UNIVERSIDADE D
COIMBRA

Joel Möllering Torrado

MODELING AND ANALYSIS OF ENERGY HARVESTING IOT NETWORKS

Dissertação de Mestrado na área científica de Engenharia Electrotécnica e de Computadores orientada pelo Senhor Professor José Luís Esteves dos Santos e apresentada ao Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Setembro de 2018

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Modeling and Analysis of Energy Harvesting IoT Networks

Joel Möllering Torrado

Coimbra, September 2018

Modeling and Analysis of Energy Harvesting IoT Networks

Supervisor:
Prof. Dr. José Luís Esteves dos Santos

Co-Supervisor:
Prof. Dr. Marilia Pascoal Curado

Jury:
Prof. Dr. Paulo José Monteiro Peixoto
Prof. Dr. Vasco Nuno Sousa Simões Pereira
Prof. Dr. José Luís Esteves dos Santos

Dissertation submitted in partial fulfillment for the degree of Master of Science in Electrical and Computer Engineering.

Coimbra, September 2018

Acknowledgments

Acredito que uma pessoa perfaz-se enquanto pessoa com a ajuda de outros com os quais se cruza na caminhada da vida. Eu não sou excepção e portanto sou o que sou hoje e cheguei até aqui devido ao facto de contar com a vossa companhia, camaradagem, afecto e ajuda em algum dos momentos da minha vida.

Assim, começo por escrever o meu apreço a todos aqueles a que eu posso chamar de colegas – que ainda são bastantes –, desde a escola primária em Aldeia de S. Sebastião, à Escola de Vilar Formoso, passando pela Infanta D. Maria em Coimbra, até à Universidade. Todos vocês contribuíram de alguma forma ou outra através das vossas opiniões e amizades.

Nessas andanças estudantis, quero dar um destaque à malta da Sagrada Casa dos Gabirus, da Orxestra Pitagórica (EAMT2B!) e do BEST Coimbra, pelas aventuras partilhadas e pelas histórias que ficarão sempre para contar. E ainda ao pessoal do G.6.7 pelos conselhos e camaradagem.

Aos meus amigos da Anarkia, mas principalmente aos meus grandes amigos do Set'Up pela vossa amizade ao longo destes anos todos. Que haja muitos mais! E que todos nós atinjamos o sucesso que nos é devido.

À minha família, em particular à minha avó Graça e ao meu avô Adelino (já falecido, que em paz descanse) por terem acreditado sempre em mim desde que era pequeno, e por me mostrarem o valor da humildade.

Ao meu irmão Guilherme, que recorrentemente me prova que o reconhecimento enquanto pessoa advém do trabalho e da dedicação que é imposta a tudo o que uma pessoa se dispõe a fazer.

Aos meus pais, por serem o grande suporte que sempre foram e que hoje me faz ter os valores que acredito serem importantes para singrar no futuro. Em detalhe, ao meu pai Jan pela confiança depositada em mim e pela ajuda nas horas de necessidade. À minha mãe Lina por tudo o que teve de endurar pelo melhor para mim, pelo carinho, afecto, conselhos e conversas; a sua forma de ser faz-me crer que o Mundo ainda tem espaço para as pessoas boas.

Por último, um especial agradecimento à Rita. Por todos os bons momentos passados em conjunto. Pelo imenso apoio, carinho e amor que alguém consegue proporcionar em todos os instantes e por conseguir sempre fazer de mim uma pessoa melhor. Por um futuro feliz, que será até quando nós quisermos.

Por tudo, obrigado.

The work team that contributed to this dissertation is constituted by André Riker (Laboratory of Communications and Telematics (LCT) – Center for Informatics and Systems of the University of Coimbra (CISUC)), Prof. Dr. José Luís dos Santos (Center for Mathematics of the University of Coimbra (CMUC)), and Prof. Dr. Marilia Curado (Laboratory of Communications and Telematics – Center for Informatics and Systems of the University of Coimbra). Let me express here my gratitude towards them for their availability, suggestions of improvements, and overall contributions to this work.

The work presented in this dissertation was partially carried out in the scope of the MobiWise project: From mobile sensing to mobility advising (P2020 SAICTPAC/0011/2015), co-financed by COMPETE 2020, Portugal 2020 - Operational Program for Competitiveness and Internationalization (POCI), European Union's ERDF (European Regional Development Fund), and the Portuguese Foundation for Science and Technology (FCT).

UNIÃO EUROPEIA
Fundo Europeu
de Desenvolvimento Regional

Resumo

Redes de Internet das Coisas em que existe recolha de energia do ambiente são o alicerce de uma série de conceitos de alta conectividade contemporâneos nos quais é necessário extrair cada vez mais informação do ambiente, ao mesmo tempo que se mantém a perpetuidade operacional dos mesmos.

Maximizar a recolha de dados do meio envolvente usando uma rede deste tipo com conectividade multi-etapa enquanto se mantém uma operação energética neutra é uma tarefa complexa pois é necessário equilibrar a utilidade global da rede e a escassez de recursos energéticos.

Para concretizar uma operação óptima de uma rede de Internet das Coisas com recolha de energia do ambiente recorre-se à construção de um modelo de optimização baseado em Programação Linear Inteira Mista, em que são considerados dois aspectos adicionais: aquisição obrigatória de ocorrências notáveis ao longo do tempo e do espaço, denominadas de eventos; e a transmissão de dados deve utilizar um mecanismo de agregação de dados, de forma a tornar a comunicação mais eficiente.

Para que o modelo proposto seja validado e analisado adequadamente, é desenvolvida uma plataforma que permite efectuar simulações com o modelo, as quais têm como parâmetros de entrada dados que refletem condições operacionais estudadas. Com base nos resultados das simulações realizadas, constata-se que o modelo apresentado é bem sucedido nos vários cenários experimentais considerados.

Os resultados obtidos confirmam a validade do modelo proposto. Os dados recolhidos permitem identificar padrões no comportamento da rede que possibilitam a construção de uma heurística distribuída sub-óptima que atinja um desempenho semelhante em tempo real.

Palavras-chave – Internet das Coisas, Recolha de Energia do Ambiente, Operação Energética Neutra, Programação Linear Inteira Mista

Abstract

Energy Harvesting Internet of Things Networks serve nowadays as the backbone of a myriad of high-connectivity concepts in where it is necessary to extract more and more information from the environment while maintaining a perpetual operational state.

Maximizing the collection of environmental data employing a multi-hop Energy Harvesting IoT Network while maintaining an Energy Neutral Operation is a challenging task because it is necessary to balance overall network utility with the scarcity of energy resources, typical of Energy Harvesting techniques.

A Mixed-Integer Linear Programming optimization model is constructed to achieve an optimal operation in an Energy Harvesting IoT Network. Two additional aspects are considered: enforced acquisition of spatio-temporal notable occurrences labeled as events; and the use of a data aggregation mechanism when performing data communication.

A framework is developed which can perform simulations with the proposed model so it can be validated and appropriately analyzed. The model simulations are supplied with input data that mirrors a studied operational setting. Based on the simulations results, the effectiveness of the presented network model is derived under different operational schemes.

The obtained results confirm the validity of the proposed model. The gathered data facilitates the identification of patterns in the network behavior, which enable the formulation of a distributed real-time sub-optimal heuristic that achieves a similar performance.

Keywords – Internet of Things, Energy Harvesting, Energy Neutral Operation, Mixed-Integer Linear Programming

"Faz porque queres e sentes. Não porque deves e tens."

— Samuel Mira, *À Procura da Perfeita Repetição*

Contents

List of Acronyms

2D	**bi-dimensional**
6LowPan	**IPv6 over Low-Power Wireless Personal Area Networks**
B&B	**Branch and Bound**
CoAP	**Constrained Application Protocol**
CDMA	**Code Division Multiple Access**
CISUC	**Center for Informatics and Systems of the University of Coimbra**
CMUC	**Center for Mathematics of the University of Coimbra**
EH-IoTN	**Energy Harvesting IoT Network**
EH-WSN	**Energy Harvesting Wireless Sensor Network**
ENO	**Energy Neutral Operation**
FDMA	**Frequency Division Multiple Access**
GHI	**Global Horizontal Irradiation**
GUI	**Graphical User Interface**
IEEE	**Institute of Electrical and Electronics Engineers**
IDE	**Integrated Development Environment**
IP	**Integer Program(ming)**
IoT	**Internet of Things**
LCT	**Laboratory of Communications and Telematics**

LP	**Linear Program(ming)**
MILP	**Mixed-Integer Linear Program(ming)**
MIP	**Mixed-Integer Program(ming)**
NetVis	**Network Visualizer app**
OPL	**Optimization Programming Language**
SRAM	**Static Random-Access Memory**
RAM	**Random-Access Memory**
UDP	**User Datagram Protocol**
WSN	**Wireless Sensor Network**

List of Figures

List of Tables

List of Equations

1 Introduction

This chapter summarizes the motivations that led to this dissertation, compiles the primary goals and the major contributions, and provides an outline of this document to contextualize the reader.

1.1 Motivation and Context

The Internet of Things (IoT) is the network that interconnects physical objects to the Internet, which serves society in numerous ways by providing ample and assorted applications [1]. In IoT, a Wireless Sensor Network (WSN) takes an active role in the consolidation of physical world information. WSNs incorporate a significant number of tiny devices called sensor nodes which are deployed in the sensing domain of interest to gather its data. That is why in recent years there has been a remarkable focus on achieving more and more connected objects. The most significant restrictions to effective adoption of IoT systems have been the limited deployment possibilities if they operate on grid power or the bound operational lifetime if running on conventional batteries [1].

Energy harvesting from light, thermal or mechanical sources available in the environment provides the electrical energy required to power these devices, that is, when paired with a rechargeable battery [2].

A definite interest is to use the available energy efficiently because in networked systems with multiple harvesting nodes, these may have varying harvesting opportunities. Also, in a distributed application, the same performance may be accomplished employing mixed workload[1] allocations, which results in diverse consumptions at multiple nodes. Under these circumstances, it is essential to adjust the workload allocation with their energy availability.

Because both energy harvesting and in-network data processing are major IoT techniques, it is crucial to analyze them concurrently. This dissertation's work tackles the problem of

[1] Workload: the amount of performed work

controlling the mechanisms related to data-production and data-communication to achieve an Energy Neutral Operation (ENO) in a multi-hop[2] Energy Harvesting IoT Network (EH-IoTN).

The term ENO was introduced by Kansal *et al.* [4] as a mode of operation of an IoT node where its energy consumption is at all moments inferior or equal to the energy harvested from the environment. To achieve this kind of operation on a data-gathering IoT network is a very complex task, since the nodes do not have full knowledge of the other nodes status and their operational conditions can differ.

To achieve an optimal solution to the problem above it is proposed a mathematically constrained optimization model. The central challenge of this work was to construct, evolve and validate such proposed model, so the network behaviors revealed by the model solutions were in concordance with the expected performance of an IoT network. Therefore, an accompanying graphical tool was also developed in the scope of this dissertation to support the validation of the provided solutions.

This research project paves the way for further evolutions of the considered model and ambitious sub-optimal real-time heuristic approaches. The model and the created tools can be replicated and used in LCT for other researching prospects.

1.2 Goals

This work purpose is to present both theoretical and practical studies on how to optimize global performance in EH-IoTNs by in-network data gathering and transfer while also guaranteeing Energy Neutral Operation on the whole network.

The primary goals are reviewed as follows:

1. Study the major characteristics and approaches in networking algorithms for energy-efficient self-managed networks and define what the proposed model should be capable of;
2. Implement the intended mathematically constrained optimization model such that it is possible to extract actual results from it;
3. Provide a way to easily interact with the test parameters of the model and its results;

[2]Multi-hop routing is a type of communication in radio networks in which network coverage area is larger than radio range of single nodes. Therefore, to reach some destination a node can use other nodes as relays. [3]

4. With a comprehensive set of model simulation results, verify that the network model behavior is as expected from point 1 and that it can be used to formulate a heuristic sub-optimal model posteriorly.

1.3 Key Contributions

The achievements fulfilled throughout this work, from which some of them double up as implementations, are reviewed below:

- A refined mathematically constrained model based on previous work of the team presented in the `Acknowledgments`, that is described in Chapter 4.
- **Implementation** – An Optimization Programming Language (OPL) model based on the previously referred model that can be solved by the CPLEX engine provided by the IBM ILOG CPLEX Optimization Studio software[3]. The resulting code can be found in an open-access online repository[4].
- **Implementation** – A MATLAB Graphical User Interface (GUI) that performs as the testing platform of the OPL model, as a results validator, and as a results data explorer. This platform is described in Chapter 5, and the code can be found in an open-access online repository[5].
- **Implementation** – A model developing and testing framework that comprises both of the implementations above (OPL and MATLAB code) and a way to pass data between the two efficiently.
- An extensive amount of data from the simulations performed with variations in the input parameters – to better understand the model performance – that led to some important conclusions.

Beyond that, it is expected to target the following publications with the outcomes of the produced work in this dissertation:

- `IEEE Internet of Things Journal` with the proposed model and subsequent heuristic;
- `15th Wireless On-demand Network systems and Services Conference` with the developed **NetVis** framework and a simple version of the implemented model.

