm.

[1] Identifies the channel as free for transmission.

The Energy Detection Threshold[2] is -96 dBm. The last two also default values.

4. The link-layer of the mesh nodes uses the IBSS (Independent Basic Service Set) mode of IEEE 802.11, creating WMNs through multi-point association of nodes at the link-layer.

Epochs have a period of 90 seconds with 9 sec for agent decision. The plots show results for 8 configurations of *SmartOrg* and *SmartHeal* which are a combination of the liveness property used by *SmartHeal* in $lv = \{ss, ps\}$ (wireless *signal strength*, *partition size*, respectively) with the degree constraint assumed by *SmartOrg* in $dg = \{5, 10\}$ and *SmartHeal* in $dgh = \{5, 6, 10, 11\}$. A fixed node placement (NP) density of $625\ nodes/km^2$ was assumed. Each result combines experiments using 30 different NPs in Los Angeles County (CA) based on the location of centroids of buildings from [98], and each NP with a total of 1000 nodes. Each experiment had a maximum time of 30 epochs. *SmartOrg* applied the combined *RAND, PORD* optimizations while *SmartHeal* applied *RAND*. Finally, we vary the percentage of the total of nodes with two wireless interfaces from 0% to 50% in 5% steps, executing the *SmartHeal* agent in the nodes with a second interface.

At least 90% of nodes are spawned in the first epoch to represent the extreme case of 90% node churn on the return of a power outage. The sp threshold of the *partition size (ps)* liveness property is 0.2 for agents in the origin node and 0.1 for agents in any other node, while the st threshold of the *signal strength (ss)* liveness is 0.1 for agents on any node. The evaluation of the appropriate ranges of values for sp, st will be the subject of future work.

6.6.2 Time to convergence

We simulated the integrated operation of *SmartOrg* and *SmartHeal* enabling *REA* concurrency, a real distributed operation scenario in which agents are subject to decision-making using outdated information. We recall that the *Smart-any* agent version which does not enforce the degree bound does not produce a balanced partitioning, not fulfilling the liveness objective of *Smart-based* agents as described on Section on 5.6.

Using Figure 6.4, first we verify that the set of configurations with degree bounds in $dgh \in \{5, 6\}$ (*05-05, *05-06) demand more epochs to converge than the ones with $dgh \in \{10, 11\}$, explained by the increased competition for space in smaller partitions, causing more violations to recover from. Also, within each of the two groups of degree enforcement above, the liveness ps incurs slower convergence than ss. The explanation is that ps induce agents to compete for space in larger partitions, closer to their limits, again causing more

[2] Triggers the start of packet reception.

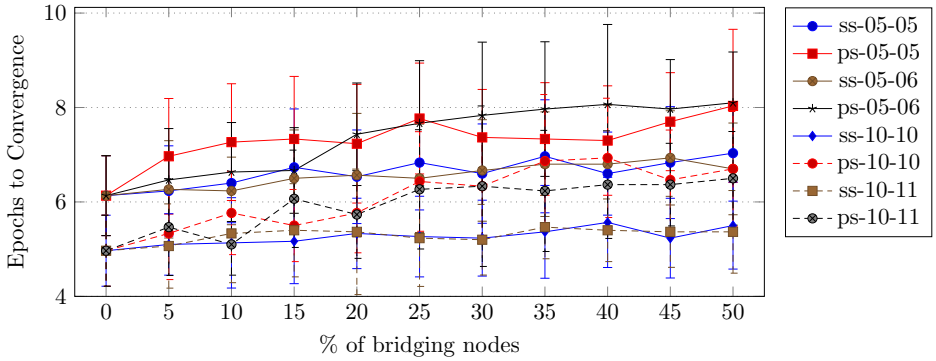


Figure 6.4: All agent configurations converge within ten epochs. The ones with liveness ps on *SmartHeal* and more restricted partitions ($dg \in \{5, 6\}$) incur longer convergence time.

violations that require further changes in partition membership. Finally, the configurations with a differing degree bound between *SmartOrg* and *SmartHeal* (*05-06, *10-11) did not impact the convergence time. However, following we will see that increasing the degree limit by one improves *SmartHeal*'s liveness: recovering global connectivity.

6.6.3 Recovering global connectivity

In this subsection, we evaluate the ability of the different SHeal agent configurations described in Section 6.6.1 to recover the global connectivity of the partitioned WMN: the goal of the *SmartHeal* agents. Figure 6.5 shows, in the Y-axis, the percentage of nodes in the largest WMN segment compared to the total of nodes. A result of 100% means a connected WMN outcome when any node can reach any other node. We vary the percentage of *SmartHeal* agents in the X-axis to compare the efficiency of the agent configurations regarding the recovery of global connectivity.

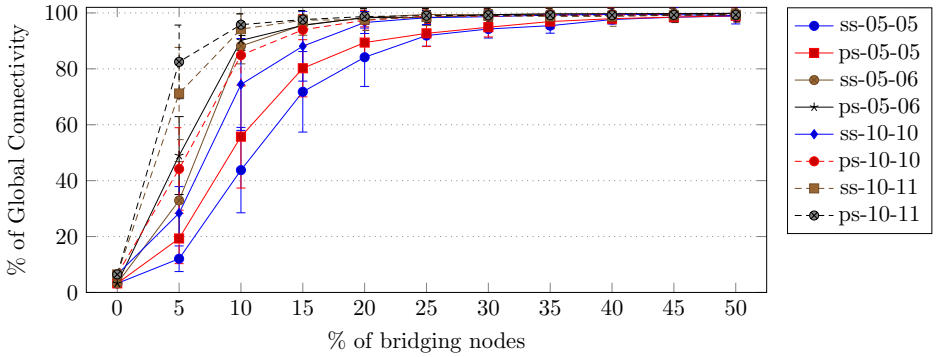


Figure 6.5: The efficiency of agent configurations to recover global connectivity, given the % of nodes with *SmartHeal* agents. A distinct degree bound improves connectivity. Relaxed degree bounds ($dgh \in \{10, 11\}$) also induce improved connectivity results.

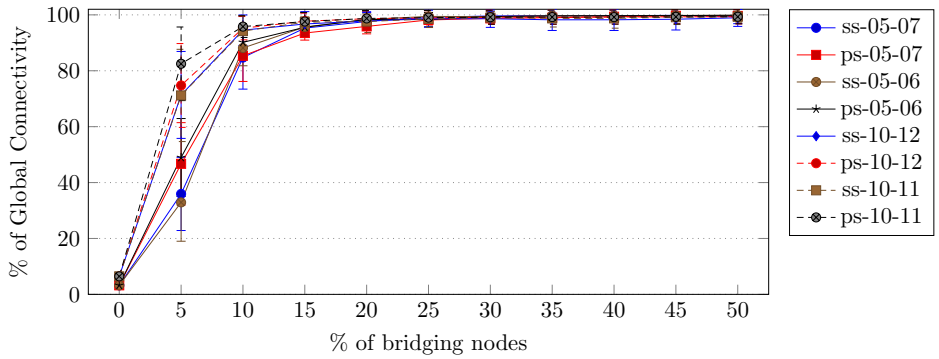


Figure 6.6: Comparing recovering global connectivity with $+1$ and $+2$ *SmartHeal* agent degree limits. As hypothesized, the $+1$ limit outperforms $+2$ due to performing a better distribution of inter-partition connections.

Figure 6.5 ascertains that *i)* a higher degree bound, *ii)* the $+1$ degree bound on *SmartHeal*, *iii)* and the *partition size* liveness on *SmartHeal* induce better connectivity at low percentage of healing agents. The higher degree bound (dg = 10, $dgh \in \{10, 11\}$), for a fixed number of nodes, produces a smaller number of large partitions, requiring less inter-partition connectivity. Chapter 5 anticipated this effect when analysing SOrg agents in isolation. The *ps* liveness induces the *SmartHeal* agents to be members of the largest possible partitions which also induces a smaller number of partitions to inter-connect.

The $+1$ degree limit on *SmartHeal* induces a better global choice of partition connectivity without global coordination. In effect, the $+1$ difference on the degree limit of *SmartHeal* creates a *surface* of partition membership that is exclusive to healing agents. *SmartOrg* agents keep partitions on their size limit due to their *ps* liveness oriented to membership to the biggest nearby partition, leaving the $+1$ surface as the available option for *SmartHeal*. Moreover, two nearby healing agents on a given partition p_1 cannot both bridge to a nearby partition p_2 using such exclusive surface, forcing them to choose different partition connectivity options. This design spreads the connectivity of healing agents through different nearby partitions, without relying on global coordination.

The Lemma 6.3 below reasons about why $+1$, and not $+2$ or more, achieves better global connectivity.

Lemma 6.3 (Degree limit on *SmartHeal* $+1$)**.** *The $+1$ degree limit on* SmartHeal *will outperform the $+2$ limit.*

Proof. Assume two nearby nodes n_1, n_2 which are members of the same partition p_1. Assume two nearby partitions p_2, p_3, reachable by n_1, n_2, operating on their organizing degree limit. Healing agents on n_1, n_2 will inter-connect to different partitions, given the $+1$ degree availability. If the healing degree limit was $+2$ or more, the outcome of both

healing agents on n_1, n_2 bridging to the same nearby partition becomes possible which renders a worse connectivity. □

Figure 6.6 compares the connectivity recovery of $+1$ and $+2$ *SmartHeal* agents. The $+1$ configuration with the ps liveness continues to be the highest performer, outperforming the new $+2$ configuration with the ps liveness (use the 5% data point for verification).

Finally, we observe that the *SmartHeal* configurations with liveness *ss* are less efficient on connectivity recovery due to inducing worse balancing of distribution of partition sizes (evaluated on Section 6.6.6).

This paper does not capture the possible benefits of the liveness *Signal Strength* (*ss*) regarding the capacity increase and the resilience to interference. *ss* induces healing agents to connect to closer partitions, turning wireless links less vulnerable to variations in attenuation (such as rainy weather or obstacles passing through the link's path), and also forming higher-speed links in variable speed wireless such as the IEEE 802.11 family. Moreover, in the sub-section 6.6.2, we showed that *ss* induces faster convergence. We expect improved WMN capacity from the *ss* liveness (higher speed/reliability links) at the cost of more healing agents to recover global connectivity.

6.6.4 Converging to defined properties

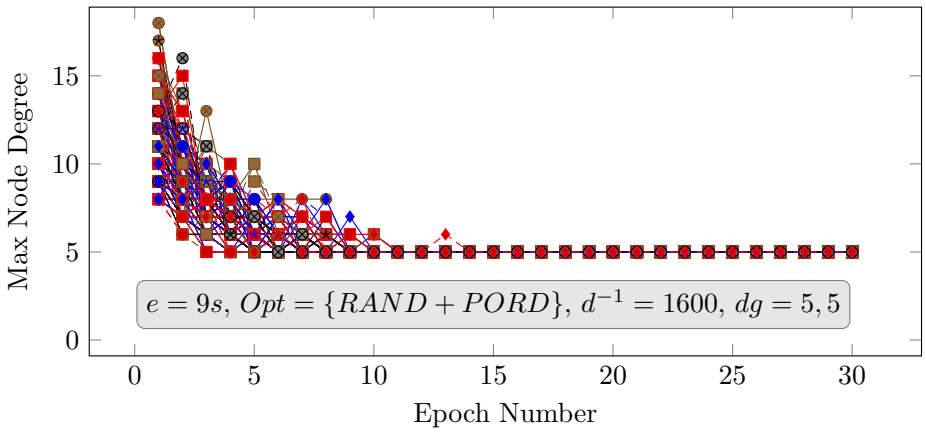


Figure 6.7: The maximum achieved node interface degree of *SmartOrg* and *SmartHeal* agents for liveness *ps*, showing a single convergence trend given the same degree bound on both agent types. Converged to dg = dgh = 5.

This subsection evaluates whether safety property goals are met under the integrated and competitive operation of *SmartOrg* and *SmartHeal*. Figures present the maximum value observed for a safety property, while operating on 30 different node placements, for a given configuration of degree limit of *SmartOrg*, degree limit of *SmartHeal*, and

SmartHeal liveness. The figures also accumulate the results of any % of healing agents from 0% to 50% in 5% steps as described in the Section 6.6.1, a total of 330 experiments per figure.

Figures 6.7, 6.8, 6.9 and 6.10 present the evolution of the maximum node degree in the experiment set over time (in epochs) while Figures 6.11, 6.12 and 6.13 present the evolution of the maximum partition diameter over time (other diameter settings omitted for space). The results show that the initially erratic control of the safety properties later converged to the defined objectives. When having 0% of healing agents, the degree upper bound reduces to the organizing degree bound. For any other percentage of healing agents, the $+1$ healing degree bound becomes the upper bound in the results as seen on Figures 6.8. Figures 6.8, 6.9 and 6.10 present double lines that represent these two upper bounds. Results for $dgh \in \{10, 11\}$ are similar, omitted in the interest of space.

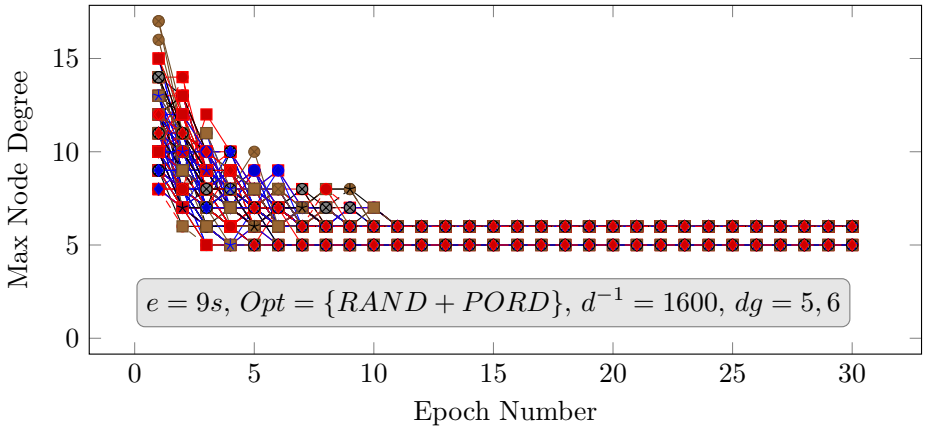


Figure 6.8: The maximum achieved node interface degree of *SmartOrg* and *SmartHeal* agents for liveness *ps*, showing double convergence trends given the different degree bounds on agent types: dg = 5, dgh = 6.

6.6.5 Effort to convergence

Figures 6.14, 6.15, and 6.16 respectively describe the effort to convergence in agent moves between partitions of the integrated operation of *SmartOrg* and *SmartHeal*, the *SmartOrg* agents effort, and the *SmartHeal* agents effort.

Figure 6.14 shows a linear increase of effort (moves) with the increase of *SmartHeal* agents. The increased effort stems from the increase in effort of the *SmartHeal* agents, as seen in Figure 6.16. Figure 6.15 shows that *SmartOrg* agents do not increase their effort with an increasing number of *SmartHeal* agents. The secondary nature of *SmartHeal* explains the latter: *SmartOrg* agents disregard the effect of *SmartHeal* agents on their properties, requiring *SmartHeal* to react to organization actions but not the opposite.