[3] `https://www.ibm.com/products/ilog-cplex-optimization-studio`
[4] `https://bitbucket.org/mollering/mobiwise-cplex`
[5] `https://bitbucket.org/mollering/mobiwise-matlab`

1.4 Structure of Dissertation

This dissertation is divided into eight chapters.

The current chapter provides the motivations, overall goals and expected contributions of this work.

The related research and the background for this work is presented in **Chapter 2**.

In **Chapter 3**, the theoretical model that is the premise of this work is analyzed and then, in **Chapter 4**, the mathematical implementation of such model – that is the core of this work – is thoroughly described.

The overall aspects of the implemented software used to validate the produced model are provided in **Chapter 5**. In **Chapter 6**, it is explained the parameters fed to that software to validate the model.

The achieved results and additional discussion are exhibited in **Chapter 6**.

Finally, in **Chapter 7** we will draw conclusions, and following that, in the same chapter, we suggest future work.

Additional material can be found in appendices: **Appendix A** exposes assumed considerations for data used in the model evaluation, and **Appendix B** is the user's guide for the software described in Chapter 5.

2 Background

This chapter expresses an overall examination of the field revolving around Energy Harvesting IoT Networks while contextualizing with relevant academic works.

2.1 Concept

Energy Harvesting Wireless Sensor Networks (EH-WSNs) attracted enormous interest in the past two decades [5, 6, 7]. The combined capabilities of detecting and reporting the temporal and spatial dynamics from the environment plus maintaining an operational state through energy harvesting from it are crucial for many network design solutions. These abilities are particularly useful in autonomous operations such as environmental monitoring tasks – hazardous (earthquakes, volcanoes, fires) [8, 9, 10] and non-hazardous (weather, habitats, machinery) [6, 11, 12] – structural monitoring [13, 14, 15], personal or animal tracking [16, 17, 18], and wireless robot systems [19].

Energy Harvesting IoT Networks are the next step of the technological evolution in this field: they combine the almost-perpetual sensing abilities of EH-WSNs with the capacity of actuating on the environment. EH-IoTNs perform a crucial part in achieving the ambitions of many rising concepts, like Smart Cities, Smart Grids, and Industrial Internet of Things. Considerable numbers of small, battery-operated sensing and actuating devices can be deployed in all sorts of environments to gather data, propagate information and execute actions [20].

The concept of energy harvesting releases the network devices from having an *always on* energy infrastructure and grants a way of operating the network with a potentially perpetual lifetime. Other advantages arise, such as reduced operation cost [21], environment-friendliness [21, 22], and in many cases may be the only option available due to practical constraints [8, 23, 24]. The most common sources of the harvested energy for EH-IoTNs are:

- **Solar** – Photovoltaic harvesting systems for uninterrupted EH-IoTNs operations should

be simplistic, robust, and operate without human mediation for long periods of time [25, 26, 27]. The nature of solar power makes it abundant and accessible, but highly dynamic: a few hours of daily direct sunlight are expectable in most regions, and there is the likelihood of existing heterogeneous harvesting across distinct devices in a sensing space due to shading or cloud coverage.

- **Kinetic** – Commonly the most versatile and omnipresent ambient energy source available [28]. Especially prominent in bridges, roads and rail tracks. Motion source is usually machine vibration [13, 28, 29] and fluid flow (e.g., wind or water currents) [24, 30]. Successful exploitation possibly requires the combined design of the entire wireless system, including a power-aware control of the powered device.
- **Thermal** – Thermogenerators utilize the temperature variations or gradients to generate electricity, e.g., between the human body and the surrounding environment [31, 32, 33]. Due to the absence of moving pieces in thermal energy harvesting devices, they tend to be more resilient than kinetic-based devices.
- **Wireless Energy Transfer** – Objects resonating at an equivalent frequency tend to exchange energy efficiently while scattering comparatively little energy in external off-resonant objects. This way, it is possible to harvest radiation from multiple power sources [27, 34, 35].

Energy harvesting is a gradual process, its rate usually is insufficient to support a constant performance of typically small devices, and is inherently time varying [2, 36, 37]. Hence, powering sensor networks and associated actuators with energy harvesting technology presents essential design and optimization challenges, which have captivated a considerable research interest in recent years [2, 36, 37]. To improve energy efficiency it is required to design optimal activation policies and scheduling algorithms for the sensors, so it is possible to maximize the entire network performance under energy allocation constraints.

2.2 Related Work

Most current EH-IoTNs related works focus on presenting both theoretical and practical studies on how to optimize network performance achieving sustainable operation – i.e., no sensor node runs out of energy – through one or various of the following topics:

- **Sensing and networking algorithm design** – Network-wide protocols that estimate the energy spent by sensors plus wireless transceivers for sensing and data communica-

tions respectively. They can be distributed and adaptative to accomplish autonomous multi-hop data acquisition [38, 39, 40].

- **In-network data processing** – An efficient approach to diminish bandwidth consumption, where data is processed and reduced inside the network [41]. Common in-network processing methods comprise raw sensor data compression, aggregation, fusion, and feature extraction [42, 43, 44].
- **Power management schemes** – Self-management capacities of every single node that optimizes its long-term harvesting power usage [25, 45, 46].
- **Path traveling optimization** – Routing protocols that adaptively determine and dynamically adapt the end-to-end path to bypass the time-varying routing hot-spots (i.e., nodes with low battery) [47, 48]. Multi-hop networks can turn this optimization unmanageable [49, 50].
- **Energy awareness** – Constructing realistic models for energy storage (i.e., rechargeable battery with finite capacity), energy harvesting, and operational energy consumption states [30, 51, 52].

One of the most usual approaches to achieve validity on proposed solutions encompassing the topics mentioned above is to formulate an optimization problem, and next propose an algorithm which can solve that problem. Finally, through testing, it is proven that the stated algorithm achieves the desired network operation and utility performance within some given bounds. Similarly to this work, some of these solutions (see Table 2.1) propose routing mechanisms and operations scheduling such as to maximize aggregated network utility while dealing with performance constraints.

Features	[53]	[54]	[55]	[13]	[25]	This Work
Energy Neutral Operation	no	yes	no	yes	no	yes
Event Capture	yes	no	no	no	yes	yes
Data Aggregation	no	no	yes	no	no	yes
Multi-hop Network	yes	no	yes	yes	yes	yes

Table 2.1: Related works features differentiation

Zareei *et al.* (2018) [53] propose a protocol for EH-IoTNs that uses adaptive transmission power to enhance the overall performance of the network. The core idea is to extend the transmission reach of the nodes that have elevated energy levels when their neighbor nodes

possess inferior energy levels, while attempting to avoid that nodes in critical energy conditions perform communications. Although this proposal improves the end to end performance of a multi-hop network while maintaining a regular event sensing rate, it allows nodes to reach critical energy conditions – it does not ensure an Energy Neutral Operation – and the data transmissions do not consider any compression or data aggregation aspect.

Jackson *et al.* (2017) [54] formulate an optimization problem to maximize the operational activity and battery lifetime of each network node while acknowledging the degradation of the battery capacity. They achieve a solution to this problem through a proposed algorithm that can be used by an arbitrary IoT application. This work focuses highly in maintaining an Energy Neutral Operation of all the network nodes but does not refer any multi-hop routing scheme, where data can flow in many possible transmission routes. Data aggregation and the existence of particular events are also not mentioned.

Yang *et al.* (2016) [55] present an online algorithm to solve a stochastic optimization problem based on practical energy and network models. Their formulations influenced this work significantly, as they use in-network data processing (e.g., data aggregation/compression/fusion) and the concept of sustainable network operation in their models. However, it is explicitly indicated that this sustainable operation is that no sensor node runs out of energy; thus it can not be considered ENO. The extra sensing restrictions imposed by the existence of environmental events are also not considered.

Gaglione, Rodenas-Herraiz *et al.* (2018) [13] characterize a multi-hop vibration EH-WSN system for utilization in bridges. The evaluations performed in this paper demonstrate that their system achieves Energy Neutral Operation in a 4-hop sensor network. Although it is noticed that the vehicle traffic in bridges is transient in nature – as in events –, this information is applied to the energy harvesting model and not to the data utility performance. Data aggregation is not applied in this work.

Dehwah *et al.* (2017) [25] have as objective to find the best network policy to optimize the remaining energy in a solar-powered wireless sensor network to monitor flood occurrences. They achieve good practical results with management policies in a multi-hop routing scheme that takes into account the existence of events (i.e., floods) and energy maximization. This maximization can be seen as a sort of Energy Neutral Operation, but it is really not well defined here. This work assumes that no data compression or aggregation is available.

2.3 Research Scope

The work presented in this dissertation follows a similar line of thought as the works presented in Section 2.2. Previous related work performed at CISUC such as from Riker *et al.* (2015, 2017) [44, 56] and by other sources, such as Yang *et al.* (2016) [55] and Mehrabi *et al.* (2017) [57], lead to the design of a concept model which purpose is to maximize the aggregate data traffic utility of a **multi-hop** EH-IoTN into a single *sink node*.

For this goal, we study the problem of composing an optimal scheduling algorithm in a discrete finite-horizon network. Finite capacity batteries power the nodes; still, they are capable of harvesting energy.

Allow us to reflect throughout the topics specified in Section 2.2. Every *time slot*, the network settles how much new data to admit through sensing and how much power to allocate over each communication link for data transmission *(sensing and networking algorithm design; power management schemes)*. The sensing operational allocation is further constrained by the mandatory capture of stochastic spatio-temporal occurrences designated as **events** *(sensing and networking algorithm design)*. Besides, energy consumption for data transmissions can sometimes be minimized by employing a **data aggregation** mechanism, where several data payloads can be integrated into the same data packet *(in-network data processing; power management schemes)*.

The network operation is subject to an **Energy Neutral Operation**, where the average power consumption of the nodes is less or equal than the average power harvested from the environment *(power management schemes)*. The energy availability constraint dramatically complicates the operation of an efficient scheduling algorithm, since the current energy expenditure decision may cause an energy outage in the future and consequently affect the subsequent decisions.

The concept of *path traveling optimization* is also explored implicitly by the optimization model results. As for *energy awareness*, a considerable effort was put into having the most realistic energy profiles as possible (see Appendix A).

The finite-horizon problem is essential but challenging and turns out to be highly sophisticated. The complexity arises from the fact that it compels to optimizing metrics over the short term rather than metrics that are averaged over an extended period. There is also the time coupling property, which implies that current decisions can influence future

performance.

It is possible to infer from the overall description above (and from Table 2.1) that this work studies jointly these four important IoT techniques: **Energy Neutral Operation** of a **multi-hop network** with **data aggregation** and environmental **events capture**. To the best of our knowledge, this quartet optimization has not been studied yet.

It is not intended with this work to claim that the obtained results are suitable to implement the evaluated algorithms in a real EH-IoTN setting, at least in the present form. This work focuses on obtaining behavioral patterns in the operation of EH-IoTNs through time, so a solid theoretical foundation supports further algorithmic research to be able to mimic the extracted optimal behaviors into a heuristic distributed setting.

3 Theoretical Model

We analyze an energy harvesting network that consists of a set of statically-deployed nodes $\mathcal{N}$ that includes sensor nodes $\mathcal{S}$ and only one IoT gateway (*sink node*) $\odot$ such that $\mathcal{N} = \mathcal{S} \cup \odot$. The network can be expressed as a graph $\mathbf{G}(\mathcal{N}, \mathcal{L})$ with arbitrary topologies, where $\mathcal{L}$ represents the set of all possible wireless links. These links are directed, though we assume connectivity to be symmetric, i.e., link $(j, i) \in \mathcal{L}$ if and only if $(i, j) \in \mathcal{L}$. Let $N \geq 2$ be the order of the graph (its number of nodes), i.e., $N = |\mathcal{N}|$; and let $L \geq 1$ be the size of the graph (its number of links), i.e., $L = |\mathcal{L}|$. Let $\mathcal{N}_i \subseteq \mathcal{N}$ be the set of all one-hop neighbors of each node $i \in \mathcal{N}$. See visual examples of a network in Figure B.4.

The network operates in a finite-horizon period consisting of discrete *time slots* $t \in \{0, 1, 2, \ldots, T\}, T < \infty$, where $t = 0$ is the initial state of the network (with associated start conditions). Each *time slot* has a fixed time span Δ where all the operations within the scope of the states below can be performed in its entirety, i.e., all operations that need to be finished in one *time slot* have enough time to end before starting a new *time slot*.

In a particular *time slot*, there are four possible states $\mathcal{M}$ for a sensor node: staying idle (sleeping) $\mathtt{z}$, collecting raw sensor readings (sensing) $\mathtt{s}$, transmitting $\mathtt{Tx}$ or receiving $\mathtt{Rx}$ *data packets* over a wireless link. This means $\mathcal{M} = \{\mathtt{z}, \mathtt{s}, \mathtt{Tx}, \mathtt{Rx}\}$. A node can only be in one operational state at each *time slot*, i.e.,

$$\sum_{\mathbf{m} \in \mathcal{M}} S_i^{\mathbf{m}}(t) = S_i^{\mathtt{z}}(t) + S_i^{\mathtt{s}}(t) + S_i^{\mathtt{Tx}}(t) + S_i^{\mathtt{Rx}}(t) = 1, \quad \forall i \in \mathcal{N}, \;\; 1 \leq t \leq T \tag{3.1}$$

where $S_i^{\mathbf{m}}(t)$ is a binary variable that indicates if the sensor node i at *time slot* t is in the state $\mathbf{m} \in \mathcal{M}$ or not, i.e., $S_i^{\mathbf{m}}(t) = 1$ and $S_i^{\mathbf{m}}(t) = 0$ respectively.

In our model, the sink node $\odot$ does not have any battery limitations and has virtually infinite *buffer* size.

Additionally, all the sensor nodes have harvesting devices and batteries with the same characteristics. This implies that all the nodes are in the same initial technical conditions.

3.1 Data and Communication Models

1) Wireless Interference Model: We consider that the network has primary interference: links that share a common node cannot transmit and receive simultaneously, but links that do not share nodes can do so. An identical interference model has been used in e.g. [50, 58, 59] - it is about a wireless network with multiple channels available for transmission. Examples are orthogonal CDMA or FDMA channels that enable simultaneous communications in a neighborhood.

We use a binary variable $a_{i,j}(t)$ to indicate if a link $(i,j) \in \mathcal{L}$ is active ($a_{i,j}(t) = 1$) at *time slot* t. This activity exists only when node i is in `Tx` state and j is in `Rx` state during that *time slot*, i.e.,

$$S_i^{\texttt{Tx}}(t) + S_j^{\texttt{Rx}}(t) \geq 2 \cdot a_{i,j}(t), \quad \forall (i,j) \in \mathcal{L}, \;\; 1 \leq t \leq T \tag{3.2}$$

We can see the influence of the considered interference model on (3.2), where the restriction is formulated as inequality and not equality. Thus, this model then allows the following in a particular *time slot* t: a node in transmitting state can send data to multiple neighbor nodes if those are in receiving state; a node in receiving state can receive data from various neighbor nodes if those are in transmitting state.

2) Data Packets: A *data packet* is constructed by two parts: one *header* and a variable number of payloads. The *header* contains the communication protocol information, so the *header* size is constant; a *payload* carries sensed data. In our model, the *payload*'s size is smaller than the *header*'s. A *data packet* must have at least one *payload* and can have a maximum of P_{max} payloads.

Let $p_{i,j}(t)$ and $h_{i,j}(t)$ be the integer number of payloads and headers (*data packets*) respectively sent, for a successful transmission (3.2) over the wireless link $(i,j) \in \mathcal{L}$ at *time slot* t.

We can model the relationship between the number of *data packets* and payloads involved in that transmission as

$$h_{i,j}(t) = \left\lceil \frac{p_{i,j}(t)}{P_{max}} \right\rceil, \quad \forall (i,j) \in \mathcal{L} : a_{i,j}(t) = 1, \;\; 1 \leq t \leq T \tag{3.3}$$

3) Link Capacity and Transmission Cost: Let W_{max} be the constant maximum capacity of a wireless link $(i,j) \in \mathcal{L}$ for any t, i.e., the maximum integer number of *data packets* that can be successfully transmitted from i to j during any *time slot* t. Thus,

$$h_{i,j}(t) \leq W_{max}, \quad \forall (i,j) \in \mathcal{L} : a_{i,j}(t) = 1, \;\; 1 \leq t \leq T \tag{3.4}$$

Also, denote by $C_p^{\mathtt{Tx}}$ and by $C_h^{\mathtt{Tx}}$ a fixed energy cost for successfully transmit a *payload* and a *header* respectively. In the same fashion, designate by $C_p^{\mathtt{Rx}}$ and by $C_h^{\mathtt{Rx}}$ the corresponding cost for a node to successfully receive a *payload* and a *header*. We can then define the overall energy cost of transmitting p payloads inserted into h *data packets* over a wireless link $(i,j) \in \mathcal{L}$ at a *time slot* t as

$$C_p^{\mathtt{Tx}} \cdot p_{i,j}(t) + C_h^{\mathtt{Tx}} \cdot h_{i,j}(t)$$

and

$$C_p^{\mathtt{Rx}} \cdot p_{i,j}(t) + C_h^{\mathtt{Rx}} \cdot h_{i,j}(t)$$

be the total energy cost of receiving that same amount of information.

4) Data Buffer Model: Each sensor node $i \in \mathcal{S}$ maintains a *data buffer* $\mathcal{Q}_i(t)$ to store its own sensed *data payloads* and *data payloads* received from its neighbors in $\mathcal{N}_i$. Therefore, the *buffer* $\mathcal{Q}_i(t)$ represents the set of *data payloads* in node i's *data buffer* at *time slot* t. We assume that the headers do not make part of this *buffer* because they are attached and detached only at the instants of sending or receiving *data packets*, respectively (see Data Aggregation).

Let $Q_i(t) \geq 0$ be the size of *buffer* $\mathcal{Q}_i(t)$, i.e., $Q_i(t) = |\mathcal{Q}_i(t)|$. Since sensor nodes normally have limited RAM resources, we consider a finite *buffer* size Q_{max} in our model, i.e., $\forall i \in \mathcal{S}, \forall t, Q_i(t) \leq Q_{max}$.

5) Sensing Model: At each *time slot* t, every node i that is in sensing state ($S_i^{\mathtt{s}}(t)$) collects *raw data readings* from a hardware sensor. This activity produces a *payload* of information on that node. We denote the total energy cost for the sensing operation of $i \in \mathcal{S}$ as a constant $C^{\mathtt{s}}$ as we consider that all sensor nodes in the network have the same hardware sensor.

6) Data Aggregation: We implicitly assume that exists some underlying mechanism that perform operations with the existing data inside each nodes' *data buffer* such that the payloads and headers that constitute a *data packet* are organized as stated in the Data Packet construction above. This method allows that data from different sources can be opportunistically aggregated at intermediate nodes before reaching the *sink node*.

7) Data Buffer Size Dynamics: Considering the sensing, transmitting and receiving operations, it can be seen that the *buffer* size dynamics of a sensor node $i \in \mathcal{S}$ can be described as

$$0 \leq Q_i(t) \leq Q_{max}, \quad \forall i \in \mathcal{S}, \;\; 1 \leq t \leq T \tag{3.5}$$

$$Q_i(t) = Q_i(t-1) - p_i^{out}(t) + p_i^{in}(t) + S_i^{\mathtt{s}}(t), \quad \forall i \in \mathcal{S}, \;\; 1 \leq t \leq T \tag{3.6}$$

where

$$p_i^{out}(t) = \sum_{j:(i,j)\in\mathcal{L}} p_{i,j}(t) \qquad \text{and} \qquad p_i^{in}(t) = \sum_{j:(j,i)\in\mathcal{L}} p_{j,i}(t)$$

represent the total number of transmitted and received *data payloads* by that node i at *time slot* t respectively. Here, (3.6) represents the filling and clearing process of the node *buffer*; and (3.5) highlights the bounded *buffer* capacity, as seen in the Data Buffer Model above.

3.2 Energy Harvesting, Storage, and Consumption Models

There are three fundamental components in the embedded energy harvesting system of each sensor node: energy harvester, energy storage, and energy consumers. Specifically, consider a node $i \in \mathcal{S}$ at *time slot* $t \in [1, \ldots, T]$:

- $H_i(t) \geq 0$ is the amount of its harvested energy from the environment and we consider this amount as deterministic;
- $B_i(t) \geq 0$ is its residual battery level. We consider a battery with finite-capacity, i.e., $\forall i, t, \; B_i(t) \leq B_{max}$;

- $C_i(t) \geq 0$ is its total energy consumption. Each of the four possible states in $\mathcal{M}$ have a different energy consumption:

  - if the node is sleeping ($\mathtt{z}$), the energy consumption is very low and it is constant, i.e., $C_i(t) = S_i^{\mathtt{z}}(t) \cdot C^{\mathtt{z}}$;
  - if the node is sensing ($\mathtt{s}$), the energy consumption is also constant, i.e., $C_i(t) = S_i^{\mathtt{s}}(t) \cdot C^{\mathtt{s}}$ and usually $C^{\mathtt{s}} \gg C^{\mathtt{z}}$;
  - if the node is transmitting ($\mathtt{Tx}$) or receiving ($\mathtt{Rx}$) *data packets*, its energy consumption will have a constant component and a component that is a function of how many headers and payloads it is transmitting/receiving; and we can denote this by

    $$C_i(t) = S_i^{\mathtt{Tx}}(t) \cdot C^{\mathtt{Tx}} \;\; + \;\; C_p^{\mathtt{Tx}} \cdot p_i^{out}(t) + C_h^{\mathtt{Tx}} \cdot h_i^{out}(t)$$

    if transmitting or by

    $$C_i(t) = S_i^{\mathtt{Rx}}(t) \cdot C^{\mathtt{Rx}} \;\; + \;\; C_p^{\mathtt{Rx}} \cdot p_i^{in}(t) + C_h^{\mathtt{Rx}} \cdot h_i^{in}(t)$$

    if receiving, where

    $$h_i^{out}(t) = \sum_{j:(i,j)\in\mathcal{L}} h_{i,j}(t) \qquad \text{and} \qquad h_i^{in}(t) = \sum_{j:(j,i)\in\mathcal{L}} h_{j,i}(t)$$

    represent the total number of transmitted and received *data packets*, respectively. Here, the definition of $p_i^{out}(t)$ and $p_i^{in}(t)$ is the same as in the preceding section.

Considering the hardware of real sensor nodes, the total per-*slot* energy consumption $C_i(t)$ should be upper bounded by a finite value C_{max} (i.e., $\forall i, t,\ C_i(t) \leq C_{max}$), which depends on the maximum total power consumption of sensor nodes and the duration of a *time slot*. Practically, $C_{max} \ll B_{max}$, because C_{max} (typically in mJ) is multiple orders of magnitude smaller than B_{max} (typically in kJ).

The dynamic energy system of each sensor node can be modelled as:

$$B_{min} \leq B_i(t) \leq B_{max}, \quad \forall i \in \mathcal{S},\ \ 1 \leq t \leq T \tag{3.7}$$

$$B_i(t) = B_i(t-1) - C_i(t) + H_i(t), \quad \forall i \in \mathcal{S},\ \ 1 \leq t \leq T \tag{3.8}$$

$$B_i(T) \geq B_i(0), \quad \forall i \in \mathcal{S} \tag{3.9}$$

Here, (3.8) represents the recharging and discharging process of the battery; and (3.7) highlights the bounded battery capacity. More importantly, the constraint (3.9) ensures *neutral operation*, which is expected to be achieved by every sensor node in the network, i.e., a node must consume only the energy it harvested – its residual battery level at the end of operation must be always greater or equal than at the beginning of operation.

3.3 Events

We can consider that an event is a notable occurrence at a particular point in time. In our model, we do not define particularly what an event is, as it depends on the general purpose of the wireless network and the kind of hardware sensor mounted on the set of sensor nodes $\mathcal{S}$. For example, if the deployed nodes are monitoring the fluid pressure at specific points of an industrial piping system, sudden rise or drop of pressure (events) in the neighborhood of those points could be interpreted as pipe obstructions or leakage in those areas, respectively.

Whatever is the case, the general approach is to ensure that every point of the deployment area is covered all the time by at least one active sensing sensor, provided that there is an available sensor in the random placement [60]. Some algorithms, e.g. [61], are designed to minimize the probability that any given point is not covered by an active sensor, provided that it could be covered. Doing so will also maximize the detection probability of any event and ensure instantaneous detection, if the event is within range of at least one sensor that is sensing. In some applications for transient events (e.g., an animal which arrives and then leaves), the goal may be to maximize the detection probability of the events before they disappear, and some delay in the detection is acceptable.

Furthermore, it is widely accepted that real-world events can be modelled as stochastic processes [62]. So, for this model, we consider that the starting *time slots* and locations of the occurring events are known beforehand, as they follow a predictable distribution. Thus, let $E_i(t)$ be a binary variable indicating that at *time slot* t the node $i \in \mathcal{S}$ is close enough to detect an event ($E_i(t) = 1$) or not ($E_i(t) = 0$) before it disappears, as all events have a fixed duration δ. This suggests that is possible that E_i can be 1 to more than one node at the same time, and can be 1 several *time slots* in a row for the same subset of sensor nodes.

For a certain event be declared as detected or captured, there must be at least one sensor node $i \in \mathcal{S}$ in its neighborhood that is in sensing state while that event exists, i.e.,

$$\exists i \in \mathcal{S}, \exists t \in [\delta_{start}, \delta_{end}] : \quad (S_i^{\mathsf{s}}(t) = 1 \wedge E_i(t) = 1) \tag{3.10}$$

where $\delta_{start} \in \{1, ..., T - \delta\}$ and $\delta_{end} = \delta_{start} + \delta$ are the starting and ending *time slots* of the event, respectively.