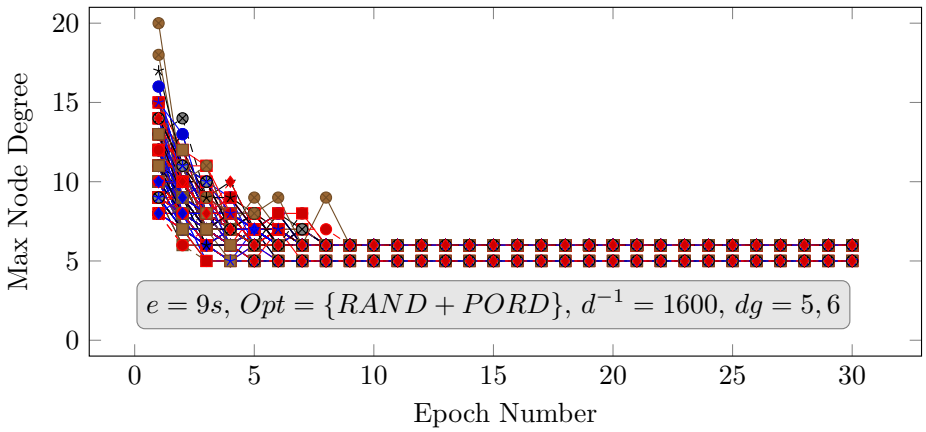


Figure 6.9: The maximum achieved node interface degree of *SmartOrg* and *SmartHeal* agents, also showing convergence to the defined properties for the liveness *ss*.

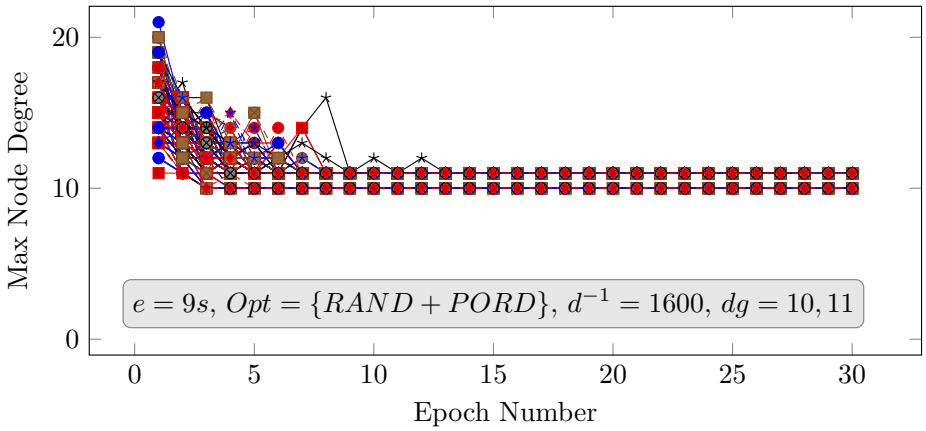


Figure 6.10: The maximum node degree of *Smart-dg* and *SmartHeal* agents for liveness *ps*, showing double convergence trends given the different max degree on agent types: dg = 10, dgh = 11.

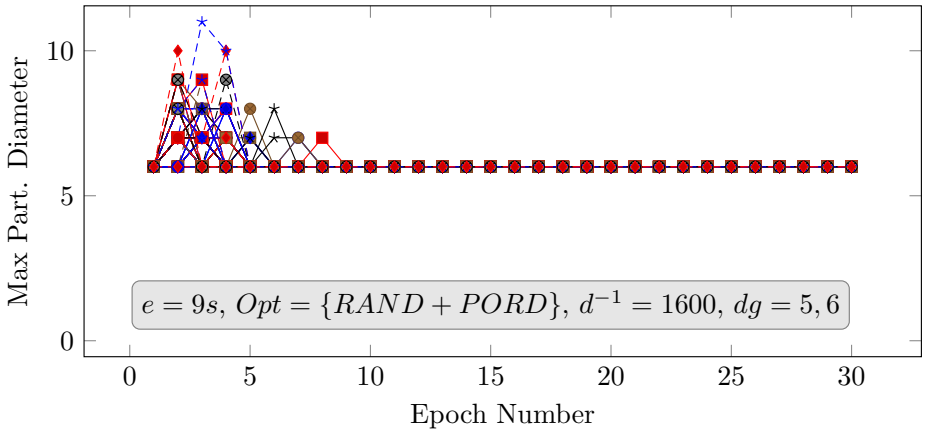


Figure 6.11: The maximum partition diameter for liveness *ps*, showing convergence to objectives for dg = 5, dgh = 6.

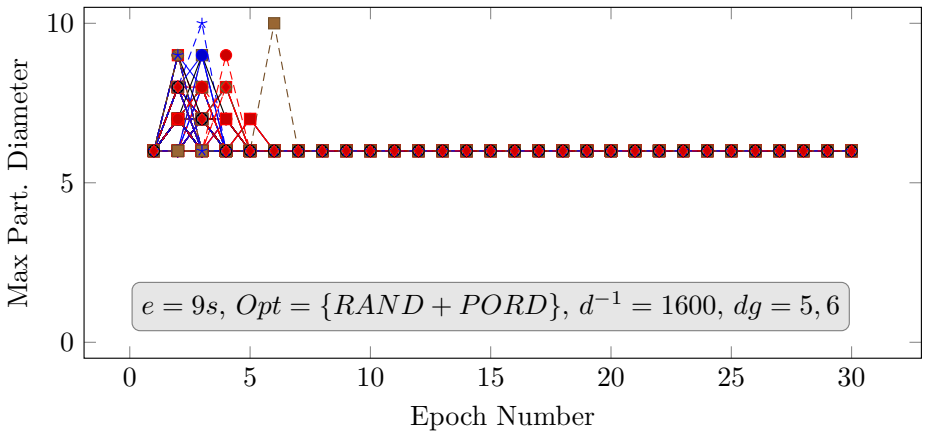


Figure 6.12: The maximum partition diameter for liveness *ss*, also showing convergence to objectives for dg = 5, dgh = 6.

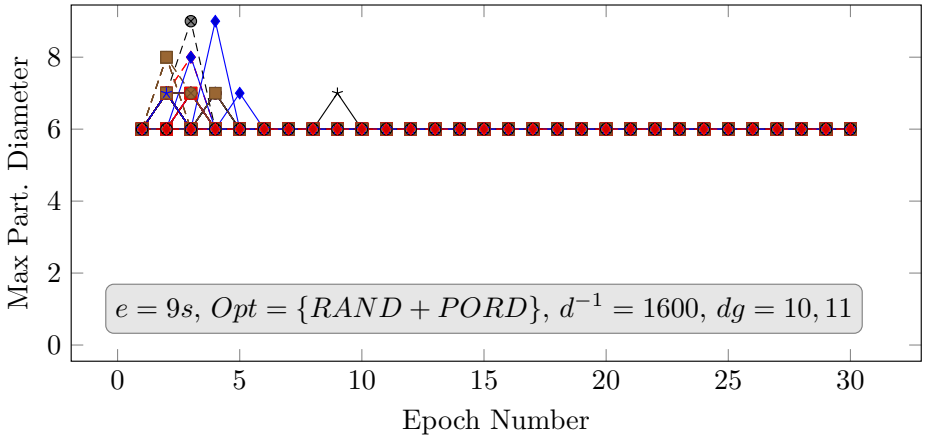


Figure 6.13: The maximum partition diameter for liveness *ps*, showing convergence to objectives for dg = 10, dgh = 11.

When comparing the different agent configurations, we perceive that the more constraining degrees (dgh = $\{5, 6\}$) demand higher effort to convergence (Figure 6.15). Also, the *ps* liveness on *SmartHeal* induces a trend of higher effort to both *SmartOrg* and *SmartHeal* when compared to the *ss* liveness, although a small increase in effort. These results are consistent with the findings in the time to convergence (Section 6.6.2).

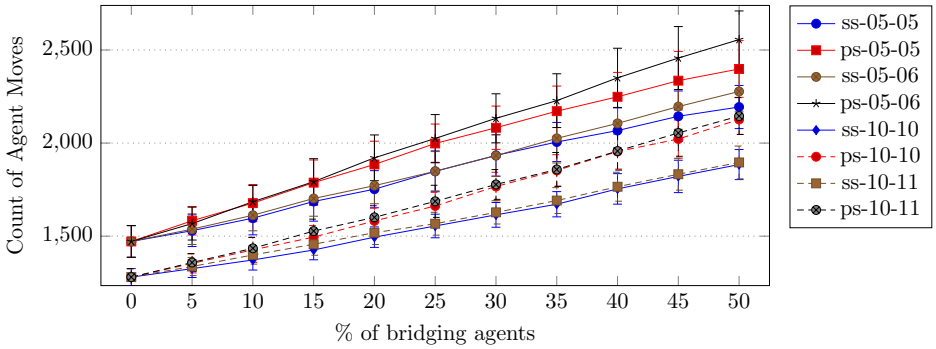


Figure 6.14: Effort to convergence in agent moves between partitions. Shows moves from both *SmartOrg* and *SmartHeal* for the agent configurations described in Section 6.6.1. A linear increase of effort with the increase of *SmartHeal* agents.

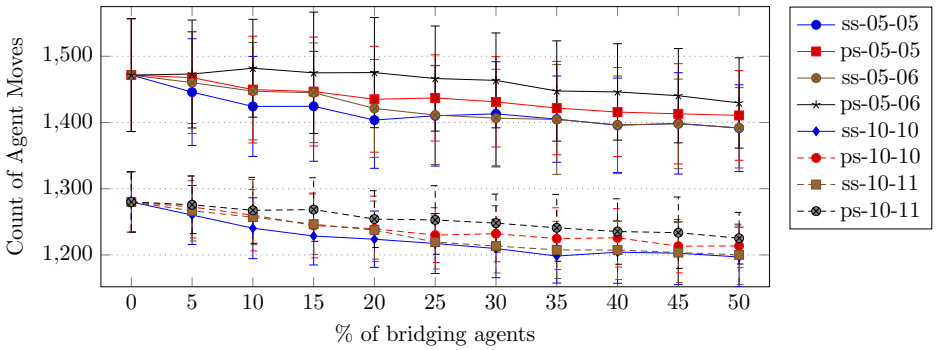


Figure 6.15: Effort to convergence of *SmartOrg* agents in moves between partitions. Same configurations of Section 6.6.1. A small decrease of effort with the increase of *SmartHeal* agents.

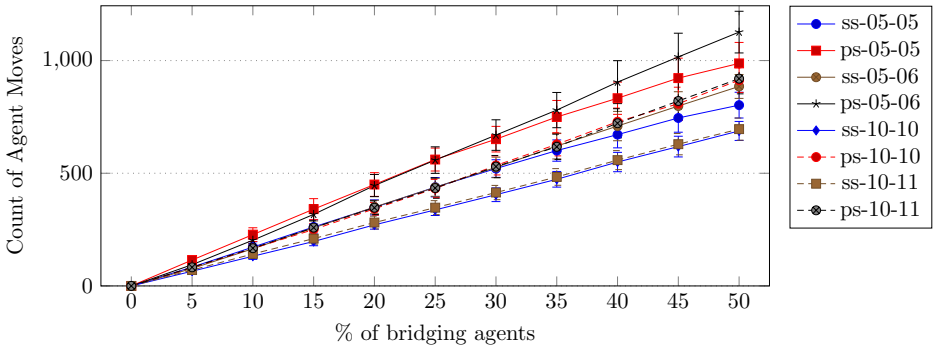


Figure 6.16: The effort to convergence in agent moves between partitions. Shows moves of the *SmartHeal* agents for the configurations described in Section 6.6.1: a linear increase of effort with the increase of *SmartHeal* agents.

Finally, from Figure 6.16 we observe that *SmartHeal* has an average number of moves $m > 1$ (50% of *SmartHeal* agents implies 500 agents given that experiments used 1000 nodes; also, there are at least 700 and up to 1100 moves for the 50% data point). The

reactive nature of *SmartHeal* agents, which must not be in the same partition of companion *SmartOrg* agents, explains this result: beyond enforcing its safety and improving its liveness, *SmartHeal* reacts to decisions of the *SmartOrg* agents in the same mesh node.

6.6.6 Topology structure under integrated organizing and healing

This section analyzes the resulting WMN topology structure. We observe the differences in the balancing of partition sizes induced by the two livenesses options evaluated for *SmartHeal*. We recall that the combined bounding on node degree and partition diameter induces an upper limit on the size of the partitions: the Degree/Diameter Graph problem [14]. Partition sizes represent the sum of organizing agents as regular members and healing agents which implement bridges between partitions.

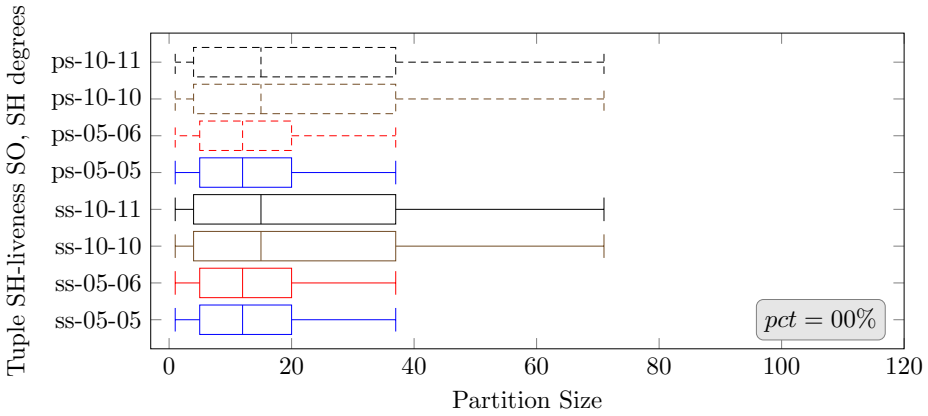


Figure 6.17: Partition size distribution without the effect of *SmartHeal* agents (0%). Sizes bounded to $\approx< 38$ for $dg = 5$ and $\approx< 76$ for $dg = 10$.

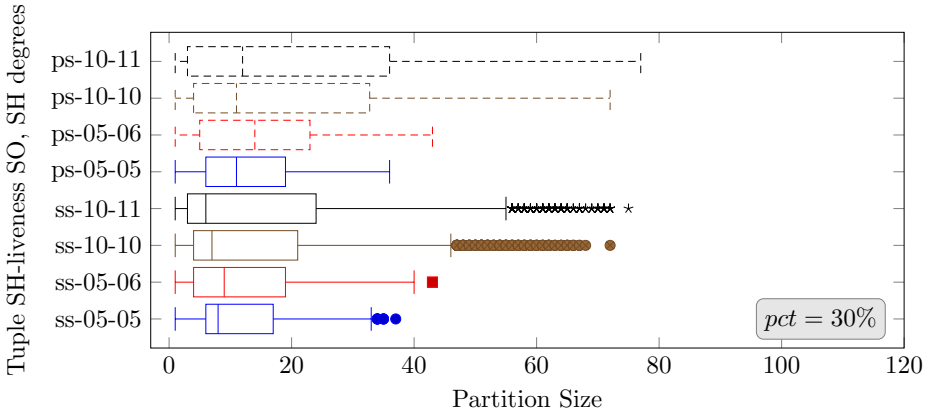


Figure 6.18: Partition size distribution for 30% of *SmartHeal* agents. Liveness *ss* deteriorates the balancing with a reduced mean and the occurrence of a large number of outlier partitions. Configurations with $dgh = dg + 1$ induce an increment in maximum size.