3.4 Optimization Objective

It is not the purpose of this work to make any probabilistic (e.g. specific distribution) and stochastic assumptions of the dynamic network states, including harvested energy, energy costs (for sensing, sleeping, transmitting, and receiving), as well as transmission and data buffering capacities, or events space-time location. The objective of this model is to seek an algorithm that can solve the following finite-horizon optimization problem, by finding the most suitable state $S_i^{\mathbf{m}}(t) \in \mathcal{M}$ and wireless *payload* transmission $p_{i,j}(t)$ for each sensor node in $\mathcal{S}$ at each *time slot* $1 \leq t \leq T$, so as to maximize the number of *data payloads* that arrive at the *sink node* $\odot$ at the end of the horizon, i.e., at $t = T$ (3.11).

maximize

$$Q_{\odot}(T) \tag{3.11}$$

subject to

Constraints (3.1) - (3.10)

4 Implemented Model

To solve the finite-horizon optimization problem of Section 3.4 it is necessary to translate it into something that can be run by a processing-capable machine, like a computer. This translation brings up additional constraints not considered in the original problem, but those are required so that the obtained solution meets the purposes of the initial design.

These extra restrictions may come from the very nature of Mixed-Integer Programming, where it's just not possible to write some given constraints precisely as they are written in theory, so some additional auxiliary variables are needed. Others become necessary after further refinement of the existing model, where its behavior was not exactly the expected one.

4.1 The need of Mixed-Integer Programming

First, some definitions:

- An optimization model is an Integer Program (IP) if *any* of its decision variables is discrete.
  - If *all* variables are discrete, the model is a pure Integer Program.
  - Otherwise, the model is a Mixed-Integer Program (MIP).
- An IP or MIP model is an Integer Linear Program or a Mixed-Integer Linear Program (MILP), respectively, if its (single) objective function and all its constraints are linear.
  - Otherwise, it is an Integer Non-Linear Program or a Mixed-Integer Non-Linear Program.

In an optimization problem, the types of mathematical relationships between the objective and constraints and the decision variables determine how hard it is to solve, the solution methods or algorithms that can be used for optimization, and the confidence one can have that the solution is truly optimal.

The optimization problem stated in Section 3.4 is described to be a Mixed-Integer Program: some of the variables are constrained to be integers (e.g., the number of *data payloads* at any node, $Q_i(t)$), while other variables are allowed to be non-integers (e.g., the battery consumption of a node at any *time slot*, $C_i(t)$).

The additional constraints imposed by the integral variables make a MIP problem to be much harder to solve. However, eliminating these integrality constraints (also designated *relaxation*) results in a Linear Program, as the derived constraints and objective function are linear. This linearity presence also implies that is guaranteed the existence of at least one globally optimal solution – and that we can get its limits if solved.

For these reasons, the implemented model follows a Mixed-Integer Linear Programming (MILP) approach.

4.2 The MILP model

Here it is described the resulting MILP model that is based in the theoretical model of Chapter 3. Most of the expressed notation at this section has the same meaning of the one in Chapter 3. Still, here an effort is made to express and describe the numerical constraints or mathematical equivalence of the referred data.

4.2.1 Data

Network

Symbol	Description
$\mathcal{N}$	Set of all nodes (sensor nodes + *sink node* $\odot$). Each node has a number that corresponds to the node identifier. $\mathcal{N} = \{1, \ldots, N\}$, where N is the number of nodes in the network.
$\odot$	The *sink node* (or just *sink*) is the only IoT gateway in $\mathcal{N}$, i.e., $\odot \in \mathcal{N}$. The goal is to maximize the received payloads in the *sink*. It does not have energy or *buffer* size limitations.
$\mathcal{L}$	Set of all possible wireless links (i, j) between the nodes $\mathcal{N}$ where all data is transferred.
D_i	Distance to the *sink node*, i.e., the length of the shortest path between node $i \in \mathcal{N}$ and *sink node* $\odot$. Distance of *sink node* to itself is 0.
T	Finite-horizon period limit, or operational *time-frame*. $t = 0$ coincide with the initial state of the network. The model finds a solution for $1 \leq t \leq T$.

Nodes Operation and Data Communication

Symbol	Description
$\mathcal{M}$	Set of all the operational states of a certain node $i \in \mathcal{N}$. $\mathcal{M} = \{\mathtt{z}, \mathtt{s}, \mathtt{Tx}, \mathtt{Rx}\}$. They are: sleeping $\mathtt{z}$, sensing $\mathtt{s}$, transmitting $\mathtt{Tx}$, or receiving $\mathtt{Rx}$.
$Q_i(0)$	*Buffer* size of a node $i \in \mathcal{N}$ at the start of the simulation, i.e., how many *data payloads* it has in the *buffer* at $t = 0$.
Q_{max}	Maximum number of *data payloads* that a sensor node can have in its *buffer* at any time.
P_{max}	Maximum number of *data payloads* that a transmitted *data packet* can contain.
α	Duration of the period, in *time slots*, in which a minimum amount of *data payloads* will be required to enter the *sink node* $\odot$. This means that from α to α *time slots* a certain minimum amount of *data payloads* (see β) need to reach the *sink node*, with $1 \leq k \leq \frac{T}{\alpha}$, being k the current period.
β	Minimum amount of *data payloads per sensor node* that the *sink node* must receive in each of the time periods k (see α).
$\mathtt{Tx}_{min}$	Minimum amount of *data transmit* ($\mathtt{Tx}$) states that a sensor node must achieve in each of the time periods k (see α).
$\mathtt{s}_{min}$	Minimum amount of *sensing* ($\mathtt{s}$) states that a sensor node must achieve in each of the time periods k (see α).
γ	Value between 0 and 1 that sets what fraction of the total number of *payloads* that enter the *sink node* throughout all the operational *time-frame* can remain outside in the sensor nodes *data buffer*.

Energy

Symbol	Description
$B_i(0)$	Battery level of node $i \in \mathcal{N} \setminus \odot$ at the start of operation. This is also the value of battery that the node must have at the end of the operational *time-frame*.
B_{min}	Minimum level of battery that any node can reach during operation.
B_{max}	Maximum level of battery that any node can reach during operation.
$H_i(t)$	Amount of increased battery level of a sensor node $i \in \mathcal{N} \setminus \odot$ in a certain *time slot* t. This amount comes from harvested energy from the environment.
$C_i(t)$	Total amount of decreased battery level of a sensor node $i \in \mathcal{N} \setminus \odot$ at *time slot* t. This amount is due to consumption profiles regarding the node operational state and data communications. The value can be further decomposed in:
$C_p^{\texttt{Tx}}$	decreased battery level when transmitting one *data payload*
$C_h^{\texttt{Tx}}$	decreased battery level when transmitting one *header* (*data packet*)
$C_p^{\texttt{Rx}}$	decreased battery level when receiving one *data payload*
$C_h^{\texttt{Rx}}$	decreased battery level when receiving one *header* (*data packet*)
$C^{\texttt{z}}$	decreased battery level when in *sleeping* (`z`) state
$C^{\texttt{s}}$	decreased battery level when in *sensing* (`s`) state
$C^{\texttt{Tx}}$	decreased battery level when in *data transmitting* (`Tx`) state
$C^{\texttt{Rx}}$	decreased battery level when in *data receiving* (`Rx`) state

Events

Symbol	Description
$E_i(t)$	Binary variable that is 1 if node $i \in \mathcal{N} \setminus \odot$ is close enough to detect an *event* at *time slot* t. Otherwise, is 0.
$v(t)$	Binary variable that is 1 if an *event* has started at *time slot* t. Otherwise, is 0. This means that $v(t) = \max\{E_i(t) : i \in \mathcal{N} \setminus \odot\}$.
δ	Fixed duration, in *time slots*, of all emerging events.

4.2.2 Decision Variables

Nodes Operation and Data Communication

Symbol	Description
$S_i^{\mathbf{m}}(t)$	Binary variable that is 1 if the node $i \in \mathcal{N}$ at *time slot* t is in the state $\mathbf{m} \in \mathcal{M}$. Otherwise, is 0.
$h_{i,j}(t)$	Number of *data packets* (*headers*) being transmitted over wireless link $(i,j) \in \mathcal{L}$ at *time slot* t.
$p_{i,j}(t)$	Number of *data payloads* being transmitted over wireless link $(i,j) \in \mathcal{L}$ at *time slot* t.
$a_{i,j}(t)$	Binary variable that is 1 if the link $(i,j) \in \mathcal{L}$ is active during *time slot* t, i.e., there is information flowing from i to j. Otherwise, is 0.
$\bar{f}_{i,j}(t)$	Binary variable that is 1 if there is a *data packet* being transmitted over a wireless link $(i,j) \in \mathcal{L}$ at *time slot* t that is not full, i.e., the number of *data payloads* in it is less than P_{max}. Otherwise, is 0.
$Q_i(t)$	*Buffer* size of node $i \in \mathcal{N}$ at *time slot* t, i.e., how many *payloads* it has in the *buffer*.

Energy

Symbol	Description
$B_i(t)$	Battery level of node $i \in \mathcal{N} \setminus \odot$ at *time slot* t.

4.2.3 Constraints

Nodes Operation

$$\sum_{\mathbf{m} \in \mathcal{M}} S_i^{\mathbf{m}}(t) = S_i^{\mathtt{z}}(t) + S_i^{\mathtt{s}}(t) + S_i^{\mathtt{Tx}}(t) + S_i^{\mathtt{Rx}}(t) = 1, \quad \forall i \in \mathcal{N}, \;\; 1 \leq t \leq T \tag{4.1}$$

A node can only be in one operational state at each *time slot*.

$$S_{\odot}^{\mathtt{Rx}}(t) = 1, \quad 1 \leq t \leq T \tag{4.2}$$

The *sink node* $\odot$ is in the receiving state at all times.

Data Communication

(4.3) $$S_i^{\texttt{Tx}}(t) \leq \sum_{j:(i,j)\in\mathcal{L}} a_{i,j}(t), \quad \forall i \in \mathcal{N},\ 1 \leq t \leq T$$

A node can only be in a transmitting state (`Tx`) if any of its outgoing links is active, i.e., the node is sending *payloads*.

(4.4) $$S_i^{\texttt{Rx}}(t) \leq \sum_{j:(j,i)\in\mathcal{L}} a_{j,i}(t), \quad \forall i \in \mathcal{N},\ 1 \leq t \leq T$$

A node can only be in a receiving state (`Rx`) if any of its incoming links is active, i.e., the node is receiving *payloads*.

(4.5) $$S_i^{\texttt{Tx}}(t) + S_j^{\texttt{Rx}}(t) \geq 2 \cdot a_{i,j}(t), \quad \forall (i,j) \in \mathcal{L},\ 1 \leq t \leq T$$

If a link (i,j) is active, then the node i must be in `Tx` state, and the node j must be in `Rx` state.

(4.6) $$a_{i,j}(t) \leq p_{i,j}(t) \leq \infty \cdot a_{i,j}(t), \quad \forall (i,j) \in \mathcal{L},\ 1 \leq t \leq T$$

If a link (i,j) is active, then there must be *data payloads* flowing through it.

(4.7) $$Q_i(0) + \sum_{\tau=1}^{t-1} p_i^{in}(\tau) + \sum_{\tau=1}^{t-1} S_i^{\texttt{s}}(\tau) \geq \sum_{\tau=1}^{t} p_i^{out}(\tau), \quad \forall i \in \mathcal{N},\ 1 \leq t \leq T$$

The number of outgoing *payloads* at a node on a *time slot* t cannot be higher than the number of its incoming *payloads* before that *time slot* $(t-1)$. That includes *payloads* received from other nodes, the *payloads* created when sensing, and the initial *payloads* in its *buffer*.[1]

(4.8) $$\overbrace{Q_i(0) + \sum_{\tau=1}^{t} p_i^{in}(\tau) + \sum_{\tau=1}^{t} S_i^{\texttt{s}}(\tau) - \sum_{\tau=1}^{t} p_i^{out}(\tau)}^{Q_i(t)} \leq Q_{max}, \quad \forall i \in \mathcal{N} \setminus \odot,\ 1 \leq t \leq T$$

The number of *payloads* at a node cannot be higher than its maximum *buffer* capacity.[1]

(4.9) $$P_{max}\left(h_{i,j}(t) - \bar{f}_{i,j}(t)\right) + 0.1 \cdot \bar{f}_{i,j}(t) \leq p_{i,j}(t) \leq P_{max} \cdot h_{i,j}(t), \quad \forall (i,j) \in \mathcal{L},\ 1 \leq t \leq T$$

Sets the correct number of *data packets* (*headers*) when transmitting *payloads*. In other words, if *data packets* are flowing through a link, only one *data packet* could not be full. When there are incomplete *data packets*, the expression $0.1 \cdot \bar{f}_{i,j}(t)$ transforms the first constraint from inequality to strict inequality.