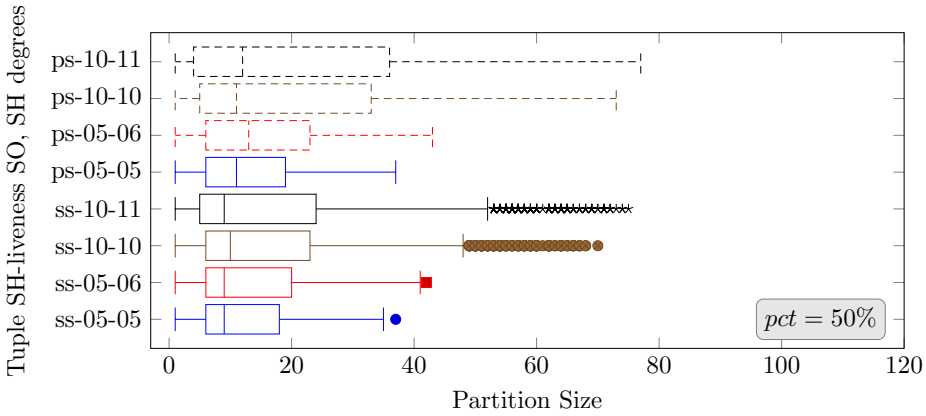


Figure 6.19: Partition size distribution for 50% of *SmartHeal* agents. Liveness *ss* continues deteriorating balancing with a reduced mean and a large number of large outlier partitions. Configurations with $dgh = dg + 1$ induce an increment in max. size.

In Chapter 5, we postulated that the resulting partitioning structure is a relevant outcome for healing agents to recover WMN's global connectivity. Ideally, SOrg agents should create the minimum number of partitions of their maximum sizes given the diameter/degree constraints, requiring a minimum amount of healing agents for inter-partition connectivity. Also, too many partitions compared to the available set of frequencies might render frequency diversity inefficient due to the reuse of frequencies by nearby WMN partitions.

Figures 6.17, 6.18 and 6.19 support the expectation above. In Figure 6.17, no *SmartHeal* agents exist (0%), while in Figures 6.18 and 6.19 there are 30% and 50% of *SmartHeal* agents w.r.t. the number of nodes, respectively. When the *SmartHeal* agents apply the *ss* liveness, the resulting size distribution of final topologies (at convergence time) show a reduction in the mean and the appearance of large outlier partitions, characterizing a deterioration in the partition size balancing. The same does not occur for the *ps* liveness (figure omitted). This trend persists for 50% of *SmartHeal*-ss, also omitted in the interest of space.

The size balancing deterioration on the topologies produced when using the *ss* correlates with the reduced efficiency of the *ss* liveness on recovering global connectivity, to confirm the expectation that a better balancing on the partitions allows for more efficient recovery of connectivity.

The current results do not evaluate the potential capacity increase in the topologies produced by the *ss* liveness, as also observed in Section 6.6.3. This aspect will be the subject of future evaluation.

Chapter 7 Explorations with SDN control planes into WMNs

This chapter consolidates work developed on the implementation of a reference architecture for SDN control planes into WMNs or applications that assumed a centralized control plane on WMNs for implementing routing schemes or time diversity increase mechanisms (TDMA).

The implementations or designs presented here had their experimental evaluation limited by the issues regarding the integration of existing software into simulated networks, as described in Section 2.11. Therefore, they represent designs or built systems that had limited simulated evaluation.

7.1 A reference architecture for SDN-based WMNs

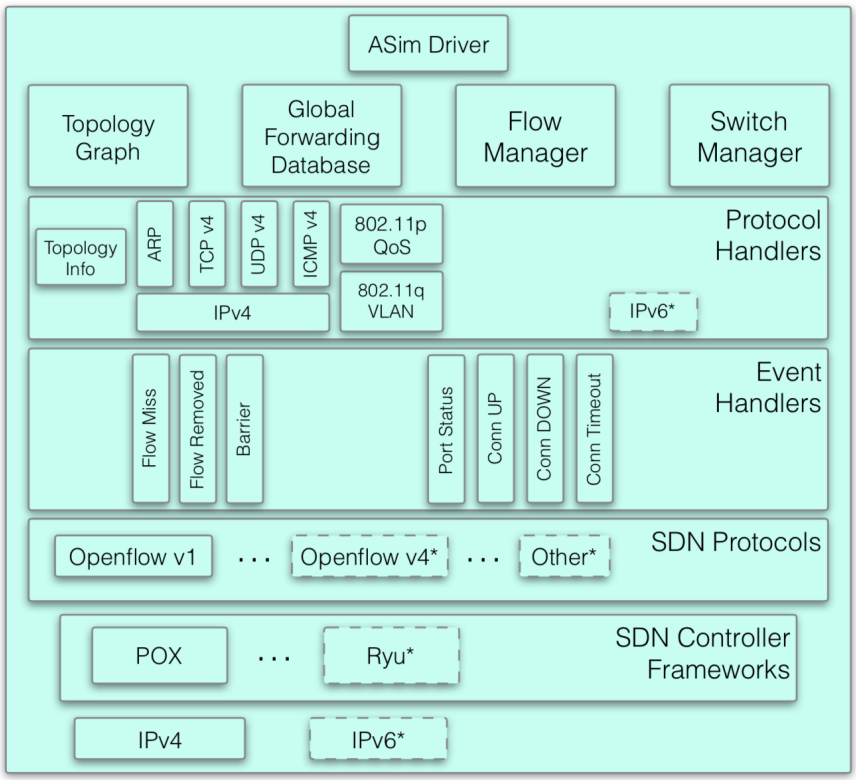


Figure 7.1: SD-WMN controller framework.

The SDN controller framework depicted in Figure 7.1 is a complete redesign of the

work in [117] specialized in the control plane of WMNs. We maintained the following principles from the original work: in-band control; multi-specialization of nodes: any mesh node forwards packets and can become a group controller as necessary; a topology graph fed by the wireless adjacency information provided by the mesh nodes. The multi-specialization fits well in our design, given that the autonomic agents in our mesh nodes can assume this additional optimization objective: deciding their role in controlling a group or not. The dashed boxes represent functions of future implementation.

We executed a preliminary evaluation limited to small WMN sizes due to the experimentation restrictions described in Section 2.11. We plan a more detailed evaluation using larger WMNs and exploring the resources advent from the centralized control to apply cross-layer routing metrics to increase WMNs capacity.

The framework is currently based on the POX Openflow controller platform [118] and supports the Openflow v1 southbound SDN protocol. It relies on a set of events provided by the controlling platform to implement the functionality necessary to the WMNs' control plane. For each type of event, an event handler is responsible for dealing with the event (see Figure 7.1 for details).

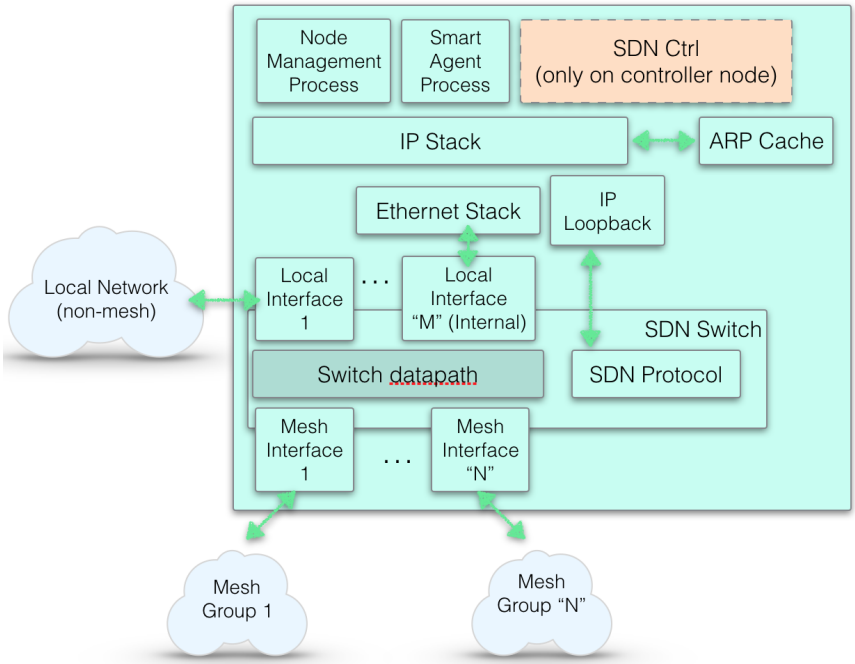


Figure 7.2: WMN node architectural view in the SDN paradigm.

Figure 7.2 describes the WMN node architecture operating in the SDN paradigm. The in-band control relies on an induction-based algorithm as follows.

- The base case is the controller node: its SDN protocol connects to the network (group) controller directly through its internal network (IP loopback).

- After the controller node's SDN switch connects, any mesh node adjacent to the controller node will be able to connect.
- Now we generalize stating that any node in the group adjacent to a connected node will be able to connect to the group's controller.

This control channel implementation benefits from the link-layer isolation enforced by our autonomic agents: a single controller is available for connectivity by the SDN protocol on the mesh nodes.

7.2 Centralized SDN-based WMN TDMA scheduler

We envision the enforcement of time diversity to promote increase scaling in the local routing regime of our hierarchical WMN architecture. We conceived a TDMA-based packet scheduling solution for WMNs anchored on an SDN control. The flow admission control inherent to the SDN paradigm is an ideal point to schedule the use of time slots. We also leverage on the *Topology Graph* of the architecture described in Section 7.1 to build the scheduling algorithm. This combination of a centralized TDMA architecture based on SDN is yet not explored in research on WMNs.

The solution includes four major components:

- a TDMA-based flow scheduler;
- a TDMA-aware mechanism to program mesh nodes (such as a TDMA-aware OpenFlow protocol);
- a TDMA Medium Access Control (MAC) in the wireless interfaces of mesh nodes;
- and a TDMA timing synchronization for the WMN.

We leave the last component - synchronization - for future work, proposing a *simulated synchronization* in a network simulation platform to allow the evaluation of the solution regarding its contribution to capacity improve. Djukic et al. [119] present a TDMA synchronization algorithm with a precision in the order of 16 microseconds which works on multi-hop wireless networks such as WMNs.

The solution output of a TDMA flow scheduler for a WMN is a set of mesh nodes that comprise the flow path, and, for each node, a set of TDMA time slots (or tokens) to be used for the transmission of the flow's packets. The scheduler can explore a combination of a modified WMN topology graph and a modified Dijkstra algorithm to provide the expected output. The graph's edges should register which flows are using each TDMA time slot, if any (time slots can be unused). The algorithm assumes the topology as a directed graph. It must perform an additional step when analyzing each neighbor vertex v_d of the current node v_s being *visited* (v_d is a potential destination node for this stage of

the path in which the current vertex we denominate v_s). The additional step comprises these actions:

1. Find the set of inbound edges to vertex v_d named E_{i_d};
2. Find the intersection of unused time slots in the edges of E_{i_d}, resulting the set T_r which is the set of time slots in which v_d can receive a packet;
3. Find the set of outbound edges to v_s named E_{o_s};
4. Find the intersection of unused time slots in the edges of E_{o_s}, resulting the set T_t which is the set of time slots in which v_s can transmit a packet;
5. Further intersect the set T_r with the set T_t: $T_a = T_r \cap T_t$;
6. The resulting set of time slots T_a can be used in the scheduling. Time slots in T_a are available because they are free for Tx in the source and free for Rx in the destination;
7. The number of time slots in T_a chosen is based on the expected QoS (in bits/s) and the speed in the edge $v_s \rightarrow v_d$.

The implementation of such an algorithm should maintain the states described above outside the graph data structure parsed by the modified Shortest Path First algorithm from Dijkstra [120], given that the final path is only known at the end of the computation. At the end of the algorithm, the chosen time slots should be committed to the graph in a particular way: any set of transmitting time slots chosen at each vertex v should also be updated in all inbound edges of the neighbors of v, reflecting the contention caused by v on its neighbors during the selected time slots.

The design described above does not account for the absence of available time slots or for the excess of available time slots. Those two cases are a restriction and opportunity, respectively. In the former, the scheduling algorithm should consider removing time slots in excess in the specific contention points. In the latter scenario, more than minimally requested bandwidth could be attributed, assuming that such behavior leads to the former case, which was already solved. Further consideration about balancing the excess bandwidth offered is necessary.

We consider this design of TDMA scheduler conservative because it admits zero interference in a time slot. However, further optimizations can be evaluated. Long-distance edges produce weak interfering signals. Thus their interference can be absorbed by a short distance edge, which produces a strong Rx signal in the receiving node. The resulting SINR level at the destination node allows decoding the Tx signal.

The OpenFlow protocol permits the use of multiple queues per switch port (a switch

port is associated with a single network interface). Those queues can be mapped to TDMA time slots. Additionally, a single OpenFlow rule admits more than one action per rule. Therefore, we could address the question of attributing more than a single time slot per-flow rule. This mapping allows a simple implementation of the second component described above: programming mesh nodes in a TDMA-aware fashion, using the OpenFlow protocol.

A possibly better alternative is using a programmable data plane (SDN switch) such as the ones supporting the P4 data plane language [121].

Further detailing should exist on how the SDN switch transfers the queue information (in fact, the time slots to be used on packet transmission) to the TDMA MAC on the wireless interface. This MAC layer ultimately is the one buffering packets until the corresponding TDMA time slots are available. One approach is the addition of special tags to the abstraction the represent the packet in the operating system. These tags are interpreted and later removed by the TDMA MAC before transmission. Such behavior internal to the node has minimum impact on the network or the network control.

7.3 Contention aware multi-path mesh routing based on centralized control

We built upon the framework described in [117] and its mesh routing algorithms to describe a *Contention-aware Mesh Routing* (CA-MR). [117] presents two mesh routing (MR) algorithms: a simple hop-count MR (a) and a higher throughput / least congested path MR (b). In (a), given a source (Src) and destination (Dst) pair, the algorithm always selects the same path, comprising the minimum amount of hops, regardless of the network load distribution. This solution is not aware of the WMN load and is unable to distribute traffic. In (b), different paths can be chosen for different flows (see [9] for the concept of flow) for the same pair $Src - Dst$. Therefore, the load distribution in the WMN interferes in the path selection. However, (b) does not take into account the contention effect in shared media wireless networks, which leads to path selections in the same contention region, limiting the benefits of this MR.

Here we elaborate on why eliminating low-speed links and about the feasibility of an iterative algorithm for that task. Let N be a Wireless Mesh Node (WNode), C a node's contention time, B the set of neighbors of a node, and $j = |B|$ the node's number of associations (to its neighbors). The set of neighbors B_i of a given node N_i is reduced by eliminating one of its associations (let this association be with the node N_e). The C_i of this node will be computed using the reduced set of associations with the size $j-1$. If the

eliminated link N_iN_e was the one with the lowest speed (in a multi-speed wireless system, links have low-speed when the perceived Signal to Interference plus Noise Ratio (SINR) of the receiver is low) in the set of association speeds of N_i, the highest contribution for C_i was also eliminated (assuming $t_{ie} = 1/s_{ie}$). Therefore, an algorithmic approach that can perform the described mechanism will have diminishing returns with time (higher contributions in the beginning) and fast convergence, assuming that the use of an iterative approach.

The implementation idea for the CA-MR is to take into account the time consumed in transmissions of a mesh node N_i, added by the time in contention due to N_i's neighbors transmissions. If N_i consumes a set of times t_{ij} for transmissions to its neighbors, all its neighbors are in contention at least for this set of times. By correspondence, N_i will be in contention during the time any of its neighbors are transmitting. Therefore, the total contention time C_i of a given node N_i in a sampling interval t_s is the sum of all its transmission times t_{ij} and all of its neighbors' transmission times t_{jp}. Furthermore, for completeness, we must add the value $t_{i_{cca}}$ to account for the time N_i spent performing a clear channel assessment (CCA). Let p denote the neighbors of each j.

$$C_i = \sum t_{ij} + \sum t_{jp} + t_{i_{cca}} \tag{7.1}$$

Although the CCA time of a wireless interface is not a commonly exposed metric by wireless APIs, by definition, this value must be small comparing to the transmitting time to reflect the efficiency of the wireless technology. Therefore, not accounting for the CCA time induces small errors to our intended contention metric.

We define the amount of actual link speed as_{ik} available when traversing a directed link N_iN_k as:

$$as_{ik} = s_{ik} \cdot \frac{t_s - C_i}{t_s} \tag{7.2}$$

Finally, the CS-MR uses the Dijkstra Shortest-Path First [120] (SPF) algorithm for path selection, assuming a derived measurement of the actual link speed metric as the weight of the edges considered in Dijkstra's SPF. The weight is:

$$w_{ik} = \frac{1}{1 + as_{ik}} \tag{7.3}$$

The expectation is that CA-MR finds paths using regions of the WMN with lower contention. We implemented the CA-MR into the reference SDN framework described in the Section 7.1; however, no evaluation of its efficacy was performed due to restrictions on experimenting in an SDN-based WMN setting as described in Section 2.11.

7.4 WMN contention minimization: a current-flow betweenness centrality application

The contention effect is as a bottleneck to parallel communications in a WMN. The contention is a necessary mechanism that allows for different network nodes to share the same communication medium: the shared wireless channel. When a node is transmitting, all its neighbors need to avoid communications: the contention time. The higher the connectivity of a node - the node's degree, the smaller is the percentage of time it has for its transmissions. Additionally, assuming wireless technology using a multi modulation physical layer to adapt to different conditions of signal strength, different link speeds will exist. The transmission time of a packet is inversely proportional to the link speed. Therefore, lower speed links will demand higher transmission times. Moreover, these lower speed links will also be the ones with longer distances, which can cause contention on a larger part of the network.

Low-speed links can be considered redundant, assuming the existence of higher speed paths that rely on multiple hops. In the WMN, intermediary nodes forward packets on behalf of an originating node. A mesh routing (MR) mechanism defines the packet forwarding by mesh nodes.

The objective of the work in this section is to reduce the contention effect in WMNs by eliminating nodes' associations, which do not contribute significantly to the network graph regarding graph connectivity and overall throughput. For example, we eliminate links with slow speeds (e.g., 1 Mbps through 13.5 Mbps) that have low network structural importance, therefore, reducing the contention introduced by them. We maintain links with high connectivity importance metric such as an edge centrality measure.

This analysis of link *importance* requires a global view of the WMN topology mapped into a graph model, which is of complex implementation on a distributed control WMN. However, assuming a centralized WMN control architecture, such as in the SDN paradigm, conventional network centrality algorithms can be applied.

7.4.1 About graph centrality

Network centrality is a fundamental mechanism for network analysis. The most basic centrality metric uses vertices' degrees to capture the ones with higher connectivity as a proxy for its centrality [122]. The betweenness concept was introduced independently by Anthonisse and Freeman, capturing the "degree to which a vertex is in a position of brokerage by summing up the fractions of shortest paths between other pairs of vertices that pass through it" [123]. In [124] a faster algorithm is introduced for computing the

betweenness centrality metric of a graph.

In [125], a novel approach for computing network centrality is presented, known as Current-flow betweenness centrality. Current-flow betweenness centrality uses the way an electric current flows in an electric circuit to model the information spreading in a graph. This strategy contrasts with the betweenness centrality strategy, which uses the shortest paths only. Current-flow betweenness centrality is also known as random-walk betweenness centrality.

7.4.2 Proposed solution

We refer to Section 7.3 to motivate the importance of link elimination in WMNs regarding exploring the overall WMN capacity. We can demonstrate the benefit of eliminating low SINR links, assuming the existence of the CA-MR algorithm. Section 2.6 presents other approaches for routing packets in a WMN using path quality metrics.

The intuition for intentional and discriminated link *elimination* is that the lower speed links are also the ones with the highest impact on the contention time of neighboring nodes: the lowest the speed the highest the transmission time maintaining the same (on average) distribution of load over the links. The basic way of deciding a flow path is the least hop count, which will tend to choose long-range links. Therefore, choosing the links with lower speeds[1]. We provide a more detailed explanation in Section 7.3.

One counter-question: why not just eliminating all low-speed links in the WMN by limiting the SINR ratio acceptable to the association between mesh nodes? This strategy can potentially limit the network connectivity in such a way that a connected WMN could be broken into many components, if this link elimination is not carefully considered by a centralized algorithm, assuming the full network connectivity. Local decisions only, therefore, are of high risk.

Low speed link elimination algorithm based on a graph centrality metric

When deciding which links should be eliminated based on the edge centrality metric, the strategy will be to remove links with low speed and low structural importance (low centrality). The criteria to define a low-speed link is a parameter to be used in this solution, starting at the mean of the edge centrality of all network links as the initial target. This initial target will be increased iteratively by a multiplying factor. The stop condition for this iterative approach will be the resulting number of components on the network graph. If the number of graph components increases in an iteration, the previous

[1] We assume a model of wireless communication with multiple speeds that are selected based on the signal quality (SINR). Higher speeds demand higher SINR, which correlates with shorter distances. Multi-speed wireless systems are common such as in the W-Fi, WiMax, and LTE physical layer models.

iteration represents the final solution. Therefore, this approach will produce a network with less low-speed connections (graph edges) without creating any additional segmentation in the network other than any previously existent. This solution can reduce the contention on the WMN without additional segmentation.

The algorithm is now presented. Let:

$\boldsymbol{G_{in}}$, an undirected Graph modeling the WMN topology with weighted edges. The edge weight w is derived from the wireless link speed, using:

$$w = \frac{150}{link\_speed} \tag{7.4}$$

$\boldsymbol{LowSpeedTarget}$, the speed limit to assume a link as a low-speed one. The following experiments assumed the value of $\frac{150}{14}$, meaning that link speeds below 14 Mega bits per second (Mbps) are considered slow speeds. The weight metric on the graph is inversely proportional to the link speed; therefore, low-speed links will have weights higher than $LowSpeedTarget$.

$\boldsymbol{G_{out}}$, the transformed version of G_{in} after the removal of low-speed links.

Input: $G_{in}, LowSpeedTarget$
Output: G_{out}

```
1  xMean = 1;
2  ec = computeEdgesCentrality(G_in);
3  comp = computeNumConnectedComponents(G_in);
4  cMean = computeMeanOfEdgesCentrality(ec);
5  while 1 do
6      cTarg = cMean · xMean;
7      G_tmp = copy(G_in);
8      for edge ∈ edges(G_in) do
9          if weight(edge) > LowSpeedTarget then
10             if edgeCentrality(edge) < cTarg then
11                 removeEdge(G_tmp);
12             else
13                 doNext = True;
14             end
15         end
16     end
17     compAfter = computeNumConnectedComponents(G_tmp);
18     if (compAfter > comp) ∨ (¬doNext) then /* Stop Conditions */
19         break;
20     end
21 end
22 G_out = lastValid(G_tmp)
```

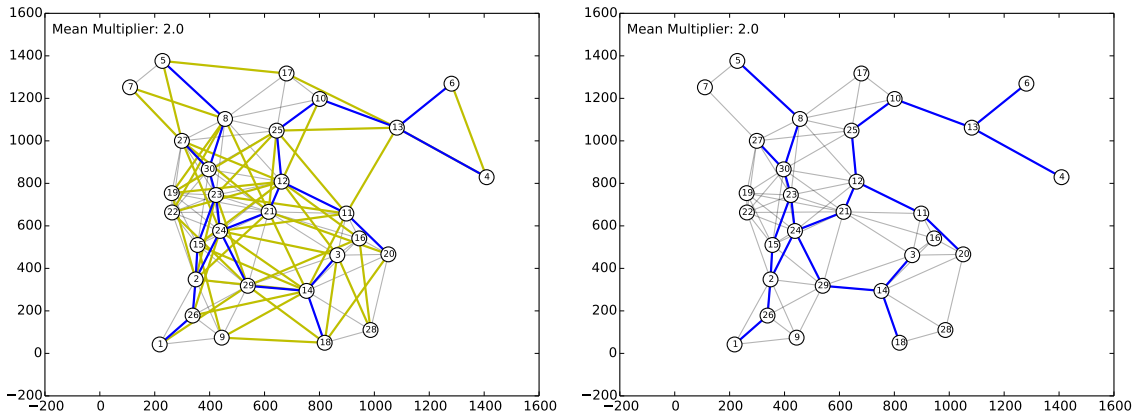
Algorithm 1: Low-speed link removal algorithm.

7.4.3 Simulated evaluation

We applied the Algorithm 1 to four different topologies produced using the the WMN emulation framework applied in [117]. The four different topologies were randomly produced by defining the position of 30 WNodes in a 1500 m x 1500 m Euclidean space. Ten of the WNodes were based on the standard IEEE 800.11g (link speeds from 6 to 54 Mbps) and twenty based on the standard IEEE 800.11n (link speeds from 6.5 to 150 Mbps). The emulation tool is capable of analytically estimating the connectivity and the respective link speed of the random topology based on WNodes' positions and Euclidean distances. The wireless signal attenuation model was the same used in [117]. Algorithm 1 was written in Python using the NetworkX library [101], which implements centrality algorithms based on [124] and [125].

The four different topologies experimented differ in their clustering characterization. They are all connected; however, the first has a single node cluster (a region with a higher

density of nodes), the second has two clusters, the third has three clusters, and, finally, the fourth has four clusters.

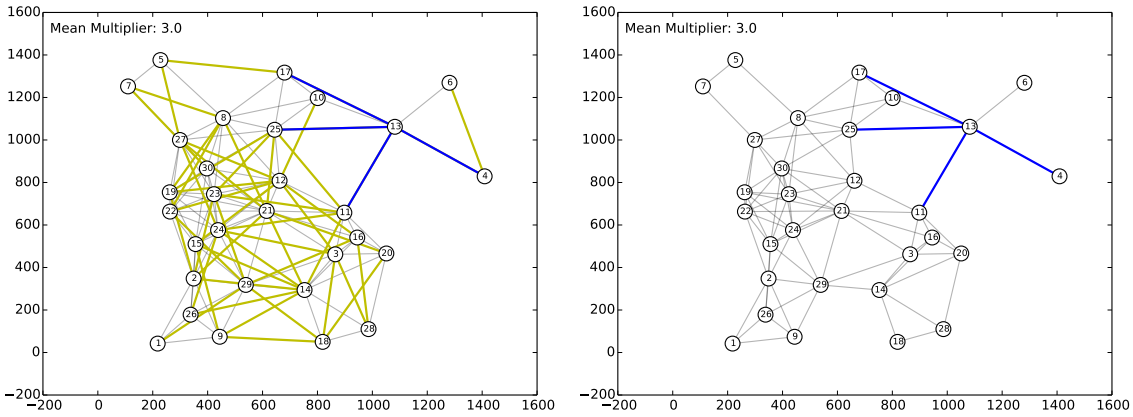


(a) Before (b) After

Figure 7.3: Single cluster network model - betweenness centrality algorithm. Blue and gray links should stay. Blue links are the *backbone*. Yellow links are appropriate for elimination.

Figure 7.3 shows the original (Before) and final (After) network models for the single cluster topology, using the *betweenness* centrality metric. The essential *backbone* of the network is captured by this centrality metric, as it values the shortest path links only. Although looking similar to a spanning-tree algorithm, our proposed algorithm does not provide any guarantees of a tree-like structure for the mentioned *backbone* highlighted in blue.

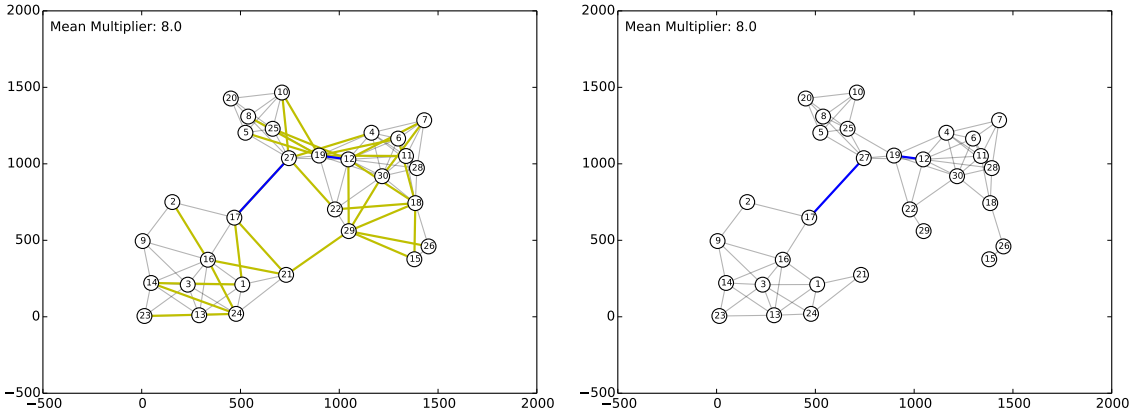
The links in yellow are low-speed links, which are candidates for elimination. In blue are the links (graph edges) with high centrality that should not be eliminated, regardless of their speeds (they are protected from elimination). For all the pictures presented here, we present only the last valid iteration of the algorithm (the final result not creating a segmentation). The final multiplying factor for this topology's target mean centrality was 2 times the mean centrality of the original network. This network originally had 133 edges, 50 edges were removed, only 1 link was a low-speed link kept out of the removal set because of its importance in the network's connectivity.



(a) Before (b) After

Figure 7.4: Single cluster network model - current-flow betweenness centrality. Blue and gray links should stay. Yellow links are good for elimination.

Figure 7.4 shows the same single cluster topology, using the *Current-flow betweenness* centrality metric. In this case, 47 out of the 133 edges were eliminated, a close value to the betweenness centrality measure (only 3 less). On the last iteration, 4 links were protected from elimination. The final topology shows more multi-path alternatives when compared with the final topology produced using the *betweenness* centrality (BC). This result is important for a mesh routing algorithm that explores multi-path links for different flows.

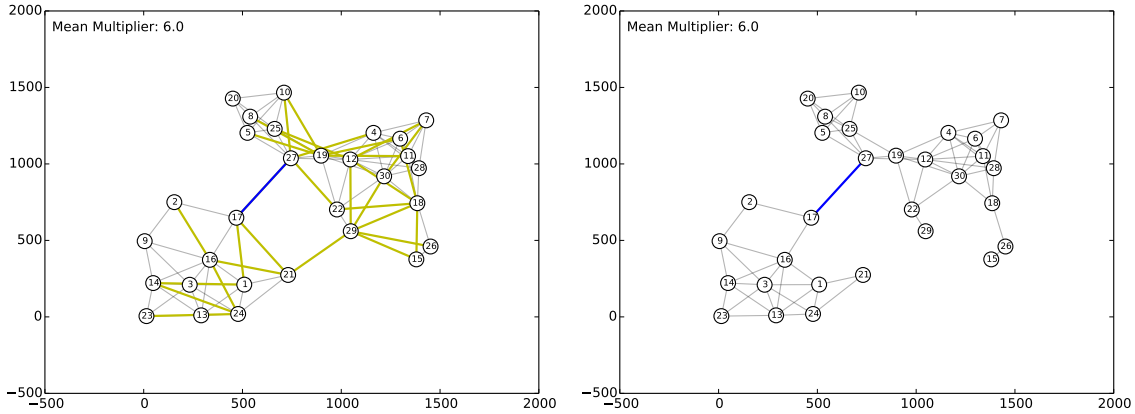


(a) Before (b) After

Figure 7.5: Two cluster network model - betweenness centrality. Blue and gray links should stay. Yellow links are good for elimination.