[1] See (3.6) at page 15 for the definition of p_i^{in} and p_i^{out}.

$$\sum_{t=(k-1)\alpha+1}^{k\alpha} \sum_{j:(j,\odot)\in\mathcal{L}} p_{j,\odot}(t) \geq (N-1)\cdot\beta, \quad 1 \leq k \leq \frac{T}{\alpha} \tag{4.10}$$

The amount of *data payloads* that the *sink node* must receive during a time period α needs to be higher than the number of sensor nodes multiplied by the parameter β, i.e., each sensor node sends an average of β *payloads* to the *sink node* per period.

$$\sum_{t=(k-1)\alpha+1}^{k\alpha} S_i^{\mathtt{Tx}}(t) \geq \mathtt{Tx}_{min}, \quad \forall i \in \mathcal{N}\setminus\odot, \;\; 1 \leq k \leq \frac{T}{\alpha} \tag{4.11}$$

A sensor node must be in a transmitting state at least $\mathtt{Tx}_{min}$ times per period k.

$$\sum_{t=(k-1)\alpha+1}^{k\alpha} S_i^{\mathtt{s}}(t) \geq \mathtt{s}_{min}, \quad \forall i \in \mathcal{N}\setminus\odot, \;\; 1 \leq k \leq \frac{T}{\alpha} \tag{4.12}$$

A sensor node must be in sensing state at least $\mathtt{s}_{min}$ times per period k.

$$\frac{\sum_{k=1}^{T/\alpha}\left(\sum_{i\in\mathcal{N}\setminus\odot}(D_i\cdot Q_i(k\cdot\alpha))\right)}{T/\alpha} \leq \gamma\cdot Q_\odot(T), \quad 0\leq\gamma\leq 1 \tag{4.13}$$

At the end of each period k, it is given a weight to each of the *data payloads* that do not reach the *sink node* – that weight is linearly dependent to their current distance (in wireless hops) to the *sink node*. This weight must not be higher than a fraction γ of the number of *data payloads* that reach the *sink node* at the end of *time-frame* T. This penalization function ensures that the sensor nodes do not accumulate *data payloads* through time. Instead, it will regularly force the nodes to flush the *payloads* on their *buffers* to nodes closer to the *sink node*. For further explanation, see `Communications` in Section 4.4 (p.28).

Events

$$\sum_{i\in\mathcal{N}\setminus\odot}\left(E_i(t)\cdot\sum_{\tau=t}^{\min\left\{t+\frac{\delta}{2},T-t\right\}} S_i^{\mathtt{s}}(\tau)\right) \geq v(t), \quad 1\leq t\leq T \tag{4.14}$$

At least one sensor node must capture a nearby *event* in the first half of its duration.

Energy

$$B_{min} \leq B_i(t) \leq B_{max}, \quad \forall i\in\mathcal{N}\setminus\odot, \;\; 1\leq t\leq T \tag{4.15}$$

The battery level of all sensor nodes must be within the imposed limits.

(4.16) $B_i(T) \geq B_i(0), \quad \forall i \in \mathcal{N} \setminus \odot$

Maintain the network in *neutral operation*, i.e., the battery level of all nodes at the end of the finite horizon period must be higher or equal than at the beginning.

(4.17) $B_i(t) = B_i(t-1) - C_i(t) + H_i(t), \quad \forall i \in \mathcal{N} \setminus \odot, \ \ 1 \leq t \leq T,$

$$C_i(t) = C_p^{\mathtt{Tx}} \cdot p_i^{out}(t) + C_h^{\mathtt{Tx}} \cdot h_i^{out}(t) + C_p^{\mathtt{Rx}} \cdot p_i^{in}(t) + C_h^{\mathtt{Rx}} \cdot h_i^{in}(t) + C^{\mathtt{z}} \cdot S_i^{\mathtt{z}}(t) + C^{\mathtt{s}} \cdot S_i^{\mathtt{s}}(t)$$

Ensures the proper recharging and discharging process of the battery throughout the entire operation.[2]

4.2.4 Objective

(4.18) **maximize** $Q_{\odot}(T)$

The MILP single objective is to maximize the total number of *data payloads* that arrive at the *sink node* over time.

4.3 Solving a MILP problem: the CPLEX Optimizer engine

Since MILP problems have integer variables, they must be solved by some systematic search and not by a potentially exhaustive one. One traditional algorithm for solving these problems is designated *Branch and Bound* (B&B).

This method begins by finding the optimal solution to the *relaxation* of the original MILP problem – removing all of the integrality restrictions results in a Linear Program, much more straightforward to solve. If in this solution, the decision variables with integer constraints have integer values, then no further attempts are required – this solution is an optimal solution of the original problem. However, this occurs very rarely.

If one or more integer variables have non-integral solutions, the *Branch and Bound* method chooses one variable that violates the integrality restriction, i.e., a variable which value is $x_i = \gamma$, with γ as non-integral, making the new optimal value an upper bound of the optimal value of the original problem. Subsequently, the problem *branches* into two new sub-MILP-problems wherein one of them has an additional restriction $x_i \leq \lfloor \gamma \rfloor$ and the other has $x_i \geq \lceil \gamma \rceil$ as extra constraint. The variable x_i is labeled as a *branching variable*.

[2] See (3.6) (p. 15) for the definition of p_i^{in} and p_i^{out}, and (3.8) (p. 16) for the definition of h_i^{in} and h_i^{out}.

If the B&B can compute optimal solutions for each of these sub-problems, then the better of these two solutions will be optimal to the starting problem: this process replaced it by two simpler (or at least more-restricted) MILPs.

The next move is to apply the same approach to these two MILPs, solving the corresponding LP relaxations and, if necessary, picking branching variables. This process constructs a *search tree*. The MILPs generated by the search procedure are termed the nodes of the tree, with the initial MILP appointed as the *root node*. The leaves of the tree are all the nodes still unbranched.

It is not necessary to branch on a specific node if the B&B has just solved its Linear Programming (LP) relaxation and all of the integrality restrictions of the root node are satisfied in this solution – a feasible solution to the original MILP has been found. This node can be called *fathomed* as is a permanent leaf of the search tree.

The *incumbent* is the best integer solution found at any point in the search. At the start of the search, there is no incumbent. By analyzing the information provided by the last integer feasible solution found, the B&B records this solution as the new incumbent (along with its objective value) if it has a better objective function value than the current incumbent (or if it has no incumbent yet). Otherwise, no incumbent update is necessary, and the B&B simply proceeds with the search.

Two other possibilities can lead to a node being fathomed. First, it can happen that the branch that directed to the present node added a restriction that made the LP relaxation infeasible. Naturally, if this node contains no feasible solution to the LP relaxation, then it holds no feasible integer solution. The other possibility is that an optimal relaxation solution is discovered, but its objective value is worse than that of the current incumbent. Clearly, this node cannot yield a better integral solution and again can be fathomed.

This algorithmic approach demands a proper optimization engine that is constructed for this kind of algorithmic approach. Here enters the CPLEX Optimizer.

The **CPLEX Optimizer** is one of the optimization engines provided by the IBM ILOG CPLEX Optimization Studio[3] and it is a mathematical programming optimization engine that implements the Simplex and Barrier methods, Mixed Integer programming, and the *Branch and Bound* method.

One of the reasons to use this software suite was the extended use amongst fellow researchers here in the LCT. Other major reason was the fact that it combines a fully featured

[3] `https://www.ibm.com/products/ilog-cplex-optimization-studio`

Integrated Development Environment (IDE) that supports the Optimization Programming Language (OPL).

OPL, the modeling language, allows us to write a mathematical representation of our problem that is separate from our data. This separation enables significant improvement because we can easily rerun the same model while adjusting the input data for different simulations.

Further reference to the CPLEX Optimizer engine on the rest of the document will be just "CPLEX" or "the CPLEX engine".

4.4 The resulting OPL Model

The MILP model described in Section 4.2 materializes itself as a computing-capable model in OPL, and this resulting model can be accessed at `https://bitbucket.org/mollering/mobiwise-cplex`. If we analyze in detail the code, we can find that its data, decision variables/expressions, constraints, and objective fit perfectly the mathematically defined MILP model in such a way that still is quite intuitive to comprehend.

However, this model – that in its core is a representation of the theoretical model of Chapter 3 – went through a significant number of modifications until the present form detailed in Section 4.2.

The GUI platform developed in the course of this dissertation and described in Chapter 5 provided an interactive build-up capability that started with the core model (with some undetected flaws at first) until a valid up-and-running model that conforms with the general guidelines presented in Chapter 3.

A compilation of the modifications that were necessary and their motivation is shown here.

Nodes Operation

Constraint (3.1) is maintained as (4.1) but expanded to include the *sink node* along the sensor nodes. Also, (4.2) is added so that the state of the *sink node* does not fluctuate as it happened before its inclusion – it is essential that the *sink* keeps the receiving state such as to maximize the reception of new *payloads*.

Data Communication

Constraints (4.3) and (4.4) were appended after some nodes would be in transmitting and receiving states even when not sending any *payloads* – virtually, this would save them battery because when sending/receiving zero *data payloads* the energy consumption was nonexistent. Similarly, constraint (4.6) was also attached to make sure that a link can only be active if *payloads* are passing through it.

Theoretical constraints (3.5) and (3.6) were rewritten as (4.7) and (4.8). In practice, the resulting constraint set is maintained. The modifications ensure that the input flow of *payloads* on a node's *buffer* equals its output flow, that is, no *payloads* are created or disappear "from/to nothing" (4.7) – this constraint also ensures that $Q_i(t)$ is never negative – and that the number of *payloads* inside a *buffer* does not go beyond its maximum capacity (4.8). Data parameter $Q_i(0)$ was supplemented accordingly.

Constraint (4.9) is just a linear equivalent of the original (3.3). Decision variable $\bar{f}_{i,j}(t)$ was added as an auxiliary in this process.

Period-based constraints (4.10), (4.11), and (4.12) were added later in the model formulation to further constraint the network to regularly perform some necessary actions, such as to create and send data to the *sink node*. Data parameters α, β, $\texttt{Tx}_{min}$ and $\texttt{s}_{min}$ were added subsequently. In deeper detail:

- An initial version attested that most of the *payloads* only reached the *sink node* near the end of the finite horizon T. In other words, the information was utmostly delayed, a situation that is not acceptable when it is needed to actuate regarding an *event* or collect the network information as early as possible. Constraint (4.10) solved this behavior by introducing the concept of time periods.
- With the inclusion of (4.10), the model put the *sink*'s closest nodes pretty active: frequently in *sensing* state to create *payloads*, and only transmitting those at the end of the time periods. In consequence, most of the time the peripheral nodes were not capable of getting their *payloads* through the inner nodes, so they remained mostly *sleeping*. This situation does make sense considering the model objective, since the transmission of the few *payloads* generated in the most distant nodes will force the passage through several nodes that will have to be in the `Tx`/`Rx` states, preventing

them from being in sensing mode (which would be more beneficial to maximize the objective function).

However, this is not desirable in a real context because the intention is to receive information from all nodes, not just from those that are close to the *sink*. Constraint (4.12) was added to force every node (most importantly, the distant ones) to obtain data from the environment and (4.11) ensured that the *payloads* not remained in the nodes' *buffers*, i.e., that they were actually transmitted.

- The model approach to overcome the previous constraints was to play some "ping-pong" game with the *payloads* acquired by the most distant nodes: as all nodes had a chance to be transmitting once per period, constraint (4.11) was successfully enforced.

To minimize this impact on the obtained solutions, constraint (4.13) ensures that the optimization objective has an increasing penalty as nodes further away keep *payloads* inside their *buffers*. Thus, at each period, this penalty is minimized when the network "pushes" *payloads* closer to the *sink node* – the minimum possible penalty is when the *payloads* enter the *sink node*. The included data parameter γ controls what fraction of the total number of *payloads* acquired by the *sink node* at the end of the operational timeframe can stay inside the network. The distance-to-sink data D_i was also required for this constraint.

Events

As constraint (3.10) was difficult to implement as is written theoretically, it was rewritten as the linear constraint (4.14). The final linear expansion needed data $v(t)$ to achieve this.

5 Simulation and Analysis Platform

What started by being a secondary idea of having a simple way to represent in some visual form the results obtained by running the CPLEX engine on the OPL model, quickly developed into being a main project within the scope of this thesis.

This chapter describes the specifications and functionalities of a MATLAB-based GUI that we call `Network Visualizer` app (**NetVis**). This app allows interoperability with the CPLEX Studio (Section 4.3) and enables us with the ability to validate and analyze the dynamic network states that correspond to optimal or sub-optimal solutions of the optimization problem of Section 3.4. The **NetVis** User's Guide is in Appendix B.

5.1 Conception and Requirements

In the early stages of the core model development, it was immediately clear that it was necessary to find a way to validate the model optimization results. We were obtaining data, but we lacked a proper method for:

- Evaluating if the network model was valid from a real-world perspective, i.e., it followed the physical rules of an IoT sensorial network.
- Assessing if the behavior exposed by the results met our goals for that network, even if they were valid.
- Perform continued experimentations with different data parameters or constraints in an agile[1] development cycle.

The results provided by CPLEX come in a numerical array format. This representation is good for a computer, but not for us trying to obtain a meaningful and practical understanding

[1] Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional projects.

of the model response. Another necessity was to obtain and manage data quickly to feed as input to the CPLEX, as without proper input data the results had no real meaning.

The obstacles presented above created the need for a supporting platform whose core requirements are:

- Translate the CPLEX array-style results to datasets in which it is possible to manipulate the data for further analysis.
- Provide a graphical representation of a given network, its parameters, and its operation over time. This analysis is performed by using the CPLEX results as source.
- Grant the ability to change data parameters quickly and feed them as input to CPLEX.
- Perform a series of automatic validations on the results presented by CPLEX to avoid the existence of incorrect models.

Other non-functional assumptions to point out:

- Maintain a clean graphical interface, with intuitive and straightforward operativity, and privilege mouse use.
- Possibility to add more features in the future, so the platform evolution should be as modular as possible.