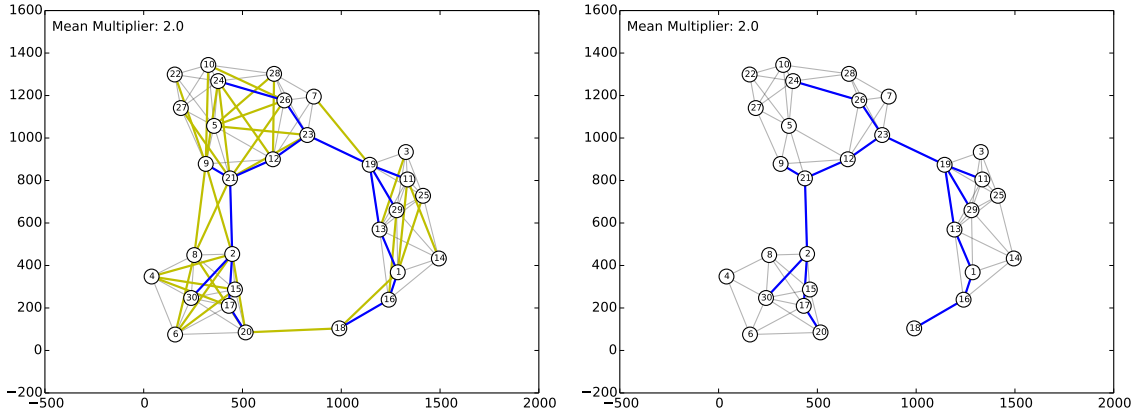
Figures 7.5 and 7.6 show the results for the two cluster topology. The final results are similar. The difference is that the *Current-flow Betweenness* centrality (CFBC) metric

demanded fewer iterations (8 on BC and 6 on CFBC). In BC, 29 low-speed links, out of 100 total links, were removed, and on CFBC, the same amount of 29 links were removed.



(a) Before (b) After

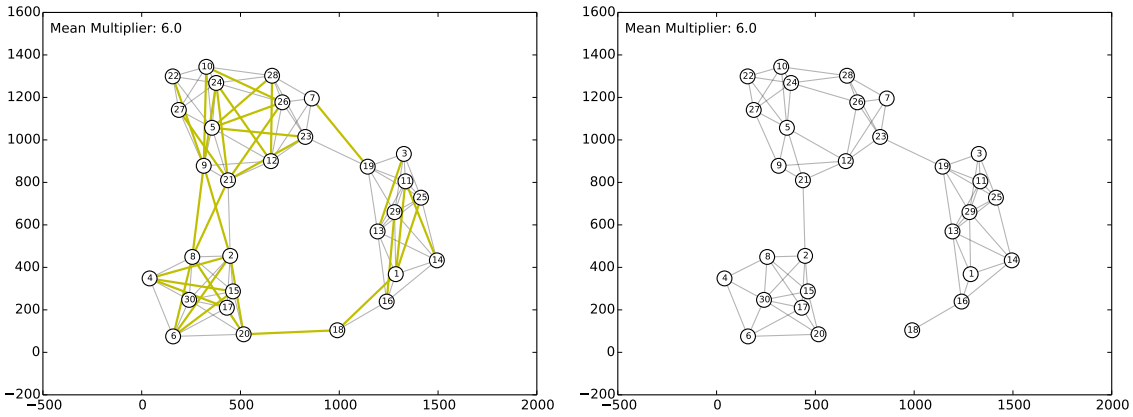
Figure 7.6: Two cluster network model - current-flow betweenness centrality. Blue and gray links should stay. Yellow links are good for elimination.



(a) Before (b) After

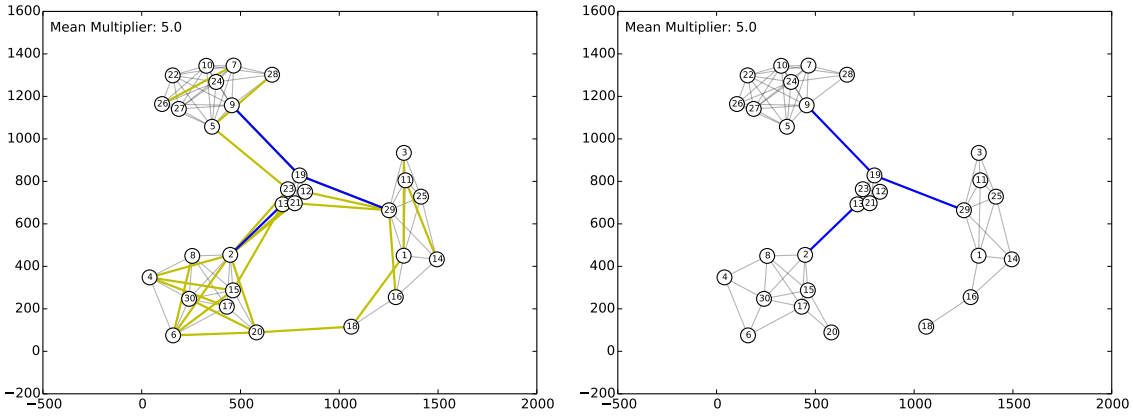
Figure 7.7: Three cluster network model - betweenness centrality. Blue and gray links should stay. Yellow links are good for elimination.

Figures 7.7 and 7.8 show the results for the three cluster topology. Again, the final results are similar. In this case, the CFBC metric demanded much more iterations (2 on BC over 6 on CFBC). In BC, 30 low-speed links, out of 100 total links, were removed, and on CFBC, 31 links were removed (only one more).



(a) Before (b) After

Figure 7.8: Three cluster network model - current-flow betweenness centrality. Blue and gray links should stay. Yellow links are good for elimination.



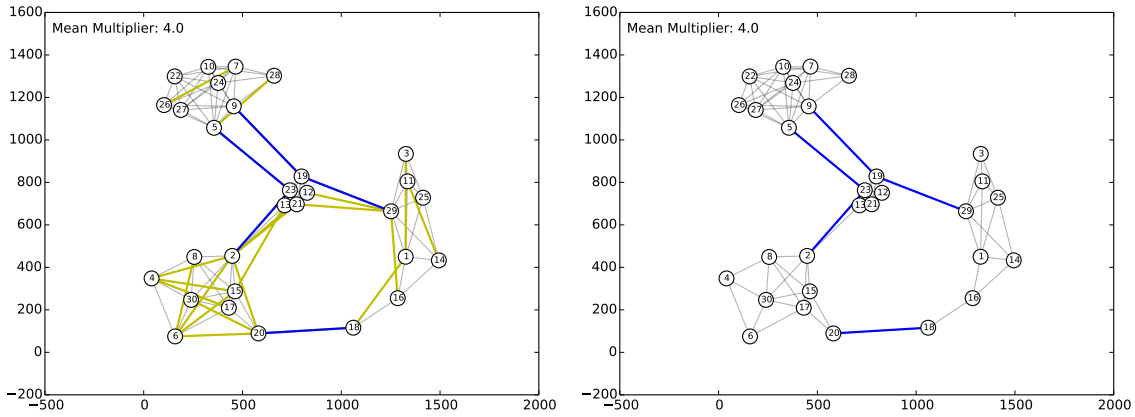
(a) Before (b) After

Figure 7.9: Four clusters network model - betweenness centrality. Blue and gray links should stay. Yellow links are good for elimination.

Figures 7.9 and 7.10 show the results for the four cluster topology. This time the number of links eliminated diverged slightly: 23 on BC and 20 on CFBC, out of a total of 99 links on the original network. BC demanded 4 iterations and CFBC only one more: 5. Once more, the CFBC provides resulting topologies that allow more parallel communications.

Comparing with a *no-centrality* baseline

Figure 7.11 presents a comparison of the results obtained by applying Algorithm 1 using the two different centrality metrics with the results obtained by applying a simple algorithm ($Blind$) which does no use any centrality metric. The *Original* line shows



(a) Before (b) After

Figure 7.10: Four clusters network model - current-flow betweenness centrality. Blue and gray links should stay. Yellow links are good for elimination.

the original amount of edges on the topologies before link elimination. The other lines represent the two centrality metrics used in Algorithm 1, and $Blind$ is this baseline. $Blind$ behaves as follows:

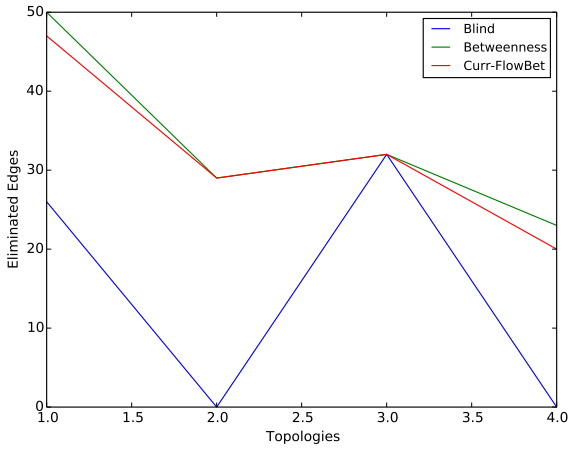


Figure 7.11: Comparing the centrality based algorithm with a *no-centrality* ($Blind$) baseline.

a) Given an initial low-speed threshold LS, $Blind$ eliminates links with speeds below LS;
b) If no segmentation was produced in the WMN, $Blind$ stops;
c) Otherwise, $Blind$ reduces LS and makes a new attempt such as in a);
d) If the LS threshold becomes 0, $Blind$ stops with no link eliminated.

The results of $Blind$ in Figure 7.11 were obtained with LS set for the same low-

speed threshold of the other experiments: 14.0 Mbps. $Blind$ serves as a baseline for the centrality based algorithm because of its simplicity and its possibility to be used in a distributed way (executed individually by all nodes). However, $Blind$'s results show that it is ineffective in 2 out of the 4 topologies experimented, and always producing inferior results.

7.4.4 Section conclusion

This section presented and demonstrated a mechanism for minimizing contention in WMNs. The mechanism is based on the SDN paradigm, which allows the execution of centralized algorithms using information representing the network model. In this setting, we proposed an algorithm using edge centrality measurements and an iterative approach. The algorithm is capable of identifying links with low speed and low structural importance (low edge centrality measure), allowing the *elimination* of these links in the network connectivity. We expect the resulting topology with reduced connectivity in the number of links to present reduced contention and a higher probability for parallel communications, which can produce higher overall throughput. No additional segmentation on the networks' models is allowed by design. Further evaluation of this solution should be performed with real network traffic, various topologies, and different sizes for characterization of the algorithm's contribution to the WMN capacity scalability.

The approach described here for contention minimization allows increased parallelism of transmissions in the WMN, which should increase the overall WMN throughput. However, the increased parallel transmissions imply increased interfering energy during packet reception (the N element of the Signal to Noise Ratio (SNR) or the I of the SINR). Consequently, only low SINR associations should be candidates for elimination. Additionally, the simplified prove showed in section 7.3 demonstrates that these low-speed associations are the ones that cause a higher impact on the contention time. A thorough evaluation should take into account this consequence, requiring evaluation beyond the analytical approach applied here, as mentioned above.

Chapter 8 Conclusion and future work

This decade portrayed significant evolution in communication networks such as massive capacity data center interconnects with 100/400 Mbps single link speeds between server racks and above Tbps of aggregated per-rack capacities [126] that support the ever-increasing scales of cloud computing. In the wireless realm, the 4G (LTE) and 5G wireless standards for cellular networks offered an impressive increase in per device capacity, the number of connected devices, and on supporting diverse use-case scenarios [127]. Also, new paradigms for network design transition networks from inflexible architectures to programmable control and data plane paradigms (SDN, NFV - Network Function Virtualization, P4 data-plane programming language [121]). Despite these advancements, research regarding the worldwide consumption of connectivity claims that roughly 40% of the world population is poorly or not connected [1]. Although the lack of connectivity, in general, is a more significant concern in the lower-income nations, even the population of developed and wealthy democracies such as the U.S. suffer from the lack of the required connectivity [2]. Cost is a leading factor limiting the spread of connectivity [1, 10], and alternative network architectures using wireless communication and aiming for lower cost appear as solution candidates [10].

We propose, as a contribution to include the unconnected, the modernization of the WMN architecture through the automatic support of programmable control planes at large scales and the enforcement of frequency diversity at low-cost, low-complexity settings. A sought after property of the WMN architecture is its ability to provide broad coverage, at low-cost. In WMNs, mesh nodes provided with short-range radios cooperate in forwarding each other's packets from source to destination supported by a network control mechanism at the routing layer to achieve wide coverage. Two crucial limitations restrict the applicability of WMNs at scale: inflexible and hardened control planes based on distributed protocols and reduced capacity at large scales WMN [19, 23, 32].

We devised a method to automatically manipulate the formation of Wireless Mesh Networks (WMNs), inducing topologies structured as a set of interconnected partitions. A distinct characteristic of our approach is the reliance on local information, contrasting with previous solutions that rely on centralized algorithms, and suffer from limited scalability and weak robustness to changes in topologies, or merely failures of critical elements. Our solution maintains WMNs extensible while supporting new control paradigms and

frequency diversity.

We designed a combination of self-organizing (Chapters 4 and 5) and self-healing (Chapter 6) autonomous [12] agent models that concurrently manipulate WMN topologies in realistic, low-cost, low-complexity, and size, density unconstrained WMNs. We materialize these agents as software components embedded into mesh nodes, each agent responsible for controlling the connectivity decisions of one wireless interface. Our solution requires an average number of wireless interfaces per mesh node above one. However, for simplicity and cost-efficiency, we designed this solution such that an average number below two interfaces per node is sufficient for proper operation. We show that with 1.2 interfaces per node, our method achieves global connectivity with high probability.

We named our agent designs *SmartOrg* and *SmartHeal*. The *SmartOrg* self-organizing agents dynamically and concurrently evolve a large WMN node placement into independent topology partitions. The agents enforce as invariants to the partitions their maximum diameter in mesh node hops, and the maximum degree of mesh node interfaces. Also, the induced WMN partitions are isolated at the physical - possibly different frequencies - and link layers - different link-layer network ids.

Concurrently to the operation of the self-organizing agents, we designed the secondary *SmartHeal* self-healing agents that dynamically interconnect partitions to recover broad WMN connectivity while respecting diameter and node interface degree invariants of partitions.

The physical layer partitioning allows for increased parallelism of communication flows on different partitions supported by potential frequency diversity. We elaborate on the future enforcement of frequency diversity using self-optimizing agents on the *Future Work* section that follows.

The partition invariants guaranteed by the self-organizing, self-healing agents will bound the control latency and control workload despite the unconstrained settings (density, scale) assumed, turning practical SDN control planes on each partition as detailed on Section 5.1. Moreover, our formation approach based on elected partition origin nodes turns them into well-known candidates for the execution of the SDN control plane functions on partitions or SDN controllers.

For our evaluations, we combined an experimental approach based on simulation with numerical evaluation. Our experiments applied our custom-built modular experimentation platform (Chapter 3) that supports fast prototyping of agent models and the precise simulation of wireless communication based on the reuse of well-known simulation frameworks.

Using an extensive set of experiments, we showed that the distributed and concurrent execution of *SmartOrg* and *SmartHeal* in the mesh nodes converges to stable and

reasonably balanced partitioning solutions in bounded time. After convergence, we found that the autonomic properties, or partition invariants, are maintained. We described optimizations to reduce the convergence time and the agents' effort regarding moves between partitions. To verify robustness to failures, we evaluated our method under 100% of node churn, representing a case of the synchronized activation of nodes on the return of a power outage. A complete WMN system of 1000 nodes converged to a stable solution in under seven epochs of operation, each epoch of 90 seconds. These results persisted for varied distribution and densities of node placements.

8.1 Future work

The evaluation of large-scale wireless networks is a challenge when it comes to novel paradigms that infuse software into networks such as Software Defined Networks. Simulation has been a successful strategy to evaluate new designs for wireless networks, given its high accuracy achieved at a low cost. However, simulators cannot easily reuse existing software such as SDN switches and controllers integrated with simulated wireless models. One needs to create models to represent existing software, which is inefficient and error-prone. Emulation platforms aim at closing this gap; however, existing solutions which are accurate do not support a large-scale of emulated software [83], or support large-scale emulation by relaxing an accurate representation of wireless models [72]. Moreover, emulation solutions lack the support for fast prototyping of agent models applied in autonomic networks.

In Chapter 3, we showcase an emulation architecture that splits the agent design and decision making from the implementation of network models. This design reuses well-known network simulators for network modeling, while a separate component allows agents to reason and command network decisions. Using the principle of containerization, or lightweight virtual machines, one can integrate existing software into simulated network models. However, operating system kernels demand profound adaptations to consistently synchronize the simulated timing into processes and kernel functions consumed in containers. Solving this problem permits the realization of accurate, scalable emulations that also support fast prototyping of the emulation of autonomic wireless networks.

We left for future evaluation determining whether we can yet improve the balancing of our WMN structure regarding the size of partitions by changing the threshold limits (the *sp* parameter) of the *partition size* liveness function as described in the Section 5.6.2. While reducing this threshold might enforce better balancing, it might also induce undesirable increases in the time to convergence.

How far can autonomous behavior go for the benefit of large-scale WMNs? We envision going beyond autonomous mesh nodes, turning the partitions, themselves, into autonomic entities. Autonomous partitions materialize as self-optimization agents on partition origin nodes that cooperate with the other partition members for optimizing system parameters. Examples are improving frequency diversity between partitions and adapting to the number of available resources through the selection of per partition interface degree limits, or per partition diameter limits. We envision maintaining the principle of relying on local information; however, local to the perspective of the partition: internal and neighboring partitions.

Figures 8.1 and 8.2 show a preliminary 2D and 3D perspectives of self-optimization of frequency diversity in which each agent on the partition origin nodes coordinate changes in the partition channel with the other partition members to minimize interference from neighboring or overlapping partitions. The cooperative algorithm consists of partition member nodes reporting to the self-optimizing agent on the partition origin node their wireless interference profile in the number of external nodes on each frequency. The self-optimizing agent determines the least *busy* frequency for the operation of the partition. This interference mitigation approach has the advantage of capturing the impact of other networks beyond the WMN we implement.

The abstraction of partitions as autonomous entities also permits the realization of wide-area routing. We envision a solution integrating SDN-based intra-partition routing and distributed routing, assuming our partitions as *nodes* of the distributed routing scheme. The flexibility of centralized algorithms into partition controllers allows detecting distributed routing traffic on mesh nodes interconnecting partitions and forwarding them to other interconnecting nodes. This setting characterizes a hierarchical (intra and inter partition routing regimes) and hybrid (centralized and distributed) routing. Yap *et al.* [31] is an example of integrating SDN and existing routing protocols, BGP, in their case.

We consider the alternative of cooperative, multi-domain SDN control for our partitioned WMNs. In this setting, each partition represents an SDN control plane domain, and domains cooperate in implementing a distributed routing scheme for global reachability. The advantage here is that the flexibility of control plane programmability is maintained, contrasting to the previous solution reusing existing routing tools based on fixed routing rules.

A compelling use case is the application of the principles of the autonomic formation of wireless mesh networks that we described into the setting of small cells in 5G networks. In the small cells setting, sharing resources such as a common is imperative, given the density of cells that present a small radius of 200 meters or less. In this setting, the

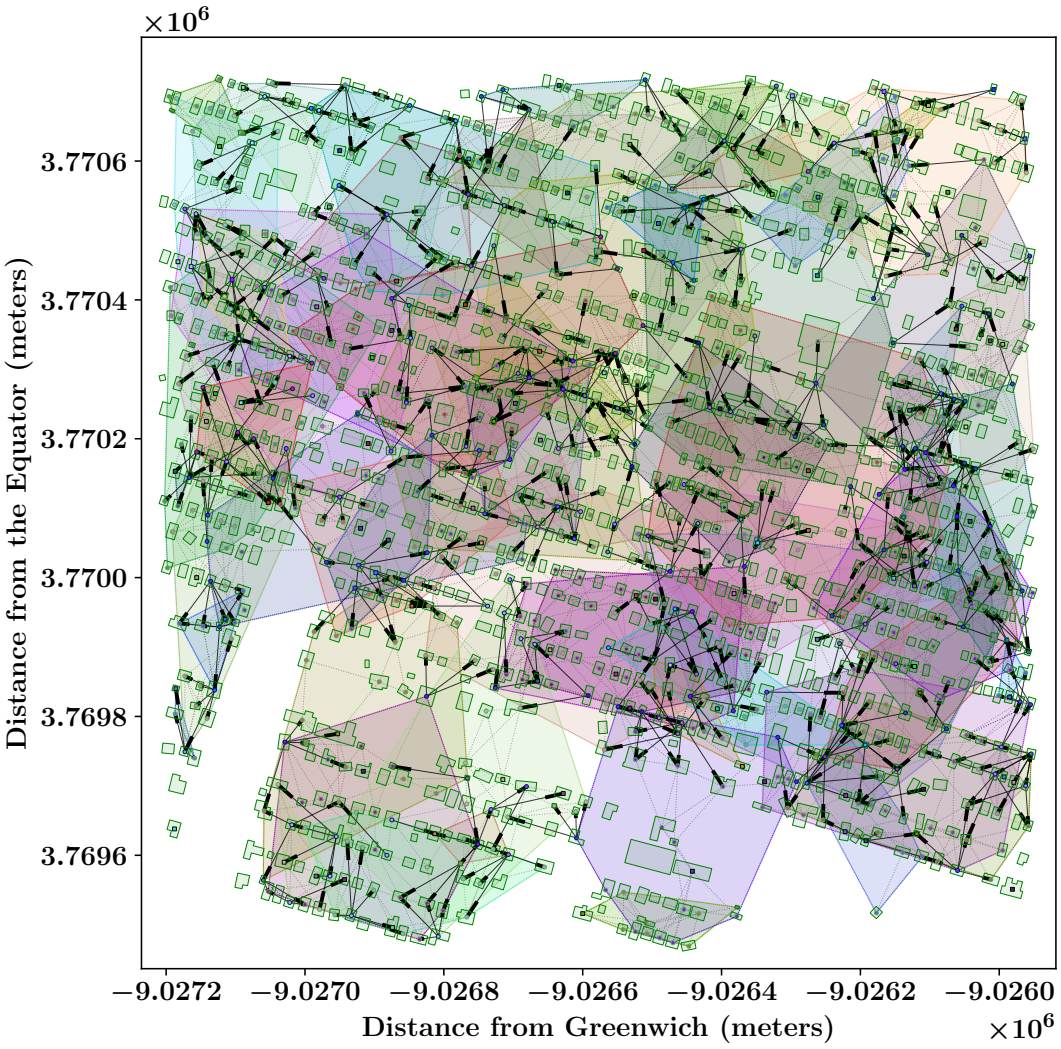


Figure 8.1: Autonomically created WMN topology partitioned and connected by the *SmartOrg* and *SmartHeal* agents. Overlapping partitions evidenced by the partitions' convex-hull. Dense edges show inter-partition connectivity by *SmartHeal* agents on 25% of nodes. Same wireless standard and node placement of Fig. 5.1. Maximum interface degree $\{5, 6\}$, maximum partition diameter 6. Final connectivity of 99.6%.

scale in the number of cells increases due to their shorter radius - advent from the use of millimeter waves - comparing to current 4G cells. We envision that such a scale of cells might benefit from an autonomous formation of clusters of cells to become efficient and cost-effective regarding resource sharing.

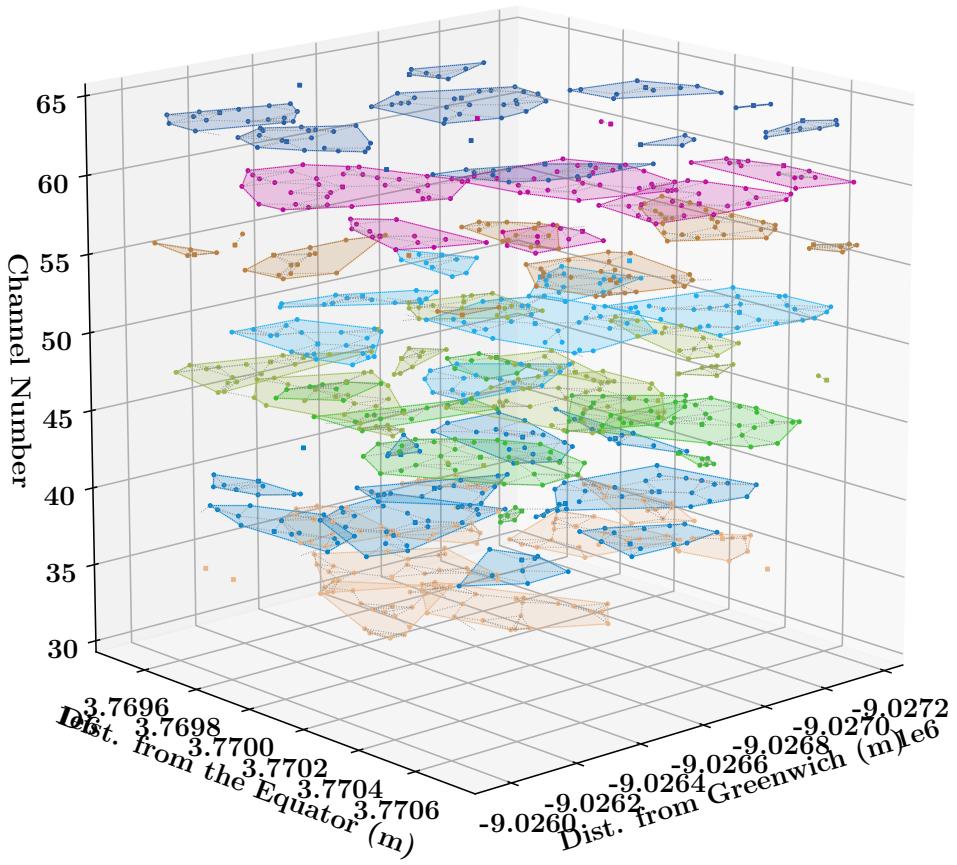


Figure 8.2: 3D perspective of the autonomically created WMN topology from Fig. 8.1, using the same settings. Healing connections removed for simplicity. Frequency diversity enforced by self-optimizing agents on origin nodes (part of future work). Shows a visual representation of achieved diversity. Each color and the channel numbers on the *Channel Number* axis represent a frequency.

Appendix A Appendix

A.1 Additional information about CWNs

In following the sub-sections, we present CWNs examples to illustrate their reason to exist, their topology organization, applications and traffic pattern.

A.1.1 Village Telco

In a presentation [128] at the International Summit for Community Wireless Networks, Steve Song describes the *Village Telco* (VT) project in South Africa, also depicted in [64]. Song enforces the importance of providing local voice communication (a local service over the underlying mesh network - VoIP) to increase the usefulness of the CWN. Due to the lack of a low-cost mesh device with an analog POTS[1] interface, Song's foundation facilitated the creation of a specialized mesh device hardware including the POTS interface and with rugged characteristics, allowing cheap and abundant analog telephones to be used in a mesh network (the Mesh Potato router - MP). Although Song's foundation started of by sponsoring groups to provide training on *hacking* computing devices and antennas (such as the *cantenna* - an antenna on a can) to create mesh routers, they found that the complexity of building mesh networks was a major restricting factor.

In the network topology described in [64], the VT network used single radio devices operating on the same frequency (no frequency diversity). Special *Supernodes* based on a one or more commercial routers with high-power radios performed the segmentation of the network into WMN islands. The supernodes had up to 2 Km reach to connect distant WMN islands. MPs would use virtual wireless interfaces operating in different L2 networks (different SSIDs and BSSIDs), however, sharing the same underlying physical wireless interface and frequency. In effect, all WMN islands connected to the same radio of a given Supernode share the same frequency. Such L2 segmentation adds no capacity to the system, and different L2 networks (the local AP-based access network, the mesh network, and the connectivity to the supernode) will interfere with each other.

Recent devices from the Village Telco project (MP2 - Mesh Potato 2 [129]) include two WiFi radios (one for the mesh network and other for giving access to users through an access point), an analog POTS phone line (remotely accessed through an Ethernet

[1] Plain Old Telephony System

cable), PoE (power over the Ethernet cable), an USB port (for additional devices such as a 3G/4G modem for Internet uplink), and internal storage to support local content such as the World Possible's *RACHEL* offline that we describe in the sub section A.1.2.

The topology derived on the VT project, namely WMN islands interconnected through bridging devices (the VT's supernodes) is equivalent to a manual implementation of the work performed by our *Standard* Agent model and its dual-objective autonomic functions: partition and recover global connectivity. The authors of [64] claim that the network architecture and its manual management tools allowed the creation of community sponsored and operated networks, providing useful local services in extreme scenarios of "... infrastructure poverty, technical literacy poverty, and financial poverty." However, the complex RF aspects of mesh management to mitigate interference still demanded specialized practitioners. Their proposed approach to tackle these issues was to create diagnostic tools and instructions for manual problem-solving.

Again, we contrast these statements with this research's approach of automated network formation and manipulation in the physical layer on which the VT's RF issues exist.

An alternative firmware for the Village Telco's MP2 mesh device (*Wildernets* - [130]) added more local services in the form of Instant Messaging (IM), made viable due to the increased MP2's memory profile. Also, Wildernets can incrementally and automatically distribute the available application services (analog - POTS - and digital - SIP - telephony, DNS, DHCP, IM, Web Serving) through the added Wilderness-based mesh routers to support more users and a distributed traffic load.

The processing capacity of the embedded devices used by Village Telco project has dramatically been surpassed by a new generation of embedded computers such as the Raspberry Pi (RPi), allowing the creation of more advanced applications that demand increased processing capacity and memory. To be found in the new generation of small computers such as RPi is the tight integration of WiFi radios into the hardware, supporting features such as external antennas and multiple radios. In the RPi and similar devices, the USB bus is the preferred mechanism for additional hardware, adding complexity and vulnerability to a rugged device setting such as a wireless mesh router application.

A.1.2 Replicating Internet content locally: World Possible's projects

Adding to the importance of providing locally available content that is independent of an Internet uplink, the Village Telco project promotes the offline use of high-quality educational web content available by the World Possible organization [131]. World Possible offers the products OER2Go (Open Educational Resources to Go), and the RACHEL packaged web serving device. OER2Go is a "... collection of educational websites repack-

aged for download and offline use ..." aiming at the consumption of education resources in areas which the Internet is limited, expensive, or non-existent. OER2Go includes content such as Wikipedia, Khan Academy, TED, and many more, using rich web, video, and applications for offline consumption. RACHEL (Remote Area Community Hotspot for Education and Learning) is a portable plug-and-play server that includes OER2Go and has a WiFi access point to allow turnkey access to educational content.