5.2 Development

Although the platform had a core of principles (Section 5.1) to follow during its construction, in reality, the process of software creation was quite experimental and iterative. As this process was completely intertwined with the process of building-up and validation of the OPL model, several times the need to identify a particularity on the CPLEX results or the necessity to rewrite some constraints changed the focus on the conception of new features or forced the modification of existent ones.

This method means that the development was not linear and some secondary requirements changed during the steps to achieve the 3800 lines GUI software that can be accessed at `https://bitbucket.org/mollering/mobiwise-matlab`. Thus we can call this method as agile considering that the optimization model and the presented analysis platform grew with each other.

The MATLAB[2] environment and language were the chosen tools to build the app since:

[2] `https://www.mathworks.com/products/matlab.html`

- MATLAB language allows easy creation of data-based algorithms, matrix manipulation, and data analysis.
- MATLAB environment provides a streamlined interface with programs written in other languages or with externally created data.
- Mathworks[3] maintains App Designer[4], a tool that integrates the two primary tasks of app building – laying out the visual components and programming app behavior.
- I have considerable experience working with them, and fellow researchers commonly use those tools.

5.3 Features and Functionalities

1) Data Creation and Storage: In Section 4.2.1 there is an enumeration of a somewhat extensive set of necessary data used as input of the implemented model. The **NetVis** app allows a user to set and modify these various parameters quickly and makes sure they can be reused in future runs of the model since they can be stored as MATLAB data.

2) Graph Creation and Visualization: Provided the number of nodes (N) in a graph representing the modeled network ($\mathbf{G}$) and other graph options, the app automatically creates the sets of all nodes ($\mathcal{N}$) and wireless links ($\mathcal{L}$) that connect those nodes. The app also defines the *sink node* ($\odot$) as being the node in the closest position to all other nodes. A possible 2D representation of such network is then displayed.

3) Interface with CPLEX: The app takes the given data and writes it in a format comprehensible to the CPLEX OPL engine. Next, it can send a command to the engine to start the process of finding an optimal model solution with the specified data, with the help of a B&B algorithm. If the model tends out to be infeasible, the app receives the information of which constraints are producing a problem. If there is an optimal or sub-optimal solution, then the app imports it and translates it back to a MATLAB suitable format.

4) Results Visualization: When the app receives a valid solution, it can display the behavior of the network through time, namely: the operational state of the sensor nodes, the amount of *data payloads* on the nodes' *buffers*, the amount of battery of each sensor node,

[3] `https://www.mathworks.com`

[4] `https://www.mathworks.com/products/matlab/app-designer.html`

and which links are active and how many data packets and payloads flow through them. See an example in Figure B.2.

5) Results Validation: The app runs a set of validation functions on the results data to ensure that those are physically consistent and the model constraints are enforced. Examples: the residual battery cannot overflow a specified maximum value, even if the energy harvesting is too much and a node can practically "live" only from the harvested energy; if a node is transmitting, then the variation of *data payloads* on its *buffer* on that time slot must be negative.

6) Result Analysis: Several functions run through the obtained data to extract further data metrics not provided by CPLEX. These functions calculate cumulative data-based tendencies through time and then present a set of different graphical representation of these metrics. See examples in Section B.4.7.

7) Import and Export of Data: The app allows the following operations:
- Save and load specific network graphs, for reuse on various model types.
- Manual import of CPLEX data, in case we want to run a model on the CPLEX Studio IDE manually.
- Save and load model optimization results along with the data parameters and other information extracted from the results in a data structure suitable for numerical analysis (MAT files). See Section B.5.2 to know more.

8) Expansion and Modification Capabilities: Create new algorithms to manipulate data, add more network settings and data validations functions, extract further metrics and graphics from data results or improving the user interface is quite easy as the App Designer provides these tools in a very intuitive way, making use of graphical development and object callbacks.

5.4 Functional Flow Overview

To successfully perform a simulation with the implemented model from Chapter 4, one must first define its input parameters. The **NetVis** app and interoperability framework facilitates this procedure and guarantees that some parameters do not invalidate each other.

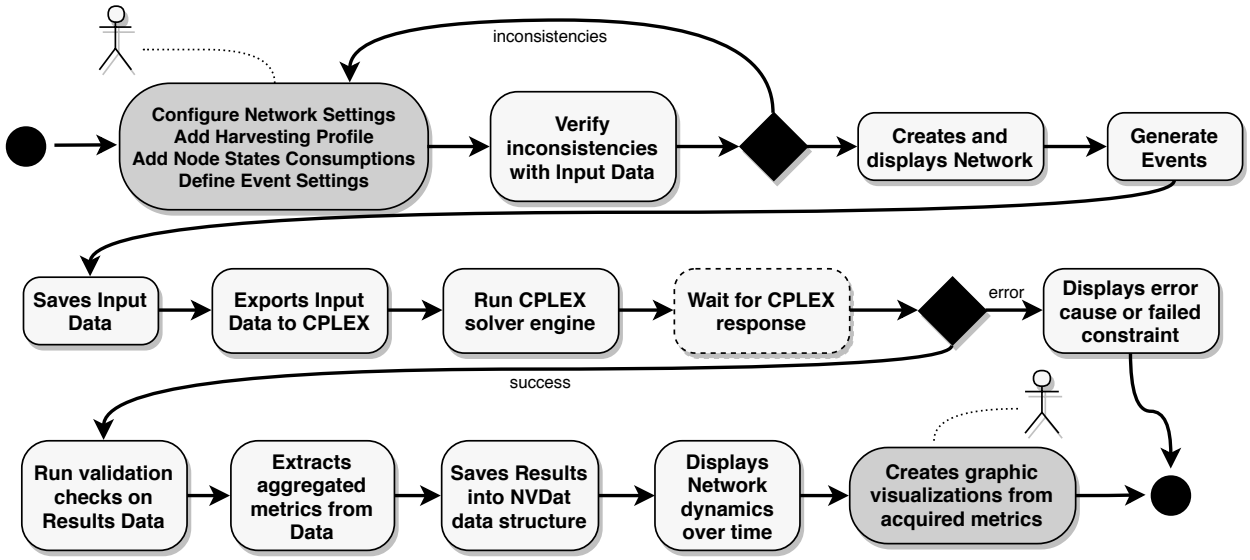


Figure 5.1: Activity diagram of a run of the **NetVis** framework

Figure 5.1 displays the course of actions followed by the app, the framework and its user in the overall process of producing optimizations results with the model. One can inspect what kind of network is modeling before running the optimization engine. When the results are ready, which can take some time depending on the provided configurations, the app aids in the process of analyzing the data and storing it for future examination.

6 Evaluation Setup and Results

It is essential to feed the model proposed in Chapter 4 with data that mirrors a realistic IoT setting to test its validity. Here it is described the reasoning behind the choices regarding the used parameters values, which metrics are evaluated and how that was achieved.

6.1 Goals

Foremost and of fundamental significance, we expect to evaluate the model validity. Validity implies that, in its core design, the model should not result in unnatural behavior. It has to reproduce as strictly as possible what is operationally expected from an EH-IoTN constructed to gather environmental data without logical faults. Skipping hardcoded rules like manipulating data beyond the corresponding operational states, overriding battery physical bounds, or allowing uncaptured events, even if that provides a better optimization objective, are examples of logical flaws to avoid.

Second and on top of ensured validity, we strive to assess the model performance. After corroborating that its behavior is within the imposed logical limits, the purpose is to uncover what are the conceivable dynamics that an EH-IoTN adopts to guarantee an optimal utilization of the resources that are available to it. However, that use of resources should, above all, exhibit the premises already well-established: the IoT network must take advantage of its multi-hop design and data aggregation capability to make data collection more efficient; it has to ensure that environmental events are captured without failure; and it needs to prove that it can operate indefinitely while maintaining a neutral operation regarding its energetic availability.

6.2 Evaluation Conditions

To demonstrate the use of the framework composed of the OPL model linked with the **NetVis** app, we resort first to study a possible physical implementation of a network with

the characteristics detailed in Chapter 4.

This practical research leads us to a point where we can assume that most of the analyzed network parameters are as close to a realistic setting as we are capable of assessing within the current project status. It is vital to note that the model remains as a simplification of real network conditions, mainly because some temporal-related data is assumed to be deterministic, e.g., the energy harvesting profiles or the events' distribution.

6.2.1 Platforms and Methods

Taking into consideration the proposed objectives in Section 6.1, testing the model involves conducting simulations that operate with the studied data parameters as input.

The framework CPLEX-**NetVis** provides these simulations, and the process – mainly automated, as briefly described in Section 5.4 – is as follows:

1. **NetVis** is employed to specify suitable data parameters;
2. A simulation is requested, that in turn runs a B&B process (Section 4.3) within the CPLEX studio engine;
3. The process in `2.` produces the optimal network dynamics if those exist;
4. **NetVis** is used to view the results and analyze them in conformity with established metrics. Consequently, it allows to perform comparisons between all the simulation results.

6.2.2 Fixed Simulation Parameters

Due to the extensive quantity of input data in the model, which produces numerous combinations of experimental conditions, we decided to fix most of the parameters and only vary two of them, so in the retrieved results we can analyze how these parameters influence the goals established in Section 6.1.

However, to settle some of the parameters is not a straightforward task, as most of them have a viable range of values, namely the ones regarding the `Nodes Operation and Data Communication` data. A sizeable amount of simulations were performed for this effect. These simulations, which are not shown in this document, served both to validate the model and to consolidate several values for some of the parameters, as specified below.

Parameter	**Fixed Value**
Time Slot Duration	$\Delta = 5\,min = 300\,s$
Time-frame	$T = 60$
Maximum Battery	$B_{max} = 648\,\mathbf{J} \longrightarrow 100\,\%$
Minimum Battery	$B_{min} = 0.1 \cdot B_{max} = 64.8\,\mathbf{J} \longrightarrow 10\,\%$ of B_{max}
Initial Battery	$B_i(0) = 0.5 \cdot B_{max} = 324\,\mathbf{J} \longrightarrow 50\,\%$ of B_{max} *for all sensor nodes*
Consumption when in sleeping state	$C^{\mathtt{z}} = 0.63\,\mathbf{mJ} \longrightarrow 0.00009722\,\%$ of B_{max} *per time slot*
Consumption when in sensing state	$C^{\mathtt{s}} = 7.67\,\mathbf{mJ} \longrightarrow 1.18333333\,\%$ of B_{max} *per time slot*
Consumption when in transmitting data state	$C^{\mathtt{Tx}} = 7.63\,\mathbf{J} \longrightarrow 1.17718056\,\%$ of B_{max} *per time slot*
Consumption when in receiving data state	$C^{\mathtt{Rx}} = C^{\mathtt{Tx}} = 7.63\,\mathbf{J} \longrightarrow 1.17718056\,\%$ of B_{max} *per time slot*
Consumption when transmitting a data packet	$C_h^{\mathtt{Tx}} = 217.96\,\mathbf{mJ} \longrightarrow 0.03363578\,\%$ of B_{max} *per data packet*
Consumption when transmitting a data payload	$C_h^{\mathtt{Rx}} = 375.85\,\mathbf{mJ} \longrightarrow 0.05800218\,\%$ of B_{max} *per data payload*
Consumption when receiving a data packet	$C_p^{\mathtt{Tx}} = 9.46\,\mathbf{mJ} \longrightarrow 0.00146047\,\%$ of B_{max} *per data packet*
Consumption when receiving a data payload	$C_p^{\mathtt{Rx}} = 6.28\,\mathbf{mJ} \longrightarrow 0.00096885\,\%$ of B_{max} *per data payload*
Number of Nodes	$N = 25$
Network Topology	Square grid (Figure B.1)
Sink Node	Node with maximum closeness centrality
Maximum data packet size	$P_{max} = 15$ *data payloads*
Maximum node *data buffer* size	$Q_{max} = 6000$ *data payloads*
Minimum *data payloads* per period	$\beta = 1$ *data payload per sensor node*
Minimum *sensings* per period	$\mathtt{s}_{min} = 1$ *each node*
Minimum transmissions per period	$\mathtt{Tx}_{min} = 1$ *each node*
Network data collection ratio	$\gamma = 0.1$ *of total data*
Initial *data payloads* inside nodes	$Q_i(0) = 2$ *in every sensor node*
Events duration	$\delta = 4$ *time slots*
Events proximity	Always nearby to at least 1 sensor node
Events probability	Spawn at $p_E = 36\,\%$ of the time slots: $W(A = 3.11, B = 3)$
Event randomness	No randomness, always the same events distribution

Table 6.1: Fixed simulation parameters

Time

As we assume a time-slotted system with a finite-time operation of T *time slots*, it is essential first to define the duration of each of the *time slots*, since all other parameters depend on it – namely the harvesting and consumption profiles.

In an effort to comprehend how similar works tackle this matter, it is presented in Table 6.2 a brief compilation of the data parameters used in the past couple years' simulations about resource allocation on Energy Harvesting Networks. It was clear when reading those works that every research group defines the time slot duration or the timeframe span as the most suitable for the type of simulation and for the results that they are trying to achieve.

Hence, as our goal is to attain a certain sense of realism in the potential uses for this sort of networks, 5 minutes per time slot is a suitable value. This value respects the assumptions made on the theoretical model of Chapter 3, where it is declared that the nodes' operational states require sufficient time to be performed in their entirety before a new time slot initiates. Considering that the active and more consuming states (i.e., sensing, receiving and transmitting) represent operations that an embedded micro-controller can ordinarily perform in less than 2 seconds, 5 minutes gives a substantial operational margin and is proper for most applications of an IoT network with environment-sensing capabilities.