Surprisingly, RACHEL has been used even in places in the U.S. such as prisons and health centers, providing a closed set of educational content without Internet access. Additional conceived use cases are on ships, or planes (real use instances not disclosed). RACHEL can be updated using a temporary Internet connection or from update packages on SD memory drives.

RACHEL is a single point of access meant for a small group of concurrent users (20 to 50 depending on the type of content consumed). In a WMN setting, one can immediately envision that such content can be spread around the WMN to reduce the contention on accessing a single point of the network and to increase the number of concurrent users supported.

A.1.3 The Serval Project

The Serval Project [132], [133] has seen success in creating what is believed to be the world's first practical unlicensed spectrum mesh mobile telephony system. It builds on top of other successful projects such as the VT's Mesh Potato A.1.1, Asterisk (a software-based telephony system). Beyond an arbitrary integration of other solutions, contributes the DNA (Distributed Numbering Architecture) that allows the dynamic and ad-hoc formation and querying of a phone numbering system (fundamental for a disaster condition the requires unassisted system set up). As a telephony system, its traffic profile is local and PtP like. In [134], authors describe a mechanism for automatically bootstrapping network addresses on a Serval-based network in the face of a disaster scenario.

The project originated in an Australian University, and one of the use cases and field demos applied the technology on the Australian outback, a remote region with few to none infrastructure at the time. This project demonstrated a minimalistic approach in which only mobile phones based on the Android system operated as client and servers. Although using mobile devices, basing the communication on WiFi technology required a non-mobile operation. Mobility is supported in between uses. No discrimination over the mesh topology is performed which can lead to unpredictable performance issues due to increased node density.

In [132] authors claim that Serval employs a security framework based on asymmetric

cryptography in which network addresses on the mesh network are public keys (no details on how to create the associated private keys given an arbitrary public key). Such a design decision provides privacy on the communication between any two node pairs. In this Youtube video [135], the Serval project is explained with the intention of use in Venezuela as a communication system alternative in the face of the centralized control of the Venezuelan's government. The presentation here serves the purpose of showing how manual and lengthy a two node set up is. In [136], authors demo a MeshMS (Mesh-based SMS) using the Serval Project. Other documented demos exist in Nigeria, New Zealand (by the Red Cross), Boston and Washington, DC in the U.S.

A.1.4 Gulfi.net

Guifi.net in Spain is a CWN that stands out due to its size (currently +30K nodes) and its degree of administrative organization [137] (Gulfi.net Foundation). It is the largest CWN that have ever existed at any point in time. Originated by enthusiasts interested in an experimentation testbed and voluntary work, it continued to have its community nature while also proving possible to be a competitive alternative for Internet service providing in Spain. Its unique peer-to-peer agreement for joining the network maintains the principles of voluntarism and shared responsibility, while its service providing initiative brings funds to support expensive to provision and maintain such as fiber-based backbones and sophisticated management.

Concerning structure, Guilfi.net uses fiber backbones for long-distance high-capacity connectivity, uses PtP, and PtM wireless for medium distance medium capacity connectivity (the majority of its links), and, lastly, wireless mesh islands comprise short distance connectivity [65]. Such an architecture requires automated and yet highly-demanding network management. Besides, Gulfi.net has Internet peering agreements with large-scale Internet providers.

Available applications initially comprised internal Internet-like services, dubbing a parallel Internet experience. Although still existent, the original internal applications now co-exist with applications available on the Internet.

A.1.5 Athens Wireless Metropolitan Network

Athens Wireless is a large (more than 2500 nodes in 2010) [62] and urban network which primary goal is not to provide Internet access, instead promote the social interaction of its members: "... we exist even if the Internet does not exist ... everybody creates services and provide services to the community". This characteristic of Athens Wireless

in unique amongst CWNs. Members replicate Internet services internally such as portals, websites, messaging systems, e-mail, VoIP (SIP protocol), IPTV broadcast and VOD (Video On Demand), file services, DNS. Community members are mostly educated and also interested in technical learning and hacking (the network as a testbed). However, the primary motivation is to socialize and become a popular member through contribution and engagement.

Regarding topological structure, there are islands of WMNs (using the OLSR routing protocol) interconnected by PtP links and Border Gateway Protocol (BGP) routers. Such an approach implements a logical (at the routing level) segmentation of routes which constrains the mesh routing with OLSR to bounded sizes [62].

Athens Wireless is organized by an association, relying only on member funding (no external grants). Attempts to obtain grants or Internet connectivity were not successful. The limited Internet access relies on VPN tunnels from clients to the rare Internet gateways [62].

Bibliography

[1] The Economist Intelligence Unit, "The Inclusive Internet Index: Bridging digital divides," Internet.org, Tech. Rep., 2017, p. 39.

[2] J. Kahan. (2019). It's time for a new approach for mapping broadband data to better serve Americans, [Online]. Available: https://blogs.microsoft.com/on-the-issues/2019/04/08/its-time-for-a-new-approach-for-mapping-broadband-data-to-better-serve-americans/ (visited on 01/12/2020).

[3] R. Katz, "Impact of broadband on the economy," International Telecommunication Union - ITU, Geneva, Switzerland, Tech. Rep., 2012.

[4] S. Srivathsan, N. Balakrishnan, and S. S. Iyengar, "Scalability in Wireless Mesh Networks," in *Guide to Wireless Mesh Networks*, 2009, ch. 13, pp. 325–347.

[5] S. M. Sheikh, R. Wolhuter, and H. A. Engelbrecht, "A survey of cross-layer protocols for IEEE 802.11 wireless multi-hop mesh networks," *International Journal of Communication Systems*, vol. 30, no. 6, pp. 1–32, 2017.

[6] L. M. a. Bettencourt, J. Lobo, D. Helbing, C. Kühnert, and G. B. West, "Growth, innovation, scaling, and the pace of life in cities.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, no. 17, pp. 7301–7306, 2007.

[7] L. M. Bettencourt and G. West, "A unified theory of urban living.," *Nature*, vol. 467, no. 7318, pp. 912–913, 2010.

[8] S. Arbesman, J. M. Kleinberg, and S. H. Strogatz, "Superlinear scaling for innovation in cities," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 79, no. 1, pp. 1–5, 2009.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 03/2008.

[10] Microsoft Inc. (2019). Microsoft Airband: An update on connecting rural America, [Online]. Available: https://news.microsoft.com/rural-broadband/ (visited on 01/10/2020).

[11] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 01/2015.

[12] A. Berns and S. Ghosh, "Dissecting self-* properties," in *SASO 2009 - 3rd IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, 2009.

[13] L. Lamport, D. Malkhi, and L. Zhou, "Vertical paxos and primary-backup replication," in *Proceedings of the 28th ACM symposium on Principles of distributed computing - PODC '09*, 2009.

[14] M. Miller, "Moore graphs and beyond: A survey of the degree/diameter problem," *Electronic Journal of Combinatorics*, vol. 20(2), 2013.

[15] G. B. West, J. H. Brown, and B. J. Enquist, "A general model for ontogenetic growth," *Nature*, vol. 413, no. 6856, pp. 628–631, 10/2001.

[16] M. I. G. Daepp, M. J. Hamilton, G. B. West, and L. M. A. Bettencourt, "The mortality of companies," *Journal of the Royal Society Interface*, vol. 12, no. 106, 2015.

[17] I. F. Akyildiz, "A survey on wireless mesh networks," *IEEE Communications Magazine*, vol. 43, no. 9, pp. 23–30, 2005.

[18] A. Özgür, O. Lévêque, and D. N. Tse, "Hierarchical cooperation achieves optimal capacity scaling in Ad Hoc networks," *IEEE Transactions on Information Theory*, vol. 53, no. 10, pp. 3549–3572, 2007.

[19] G. Alfano, M. Garetto, E. Leonardi, and V. Martina, "Capacity Scaling of Wireless Networks With Inhomogeneous Node Density: Lower Bounds," *IEEE/ACM Transactions on Networking*, vol. 18, no. 5, pp. 1624–1636, 10/2010.

[20] Q. Liu, X. Jiang, and X. Qiu, "The Effects of Topology on Throughput Capacity of Large Scale Wireless Networks," *Information*, vol. 8, no. 1, p. 32, 2017.

[21] U. C. Kozat and L. Tassiulas, "Throughput capacity of random ad hoc networks with infrastructure support," *International conference on Mobile computing and networking*, p. 55, 2003.

[22] M. Grossglauser and D. Tse, "Mobility increases the capacity of ad-hoc wireless networks," *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, vol. 3, pp. 1360–1369, 2001.

[23] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[24] S. Yi, Y. Pei, and S. Kalyanaraman, "On the capacity improvement of ad hoc wireless networks using directional antennas," *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing - MobiHoc '03*, p. 108, 2003.

[25] P. Li, C. Zhang, and Y. Fang, "The capacity of wireless ad hoc networks using directional antennas," *IEEE Transactions on Mobile Computing*, vol. 10, no. 10, pp. 1374–1387, 2011.

[26] L. Kleinrock and J. Silvester, "Optimum Transmission Radii for Packet Radio Networks or Why Six is a Magic Number," in *Proceedings of the IEEE National Telecommunications Conference*, Birmingham, Alabama, 1978, pp. 4.3.2–4.3.5.

[27] E. Royer, P. Melliar-Smith, and L. Moser, "An analysis of the optimum node density for ad hoc mobile networks," *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*, vol. 3, pp. 857–861, 2001.

[28] J. Jun and M. L. Sichitiu, "The Nominal Capacity of Wireless Mesh Networks," *IEEE Wireless Communications*, vol. 10, no. 5, pp. 8–14, 2003.

[29] T. Moscibroda, "The worst-case capacity of wireless sensor networks," *Proceedings of the 6th international conference on Information processing in sensor networks - IPSN '07*, p. 1, 2007.

[30] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat, "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network," *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 183–197, 2015.

[31] K.-K. Yap, M. Motiwala, J. Rahe, S. Padgett, M. Holliman, G. Baldus, M. Hines, T. Kim, A. Narayanan, A. Jain, V. Lin, C. Rice, B. Rogan, A. Singh, B. Tanaka, M. Verma, P. Sood, M. Tariq, M. Tierney, D. Trumic, V. Valancius, C. Ying, M. Kallahalla, B. Koley, A. Vahdat, M.-H. Holliman, A. Nara-Yanan, A.-J. Singh, and M.-H. Kallahalla, "Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering," *Sigcomm*, vol. 14, pp. 978–1, 2017.

[32] M. Franceschetti, O. Dousse, D. N. C. Tse, and P. Thiran, "Closing the Gap in the Capacity of Wireless Networks Via Percolation Theory," *IEEE Transactions on Information Theory*, vol. 53, no. 3, pp. 1009–1018, 03/2007.

[33] J. Ghaderi, L. L. Xie, and X. Shen, "Hierarchical cooperation in Ad hoc networks: Optimal clustering and achievable throughput," *IEEE Transactions on Information Theory*, vol. 55, no. 8, pp. 3425–3436, 2009.

[34] A. Raniwala, K. Gopalan, and T.-c. Chiueh, "Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 8, no. 2, p. 50, 04/2004.

[35] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," *Proceedings of the 10th annual international conference on Mobile computing and networking - MobiCom '04*, p. 114, 2004.

[36] D. S. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Wireless Networks*, vol. 11, no. 4, pp. 419–434, 2005.

[37] K. N. Ramachandran, E. M. Belding, K. C. Almeroth, and M. M. Buddhikot, "Interference-Aware Channel Assignment in Multi-Radio Wireless Mesh Networks," *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, vol. 00, no. c, pp. 1–12, 2006.

[38] M. M. Buddhikot, S. C. Miller, and A. P. Subramanian. (2013). Interference aware routing in multi-radio wireless mesh networks, [Online]. Available: https://www.google.com/patents/US8532023 (visited on).

[39] A. Subramanian, H. Gupta, S. Das, and Jing Cao, "Minimum Interference Channel Assignment in Multiradio Wireless Mesh Networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 12, pp. 1459–1473, 12/2008.

[40] D. Sajjadi, M. Tanha, and J. Pan, "A comparative study of channel switching latency for conventional and SDN-based routing in multi-hop multi-radio Wireless Mesh Networks," *2016 13th IEEE Annual Consumer Communications and Networking Conference, CCNC 2016*, pp. 330–334, 2016.

[41] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, 2003.

[42] A. G. Ganek and T. A. Corbi, "The dawning of the autonomic computing era," *IBM Systems Journal*, 2003.

[43] A. Trehan, "Algorithms for Self-Healing Networks," PhD thesis, University of New Mexico, 2010.

[44] J. Yu and P. Chong, "A survey of clustering schemes for mobile ad hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 1, pp. 32–48, 2005.

[45] E. M. Belding-Royer, "Hierarchical routing in ad hoc mobile networks," *Wireless Communications and Mobile Computing*, vol. 2, no. 5, pp. 515–532, 2002.

[46] Ying Ge, L. Lamont, and L. Villasenor, "Hierarchical OLSR - a scalable proactive routing protocol for heterogeneous ad hoc networks," in *WiMob'2005), IEEE International Conference on Wireless And Mobile Computing, Networking And Communications, 2005.*, vol. 3, IEEE, 2005, pp. 17–23.

[47] T. Clausen and P. Jacquet, Eds., *Optimized Link State Routing Protocol (OLSR)*, United States, 2003.

[48] S. Ganu, L. Raju, B. Anepu, S. Zhao, I. Seskar, and D. Raychaudhuri, "Architecture and prototyping of an 802.11-based self-organizing hierarchical ad-hoc wireless network (SOHAN)," in *2004 IEEE 15th International Symposium on Personal, Indoor and Mobile Radio Communications (IEEE Cat. No.04TH8754)*, IEEE, 2005, pp. 880–884.

[49] S. Zhao, I. Seskar, and D. Raychaudhuri, "Performance and scalability of self-organizing hierarchical ad hoc wireless networks," *2004 IEEE Wireless Communications and Networking Conference (IEEE Cat. No.04TH8733)*, pp. 132–137, 2004.

[50] X. Li, P. Djukic, and H. Zhang, "Zoning for hierarchical network optimization in software defined networks," *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pp. 1–8, 2014.

[51] C. a. Santiváñez, R. Ramanathan, and I. Stavrakakis, "Making link-state routing scale for ad hoc networks," *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing - MobiHoc '01*, p. 22, 2001.

[52] D. Wu and H. J. Chao, "Efficient bandwidth allocation and call admission control for VBR service using UPC parameters," *Proceedings - IEEE INFOCOM*, vol. 3, pp. 1044–1052, 1999.

[53] J. J. Garcia-Luna-Aceves and M. Spohn, "Source-tree routing in wireless networks," *Proceedings. Seventh International Conference on Network Protocols*, pp. 273–282, 1999.

[54] R. Ramanathan and M. Steenstrup, "Hierarchically-organized, multihop mobile wireless networks for quality-of-service support," *Mobile Networks and Applications*, 1998.

[55] A. Iwata, Ching-Chuan Chiang, Guangyu Pei, M. Gerla, and Tsu-Wei Chen, "Scalable routing strategies for ad hoc wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1369–1379, 1999.

[56] A. Santivfiez, B. Mcdonald, I. Stavrakakis, and R. Ramanathan, "On the Scalability of Ad Hoc Routing Protocols," *Infocom*, pp. 1688–1697, 2002.

[57] P. Dely, A. Kassler, and N. Bayer, "OpenFlow for Wireless Mesh Networks," in *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, IEEE, 07/2011, pp. 1–6.

[58] C. Putta and K. Prasad, "Performance of Ad Hoc Network Routing Protocols in IEEE 802.11," *International Conference on Computer and Communication Technology (ICCCT)*, pp. 371–376, 2010.

[59] J. Yi, A. Adnane, S. David, and B. Parrein, "Multipath optimized link state routing for mobile ad hoc networks," *Ad Hoc Networks*, vol. 9, no. 1, pp. 28–47, 2011.

[60] T. Uemori, E. Kohno, and Y. Kakuda, "A Routing ID-based Node-disjoint Multipath Scheme for Ad Hoc Networks," *9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing*, pp. 621–626, 12/2012.

[61] IS4CWN, "Next Steps for Community Wireless Networks," in *IS4CWN - International Summit for Community Wireless Networks*, 2010.

[62] Athens Wireless Metropolitan Network, "Athens Wireless Metropolitan Network," in *IS4CWN - International Summit for Community Wireless Networks*, 2010.

[63] P. D. Filippi. (2014). It's Time to Take Mesh Networks Seriously (And Not Just for the Reasons You Think), [Online]. Available: http://www.wired.com/2014/01/its-time-to-take-mesh-networks-seriously-and-not-just-for-the-reasons-you-think/ (visited on 04/12/2018).

[64] M. Adeyeye and P. Gardner-Stephen, "The Village Telco project: a reliable and practical wireless mesh telephony infrastructure," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 1, p. 78, 12/2011.

[65] D. Vega, L. Cerda-Alabern, L. Navarro, and R. Meseguer, "Topology patterns of a community network: Guifi.net," *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference*, pp. 612–619, 2012.

[66] C.-E. Bichot and P. Siarry, *Graph Partitioning*, C.-E. Bichot and P. Siarry, Eds., ser. ISTE. Wiley, 2013, p. 368.

[67] K. Andreev and H. Räcke, "Balanced graph partitioning," in *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures - SPAA '04*, New York, New York, USA: ACM Press, 2004, p. 120.

[68] M. Kubale, A. M. Society, and O. D. English, *Graph Colorings*, ser. Contemporary mathematics (American Mathematical Society) v. 352. American Mathematical Society, 2004.

[69] J. Evans. (2016). Mastering Chaos - A Netflix Guide to Microservices, [Online]. Available: https://www.infoq.com/presentations/netflix-chaos-microservices (visited on 04/09/2019).

[70] X. Jin, X. Li, H. Zhang, N. Foster, J. Lee, R. Soulé, C. Kim, and I. Stoica, "NetChain: Scale-Free Sub-RTT Coordination," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI '18)*, 2018.

[71] M. Gupta, J. Sommers, and P. Barford, "Fast, accurate simulation for SDN prototyping," *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*, p. 31, 2013.

[72] R. d. R. Fontes, M. Mahfoudi, W. Dabbous, T. Turletti, and C. Rothenberg, "How Far Can We Go? Towards Realistic Software-Defined Wireless Networking Experiments," *The Computer Journal*, pp. 1–14, 2017.

[73] M. Lacage and T. R. Henderson, "Yet another network simulator," in *Proceeding from the 2006 workshop on ns-2: the IP network simulator - WNS2 '06*, New York, New York, USA: ACM Press, 2006, p. 12.

[74] K. Fall, "Network emulation in the VINT/NS simulator," in *Proceedings IEEE International Symposium on Computers and Communications (Cat. No.PR00250)*, Red Sea, Egypt, Egypt: IEEE Comput. Soc, 1999, pp. 244–250.

[75] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "CORE: A Real-time Network Emulator," *IEEE Military Communications Conference (MILCOM)*, pp. 1–7, 11/2008.

[76] Y. Zheng, D. M. Nicol, D. Jin, and N. Tanaka, "A virtual time system for virtualization-based network emulations and simulations," *Journal of Simulation*, vol. 6, no. 3, pp. 205–213, 08/2012.

[77] D. Jin, Y. Zheng, H. Zhu, D. M. Nicol, and L. Winterrowd, "Virtual time integration of emulation and parallel simulation," *Proceedings - 2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation, PADS 2012*, pp. 201–210, 2012.

[78] Y. Zheng, D. Jin, and D. M. Nicol, "Impacts of application lookahead on distributed network emulation," in *Proceedings of the 2013 Winter Simulation Conference - Simulation: Making Decisions in a Complex World, WSC 2013*, 2013, pp. 2996–3007.

[79] D. L. Mills, "Network Time Protocol (NTP)," 1985.

[80] IEEE Standards Committee, "Precision clock synchronization protocol for networked measurement and control systems," *IEEE Std*, vol. 1588, 2004.

[81] K. S. Lee, H. Wang, V. Shrivastav, and H. Weatherspoon, "Globally synchronized time via datacenter networks," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ACM, 2016, pp. 454–467.

[82] Y. Geng, S. Liu, Z. Yin, A. Naik, B. Prabhakar, M. Rosenblum, and A. Vahdat, "Exploiting a Natural Network Effect for Scalable, Fine-grained Clock Synchronization," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, Renton, WA: {USENIX} Association, 2018, pp. 81–94.

[83] E. Weing, F. Schmidt, H. Lehn, T. Heer, and K. Wehrle, "SliceTime : A platform for scalable and accurate network emulation," in *NSDI*, 2011.

[84] H. Tazaki, F. Uarbani, E. Mancini, M. Lacage, D. Camara, T. Turletti, and W. Dabbous, "Direct code execution," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies - CoNEXT '13*, New York, New York, USA: ACM Press, 2013, pp. 217–228.

[85] J. Lamps, V. Adam, D. M. Nicol, and M. Caesar, "Conjoining Emulation and Network Simulators on Linux Multiprocessors," in *Proceedings of the 3rd ACM Conference on SIGSIM-Principles of Advanced Discrete Simulation - SIGSIM-PADS '15*, ser. SIGSIM PADS '15, New York, New York, USA: ACM Press, 2015, pp. 113–124.

[86] H. Fontes, T. Cardoso, and M. Ricardo, "Improving Ns-3 Emulation Performance for Fast Prototyping of Network Protocols," in *Proceedings of the Workshop on Ns-3*, ser. WNS3 '16, New York, NY, USA: ACM, 2016, pp. 108–115.

[87] D. Gupta, K. Yocum, M. McNett, A. C. Snoeren, A. Vahdat, and G. M. Voelker, "To infinity and beyond: time warped network emulation," in *NSDI*, 2005.

[88] J. Lamps, D. M. Nicol, and M. Caesar, "TimeKeeper: A Lightweight Virtual Time System for Linux," in *Proceedings of the 2nd ACM SIGSIM/PADS conference on Principles of advanced discrete simulation - SIGSIM-PADS '14*, New York, New York, USA: ACM Press, 2014, pp. 179–186.

[89] J. Yan and D. Jin, "VT-Mininet: Virtual-time-enabled Mininet for Scalable and Accurate Software-Define Network Emulation Categories and Subject Descriptors," *ACM SIGCOMM Symposium on Software Defined Networking Research - SOSR*, 27:1–27:7, 2015.

[90] D. P. Wiggins, L. Veytser, P. Deutsch, B.-n. Cheng, and W. Street, "Scaling NS-3 DCE Experiments on Multi-Core Servers," MIT Lincoln Laboratory, Lexington, MA, Tech. Rep., 2016.

[91] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies - CoNEXT '12*, New York, New York, USA: ACM Press, 2012, p. 253.

[92] B. Heller, "REPRODUCIBLE NETWORK RESEARCH WITH HIGH-FIDELITY EMULATION," PhD thesis, Stanford University, 2013.

[93] N. Simulation, *Modeling and Tools for Network Simulation*, K. Wehrle, M. Güne? And J. Gross, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[94] L. Foundation. (2017). Linux Namespaces, [Online]. Available: http://man7.org/linux/man-pages/man7/namespaces.7.html (visited on 01/01/2017).

[95] Message Passing Forum, "MPI: A Message-Passing Interface Standard," Knoxville, TN, USA, Tech. Rep., 1994.

[96] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[97] C. Hill, "Matplotlib," in *Learning Scientific Programming with Python*, Cambridge: Cambridge University Press, 2016, pp. 280–332.

[98] Microsoft Inc. (2018). US Building Footprints, [Online]. Available: https://github.com/microsoft/USBuildingFootprints (visited on 07/01/2019).

[99] (2012). Global Administrative Areas, [Online]. Available: http://www.gadm.org/home (visited on 01/30/2020).

[100] K. Jordahl, J. V. den Bossche, J. Wasserman, J. McBride, M. Fleischmann, J. Gerard, J. Tratner, M. Perry, C. Farmer, G. A. Hjelle, S. Gillies, M. Cochran, M. Bartos, L. Culbertson, N. Eubank, maxalbert, S. Rey, A. Bilogur, D. Arribas-Bel, C. Ren, J. Wilson, M. Journois, L. J. Wolf, L. Wasser, A. D. Snow, YuichiNotoya, F. Leblanc, Filipe, C. Holdgraf, and A. Greenhall, "Geopandas/geopandas: V0.6.2," version v0.6.2, 11/2019.

[101] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, 2008.

[102] M. Z. Al-Taie and S. Kadry, "Network basics," in *Advanced Information and Knowledge Processing*, 2017.

[103] M. J. Rochkind, *Advanced UNIX Programming (2nd Edition)*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 2004.

[104] M. Casoni and N. Patriciello, "Next-generation TCP for ns-3 simulator," *Simulation Modelling Practice and Theory*, vol. 66, pp. 81–93, 08/2016.

[105] P. Gupta, R. Gray, and P. R. Kumar, "An Experimental Scaling Law for Ad Hoc Networks," Tech. Rep., 2001.

[106] M. Garetto, T. Salonidis, and E. W. Knightly, *Modeling per-flow throughput and capturing starvation in CSMA multi-hop wireless networks*, 2008.

[107] C. Hua and R. Zheng, "Starvation Modeling and Identification in Dense 802.11 Wireless Community Networks," *Communications Society*, pp. 1696–1704, 2008.

[108] E. Jones, T. Oliphant, P. Peterson, *et al.* (2001). SciPy: Open source scientific tools for Python, [Online]. Available: http://www.scipy.org/ (visited on 09/10/2018).

[109] S. Gramacho, F. Gramacho, and A. Wildani, "Autonomic Partitioning for the Smart Control of Wireless Mesh Networks," in *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 10/2019, pp. 175–182.

[110] D. Ongaro and J. K. Ousterhout, "In search of an understandable consensus algorithm," in *USENIX Annual Technical Conference*, 2014, pp. 305–319.

[111] IEEE Computer Society, *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*. New York, NY, US: IEEE, 2012.

[112] L. Lamport, "Paxos Made Simple," *ACM SIGACT News*, vol. 32, no. 4, pp. 51–58, 2001.

[113] M. Burrows, "The Chubby lock service for loosely-coupled distributed systems," in *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI '06)*, 2006.

[114] P. Hunt, M. Konar, F. Junqueira, and B. Reed, "ZooKeeper: Wait-free Coordination for Internet-scale Systems.," *USENIX*, 2010.

[115] A. Ailijiang, A. Charapko, and M. Demirbas, "Consensus in the Cloud: Paxos Systems Demystified," in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, IEEE, 08/2016, pp. 1–10.

[116] S. Gramacho, F. Gramacho, and A. Wildani, "Autonomic Formation of Large-Scale Wireless Mesh Networks," in *NetSoft 2020*, Ghent, BE, 2020.

[117] I. V. Brito, S. Gramacho, I. Ferreira, M. Nazaré, L. Sampaio, and G. B. Figueiredo, "OpenWiMesh: a Framework for Software Defined Wireless Mesh Networks," in *Proceeedings of the 32nd SBRC*, Florianópolis, SC, BR, 2014.

[118] M. et al. (2015). POX Documentation, [Online]. Available: https://noxrepo.github.io/pox-doc/html/ (visited on 09/10/2018).

[119] P. Djukic and P. Mohapatra, "Soft-TDMAC: A software TDMA-based MAC over commodity 802.11 hardware," *Proceedings - IEEE INFOCOM*, pp. 1836–1844, 2009.

[120] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 12/1959.

[121] P. Bosshart, G. Varghese, D. Walker, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, and A. Vahdat, "P4: programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 07/2014.

[122] M. E. J. Newman, "The mathematics of networks," *The New Palgrave Encyclopedia of Economics*, vol. 2, pp. 1–12, 2007.

[123] U. Brandes, "On variants of shortest-path betweenness centrality and their generic computation," *Social Networks*, vol. 30, no. 2, pp. 136–145, 2008.

[124] ——, "A faster algorithm for betweenness centrality," *The Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[125] U. Brandes and D. Fleischer, "Centrality measures based on current flow," *Stacs 2005*, pp. 533–544, 2005.

[126] A. Andreyev, X. Wang, and A. Eckert. (2019). Reinventing Facebook's data center network, [Online]. Available: https://engineering.fb.com/data-center-engineering/f16-minipack/ (visited on 02/11/2020).

[127] N. Al-Falahy and O. Y. Alani, "Technologies for 5G Networks: Challenges and Opportunities," *IT Professional*, vol. 19, no. 1, pp. 12–20, 01/2017.

[128] Steve Song, "Village Telco," in *IS4CWN - International Summit for Community Wireless Networks*, Vienna, 2010.

[129] S. Song. (2016). MP2 AWD – 'All Wheel Drive' Edition, [Online]. Available: https://villagetelco.org/2016/11/mp2-awd-all-wheel-drive-edition/ (visited on 08/16/2018).

[130] K. Williamson. (2014). Introducing Wildernets, [Online]. Available: https://villagetelco.org/2014/09/introducing-wildernets/ (visited on 08/20/2018).

[131] World Possible. (2018). WORLD POSSIBLE - Connecting offline learners to the world's knowledge, [Online]. Available: https://worldpossible.org/ (visited on 08/20/2018).

[132] P. Gardner-Stephen, R. Challans, J. Lakeman, A. Bettison, D. Gardner-Stephen, and M. Lloyd, "The serval mesh: A platform for resilient communications in disaster & crisis," in *2013 IEEE Global Humanitarian Technology Conference (GHTC)*, IEEE, 10/2013, pp. 162–166.

[133] P. Gardner-Stephen, "The Serval Project - Making Telecommunications Available Anywhere and Anytime," in *IS4CWN - International Summit for Community Wireless Networks*, Vienna, 2010.

[134] Gardner-Stephen, "The rational behind the serval network layer for resilient communications," *Journal of Computer Science*, vol. 9, no. 12, pp. 1680–1685, 12/2013.

[135] VeoEscuchoOpino. (2014). Serval Project para Venezuela, [Online]. Available: https://www.youtube.com/watch?v=TthP8IBdjao (visited on 08/16/2018).

[136] P. Gardner-Stephen. (2013). Serval Mesh 0.90 Multi-Hop MeshMS & Rhizome, [Online]. Available: https://www.youtube.com/watch?v=u30KA7fk3v0 (visited on 08/27/2018).

[137] R. Baig, R. Roca, F. Freitag, and L. Navarro, "Guifi.net, a crowdsourced network infrastructure held in common," *Computer Networks*, vol. 90, pp. 150–165, 2015.