It should be remarked that the model does not use this value directly, but implicitly in the energy-related parameters. See next section or Appendix A.

Work	**Time Slot Duration**	**Time-frame Span**	T
This Work	5 minutes	5 hours	60
Baghaee *et al.* (2018) [63]	1 second	120 seconds	120
Zhang *et al.* (2018) [64]	1 second	100 seconds	100
Al-Tous *et al.* (2018) [65]	1 ms	100 ms	100
Dehwah *et al.* (2017) [25]	2 hours	12 hours	6
Kosunalp (2017) [30]	1 hours	24 hours	24
Jackson *et al.* (2017) [26]	30 minutes	24 hours	48

Table 6.2: Related works time data comparison (sorted by year)

For choosing a value for T, we had to regard simulation time: we found that simulation times raised exponentially with the value of the selected *time-frame*. As we had to perform

several simulations, the value was fixed at 60 because it proved to be a right balance between simulation time and the exposed network dynamics.

Energy Profiles

The contemplated sensor nodes are based on the Atmel XPlained Pro evaluation board which incorporates an Atmel 8-bit AVR Microcontroller ATmega256RFR2 with a *low power* 2.4GHz IEEE 802.15.4 compliant *radio transceiver* [66]. These evaluation boards also come with a mounted *temperature sensor* that can be considered the data acquisition module of the model. A typical rechargeable 3.0V/60mAh Lithium-Ion battery powers the board.

Appendix A exhibits all the premises and calculations that established the nodes energy related data.

Network Topology

For the network topology, we can consider two approaches:

- **The random network topology** – In this type of topology, N nodes are arbitrarily distributed across a bi-dimensional area. This approach is commonly used in related works as performance evaluation [53, 67]. The exact value of this considered area is not essential for the model, only the linked graph that results from connecting the nodes. These connections depend on the range radius defined for the attached radio transceivers.

  The **NetVis** app has the option to recreate such kind of topology with the help of an algorithm explicitly created for this task: it can vary the sparsity of the distributed nodes (more sparsity creates a looser network, less sparsity creates a tighter network) and the connectivity between them, simulating the variable range of the radio transceivers (more connectivity increases the number of neighbors of the nodes, less connectivity decreases the average amount of links between nodes).

  This kind of topology is the one which mirrors best how a network would exist in a practical IoT setting, but unfortunately it can not serve to our immediate evaluation purposes due to the always changing connectivity between simulations: the number of links between nodes varies, and more importantly, the number of links to the *sink node* also differ. See Figure B.4 for examples. Furthermore, as the evaluation metrics are dependent on the network structure, a lengthy analysis would need to include

averaged results from multiple simulations, making this assessment recommendable as future work (see Chapter 7).

- **The square grid network topology** – A square grid network is considered to maintain a consistent and perpetual evaluation topology: the placement of the nodes on a 2D area follows a square pattern (see Figure B.1 as an example). This works best when the number of nodes in the network N is a perfect square. As the connections between nodes are perpetually the same, it is the most natural topology in which to evaluate metrics when varying other parameters. This kind of topology features four sensor nodes connected to the *sink node*, and most of the nodes have four transmission links to other nodes, except those in the periphery of the network.

In any of the cases, the *sink node* is always the node on the network with maximum closeness centrality[1], i.e., the *sink node* stands in the closest position to all other nodes.

The model was experimented with both random and square topologies and with a variable number of nodes on the network – from 9 to 100 nodes. Nonetheless, although the model works out for a sizable amount of network nodes (which is good as IoT networks tend to have more and more nodes) we had to consider simulation running times – more nodes implies larger times waiting for optimization results –, and the size of the exhibited graphical data – after a certain number of nodes, the displayed information in such restricted canvas is visually challenging.

Thus, for evaluation purposes in this dissertation, we fixed N to be 25 along with a square topology – see Figure B.1. This choice satisfies what is expected from a network with a modest number of nodes while enabling us to assess the performance of nodes that remain at varying distances from the *sink node*. In Figure 6.4 it can be noticed that there are nodes up to four hops away and that the *sink node* is the central node.

The set of wireless links $\mathcal{L}$ in this topology contains 40 links that work in both directions. The contemplated transmitter radius for every node is sufficient to encircle its closer nodes. This radius along with the nodes placement results in a non-overlapping graph, in contrast with the examples in Figures B.4b, B.4c, and B.4d.

In short, we consider a 25-node square network with a central *sink node* with 40 bidirectional non-overlapping links. This decision further fixes the values for the distances to the

[1] In a connected graph, closeness centrality of a node is a measure of centrality in a network, calculated as the reciprocal of the sum of the length of the shortest paths between the node and all other nodes in the graph. Thus, the more central a node is, the closer it is to all other nodes. [68]

sink node D_i.

Nodes Operation and Data Communication

As asserted earlier in this document (`Nodes Operation and Data Communication` Table, Section 4.2.1), the analyzed set of four operational states of any particular node is $\mathcal{M} = \{\mathtt{z}, \mathtt{s}, \mathtt{Tx}, \mathtt{Rx}\}$ – sleeping $\mathtt{z}$, sensing $\mathtt{s}$, transmitting $\mathtt{Tx}$, or receiving $\mathtt{Rx}$.

As Riker et al. (2015) [44], we consider a node-to-node communication protocol stack composed of Constrained Application Protocol (CoAP) [69], User Datagram Protocol (UDP), 6LowPAN [70], and IEEE 802.15.4. The IEEE 802.15.4, 6LowPAN and UDP protocols can produce headers of varying sizes [44]. Considering a typical scenario, the aggregated headers of this protocol stack reach 64 bytes. The volume of a single CoAP payload is 4 bytes. This payload is used to store a data reading or the aggregated payload value. Taking into account the maximum limit of 127 bytes that IEEE 802.15.4 imposes and the Frame Check Sequence (2 bytes), the considered *data packets* are limited to contain a maximum of 15 *payloads* (i.e., 60 bytes).

The internal SRAM for the ATmega256RFR2 can go up to 32KB [66]. We acknowledge a fraction of this memory to be used as a node *data buffer*: 24KB. This value results in having enough memory space for 6000 4-byte-sized *data payloads*, therefore this does not constraint in any practical way the contemplated model.

After an exhaustive amount of simulation experiments with different values for some network parameters, we fixed the following data in subsequent performance tests, as the behavior of the network was in concordance to our expectations. The data and respective values are: $\beta = 1$, $\mathtt{Tx}_{min} = 1$, $\mathtt{s}_{min} = 1$, $\gamma = 0.1$, and $Q_i(0) = 2, \forall i \in \mathcal{S}$.

In detail, the finite-horizon operation time T is subdivided into periods of α *time slots* each. In each of those periods the *sink node* must receive at least $(N-1) \cdot \beta$ *data payloads* – as N is already fixed at 25, this results in a minimum of 24 *data payloads* into the *sink node* per period. In each one of these periods, a sensor node must perform a sensing operation ($\mathtt{s}_{min}$) and transmit data ($\mathtt{Tx}_{min}$) to other nodes at least once. In average, at the end of each period, at least 90 % (i.e.,$(1-\gamma)$) of all the created *data payloads* on the network in that period must be inside the *sink node* – this ensures that the sensor nodes actively work to get the data to neighbors closer to the sink as quickly as they can. Every sensor node starts its operation with two *data payloads* inside their *buffers* ($Q_i(0) = 2, \forall i \in \mathcal{S}$), as to replicate a practical instance where the network is launching a new operation cycle after concluding

a previous one.

Events

Events are distributed throughout space and time. The temporal distribution – events inter-arrival times –, as Ren *et al.* [62] proposed in 2014, can be modeled through a Weibull distribution $W(A, B)$. For their experiments, they use $A = 40$ and $B = 3$ as the parameters values.

Parameter B is related to the rate at which the events occur and, when is superior to 1 – as is this case where $B = 3$ –, the appearance probability of an event rises with time, i.e., the events inter-arrival times decrease over time. We use this value of B as well, and for this value, the mean of a Weibull random variable is given by $0.893A$.

Parameter A depends on the considered finite-horizon T and, for that reason, we can not use the value $A = 40$. We decided to specify this value according to the average amount of events that we wanted in a simulation, given by a fraction of the total number of *time slots* T. In other words, if we want, in average, n_E events in a simulation with T *time slots*, then we will have a fraction p_E of all the *time slots* T in which events will start happening: $n_E = p_E \cdot T$. Furthermore, for n_E events, the average inter-arrival time between them is $\frac{T}{n_E}$. This is equivalent to the previous value, so $0.893A = \frac{T}{n_E} \iff A = \frac{1}{0.893 \cdot p_E}$. Therefore, to obtain the right value for A in a simulation with any finite-horizon, we just need to input the value of p_E, e.g., if $p_E = 0.36$ we will obtain Weibull-randomly distributed events in 36 % of the *time slots*.

The model (Section 4.2) receives as event data $E_i(t)$, $v(t)$, and δ. The first two parameters are obtained after the **NetVis** app processes the values inserted in the `Percentage of Events` *box* and in the `Proximity` *slider* (see Section B.4.4) as explained above. The spatial distribution of the events is random, but depends on the `Proximity` *slider* value. The duration of the events is given by the `Duration` box in the app settings.

As a final note, the same spatio-temporal distribution of the events can be replicated in simulations with the same finite-horizon T if the `Random` *checkbox* is unchecked.

6.2.3 Metrics

As mentioned in the previous section, the validation of the model was carried out gradually through numerous tests with varied parameters. These tests are considered implicitly in the scope of this dissertation, because without them, the model would not have evolved

to the present point.

The performance assertion to the model is built on top of the validation, since no performance can be considered without having a valid model as a basis, and hence the considered metrics already include these two aspects as a whole.

These evaluation metrics are:

- The fraction of the total time that a node is in a specific operational state, in number of *time slots* in T. This metric allows us to compare the activity of the network as a whole for various energy conditions and to evaluate the particular performance of nodes at various distances from the *sink node*.
- The evolution of the energy status of each node over time, so it is possible to appraise the energy-neutral operation and to examine the network response to various harvesting profiles.
- The data communication distribution in all the existing network multi-hop links. This metric allows us to understand the adopted data routes when in variable energy conditions so the network throughput is not compromised.
- The average data aggregation as a function of distance to the *sink* to verify that this mechanism works with the goal of reducing energy waste.

With these performance metrics in mind, we extract the relevant graphical results from simulation data with the help of **NetVis**, as it is exhibited in the next section.

6.3 Results

To test the model we evaluate two significant aspects:

1. Network behavior across six different harvesting profiles. The primary goal of this aspect is to verify that the network is able to operate with diverse external harvesting situations and how its modus operandi depends on the harvesting outline. All the contemplated harvesting profiles have a lower average harvesting value per *time slot* than the suggested optimal value calculated in Appendix A, so we have an evaluation safety margin.

   Two essential segments differentiate the performed tests in the context of harvesting dynamics:

   (a) Constant harvesting values throughout all network operation. Consequently, the

overall harvesting average matches the harvesting value per *time slot*. See Figure 6.1 top left plot as an example.

(b) Variable harvesting values for the duration of the *time-frame*, dubbed dynamic harvesting. The considered profiles resemble a 5-hours generic solar energy acquisition with real hardware – i.e., see Figure 6.1 top right plot, where the harvested energy is near zero at the start and end of the operation, but peaks somewhere in the middle of it. In any case, these profiles and the constant ones share the same values of averaged harvesting per *time slot*.

Regarding the actual harvesting values, three situations are presented (all the values below are in percentage of the considered B_{max}):

(c) A low value with a mean of 0.9% per *time slot* – e.g., see Figure 6.1 top left plot –, which may correspond to a mostly cloudy period. This situation will be further referenced as *low harvesting*.

(d) An optimistic value averaging at 1.2% per *time slot* – e.g., see Figure 6.1 bottom left plot – that may be compatible with a sunny interval. This condition will be further labeled as *high harvesting*.

(e) And a median value between the two above: 1.05% per *time slot*, e.g., Figure 6.1 middle left plot. This situation is dubbed as *medium harvesting* from now on.

The range of these values may seem rather short, but as we are contemplating solar harvesting with the Appendix A hardware characteristics, these are practicable. Important note: all sensing-capable network nodes receive the same amount of energy harvesting in each *time slot*.

2. Interdependence of network behavior with various degrees of operational freedom, controlled through the α parameter. The applied values for α are 5, 10, and 20. A low value of α, e.g., $\alpha = 5$, compel the network nodes to perform an information "flush" to the *sink* every five *time slots* resulting in a low degree of operational freedom. Keeping in mind that every sensorial node has to perform at least one sensing operation and one `Tx` operation in that interval (consequently the inner nodes must be in `Rx` another time slot, because they will receive data for sure), there is no much room for the nodes to have a greedy response when it comes to creating new *data payloads* by sensing. Inversely, a higher value of α will allow the network to decide its own nodes' states in a more significant fraction of the time periods, i.e., the network is less restricted.

In brief, to fulfill the stated goals in Section 6.1, we performed six simulations that cover

the combinatorial possibilities of the considered harvesting aspects, while holding the value of α as 5; and then two more simulations that cover the remaining examined α values while maintaining fixed the harvesting in the *dynamic medium profile*. Therefore, we obtain a total of eight simulation results, and, taking into account the metrics established in the previous section, we present the graphical results ordered in the following manner:

- Figure 6.1 to Figure 6.4 present comparisons between the six regarded harvesting profiles: the left columns contain the *constant harvesting* related data results, and the right columns contain the *dynamic harvesting* related data results; the top rows show the *low harvesting* data results, the middle rows the *medium harvesting*, and the bottom rows the *high harvesting* data results.
- Figure 6.5 shows side-by-side comparisons between the data originated from simulations with $\alpha = 5$ (left column) and with $\alpha = 10$ (right column). The behavior of the results for $\alpha = 20$ is identical to that of $\alpha = 10$, so we chose not to present them thoroughly.
- Figures 6.6 and 6.7 provide an overall comparison between metrics derived from the previous graphical results.

Before explaining the results, let us examine Figure B.8s, showing an identical network topology to the one considered in our simulations. It shows the minimum distances, in *hops* – i.e., wireless links –, of all the network nodes to the *sink node*. Blue nodes are 1-hop away, green ones at 2-hops, orange at 3-hops, and red nodes are at 4-hops. Some upcoming references to the distance of the nodes to the *sink node* may be performed by using the *hops* nomenclature.

The line plots in Figure 6.1 detail the evolution of the network energy levels over the 60 operational *time slots*. The left axis relates to the nodes' battery levels. The matching plots are the average battery of all nodes at the same *sink* distance – we opted not to display each node battery individually as the resulting plot would be confusing. This way each line is representative of nodes with the same characteristics since the network is a symmetrical grid. The entire network battery average also appears as a reference (thick dark-blue line). The right axis has a different scale – it is a restricted range because the harvesting values are much lower than the battery ones –, and it visually presents the harvesting levels received by the nodes throughout the entire operation. The overall harvesting average is featured to have a visual cue when comparing constant versus dynamic harvesting. Naturally, when the harvesting is constant, the two lines will overlap. The values in the scales represent

percentage of B_{max}.

The distribution of the operational status of each node expressed in fraction of the *time-frame* appears in the bar plots of Figure 6.2. The sum of the parts of each bar is 1, and each color characterizes one of the four possible states (see respective legend). Illustrative explanation: if a bar is equally divided in four (25% each part), then, of the 60 operational time slots, the corresponding node would spend a total of 15 *time slots* in each of the states: sleeping, sensing, `Tx` and `Rx` – the exact order is not considered here. The horizontal axis presents the nodes ordered by the distance to the *sink node* from left to right, and the bars corresponding to the same distance are grouped to facilitate examination. Inside each group, the nodes at the same distance are positioned arbitrarily, but each position links to the same node in every plot. This time, we opted for not perform an overall aggregation of the results by distance, as in the previous figure, since the purpose of this plot is to interpret the nodes dynamics as a whole and as being part of a group with similar attributes.

Figure 6.3 displays the averaged data aggregation rates in the performed communications as a chart. The horizontal axis, from left to right, shows the temporal evolution of the network operation in time slots. The vertical axis arranges all the nodes at the same distance to *sink* – closer at the bottom, farther at the top. The intersections between the values in the two axes are shaped like a cell, e.g., the left bottom cell presents the averaged data aggregation rate of all the *data packets* transmitted during the first time slot by the nodes 1-hop away from the *sink*. The aggregation rates are specified by a color saturation scale: more average aggregation means a darker color – until a maximum of 1 (100% of average data aggregation), and less average aggregation a lighter color – until a minimum of 0 (no transmission occurred during that time slot).

The network topology is presented in the map plots of Figure 6.4 with the purpose of exhibiting the activity (utility) level of all the network wireless links. They also promote the global utility of the network by presenting the model optimization objective: the sum of *payloads* that reach the *sink node* (the larger circle in the middle) during operation. The network links are bi-directional (but never in the same time slot: constraints (4.1) and (4.5) guarantee that) so the arrows show the route of the information flow in the respective link. Above the arrows, it is revealed in square brackets the total sum of all the *data packets* that passed through the corresponding link throughout all the network operation. The other indicated value represents the equivalent in *data payloads*. Additionally, the link color saturation highlights the links' activity. If a certain link does not appear it is because no activity happened in that link. The values inside the nodes are the number of *payloads* that

were left inside the respective nodes' *buffer* (no visible number relates to an empty *buffer*) at the end of the operation – therefore those *payloads* would not reach the *sink* in that operational *time-frame*.

Figure 6.5 reuses previous plot types to compare the network performance in all the evaluated metrics. The plots in the left column are the same as those in the middle right position in the previous four figures (*medium dynamic harvesting* with $\alpha = 5$) while the right column shows an identical harvesting profile, but with $\alpha = 10$.

In Figure 6.6 we have line plots to compare the network utility over time. The used metric is the number of *data payloads* that get to the *sink node*. The left-hand plot evaluates the effect of the considered harvesting profiles in the overall network utility. Performances with the three α values are assessed in the right plot. By having the two plots side by side with the same scale, we can globally compare the potentials of all the considered simulations.

We apply the same-scale reasoning in Figure 6.7 to evaluate the total averaged data aggregation rates between all the eight experimental settings. The values of the bars are equivalent to the agglomeration of the values of each line of the charts of Figure 6.3. The axes' information follows similar reasoning as the ones in Figure 6.2.

The ensuing section is devoted to the discussion of the graphical results presented from Figure 6.1 to Figure 6.7.

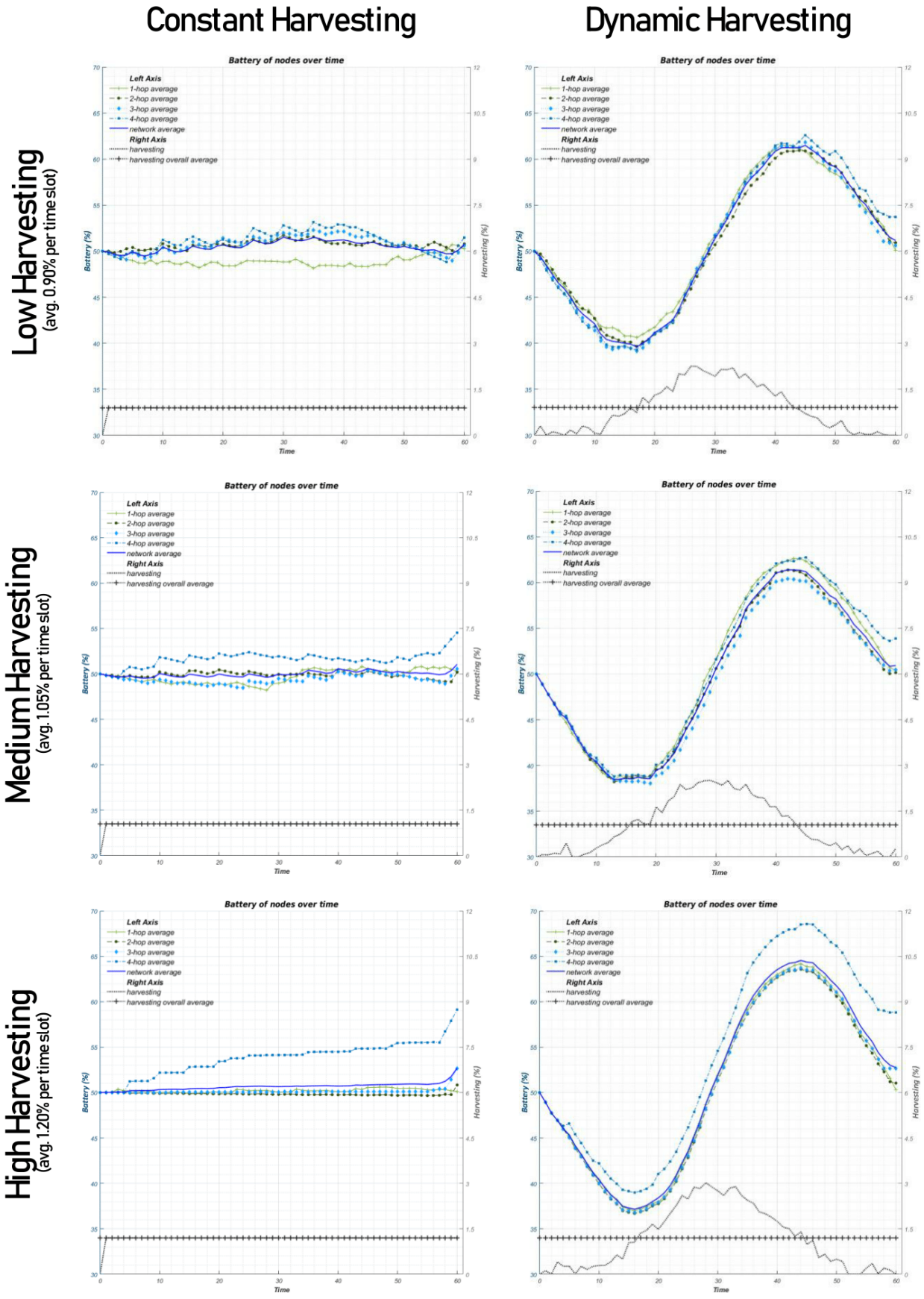


Figure 6.1: Battery of network nodes over time: harvesting profiles comparison

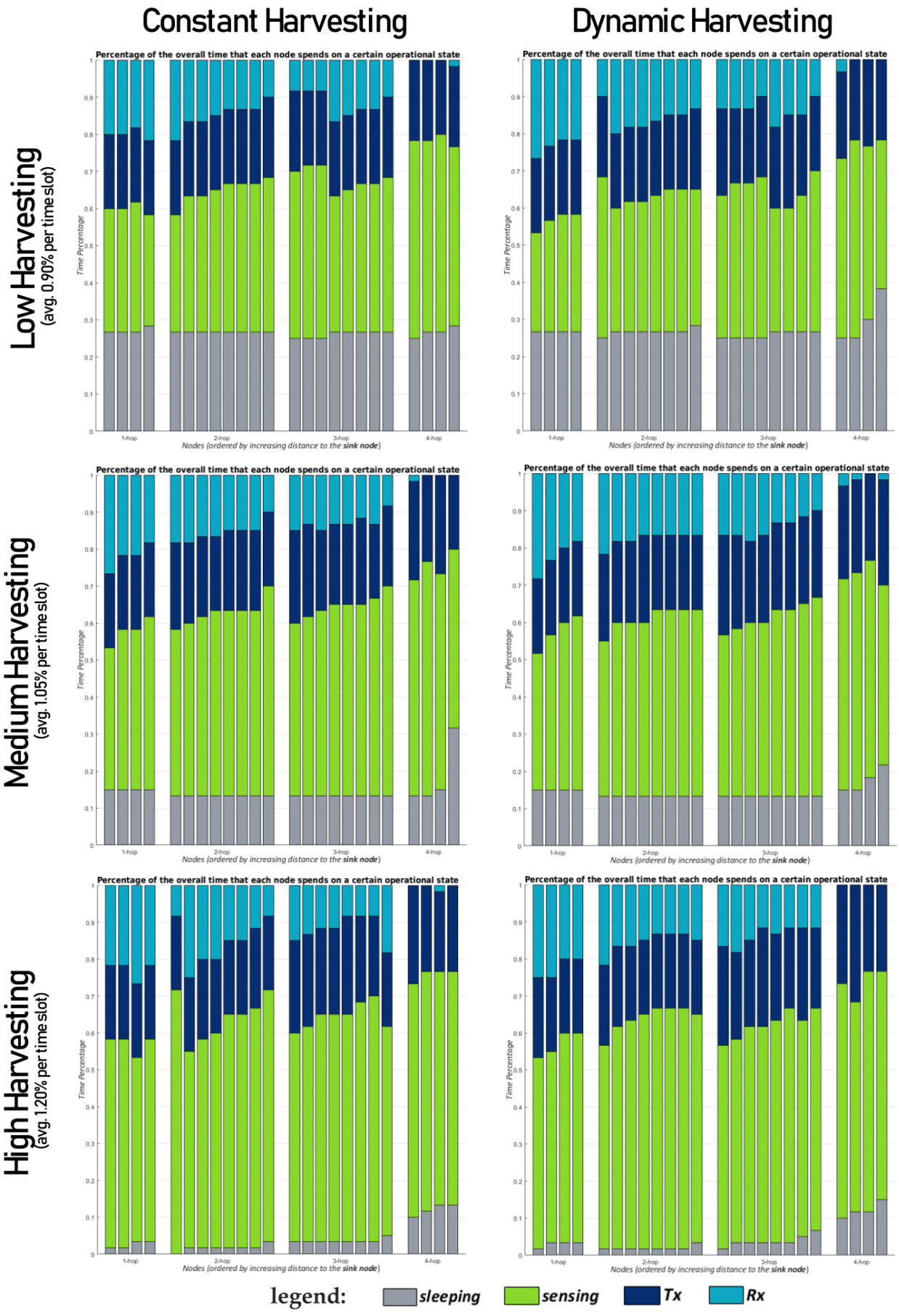


Figure 6.2: Operational status distribution by distance: harvesting profiles comparison

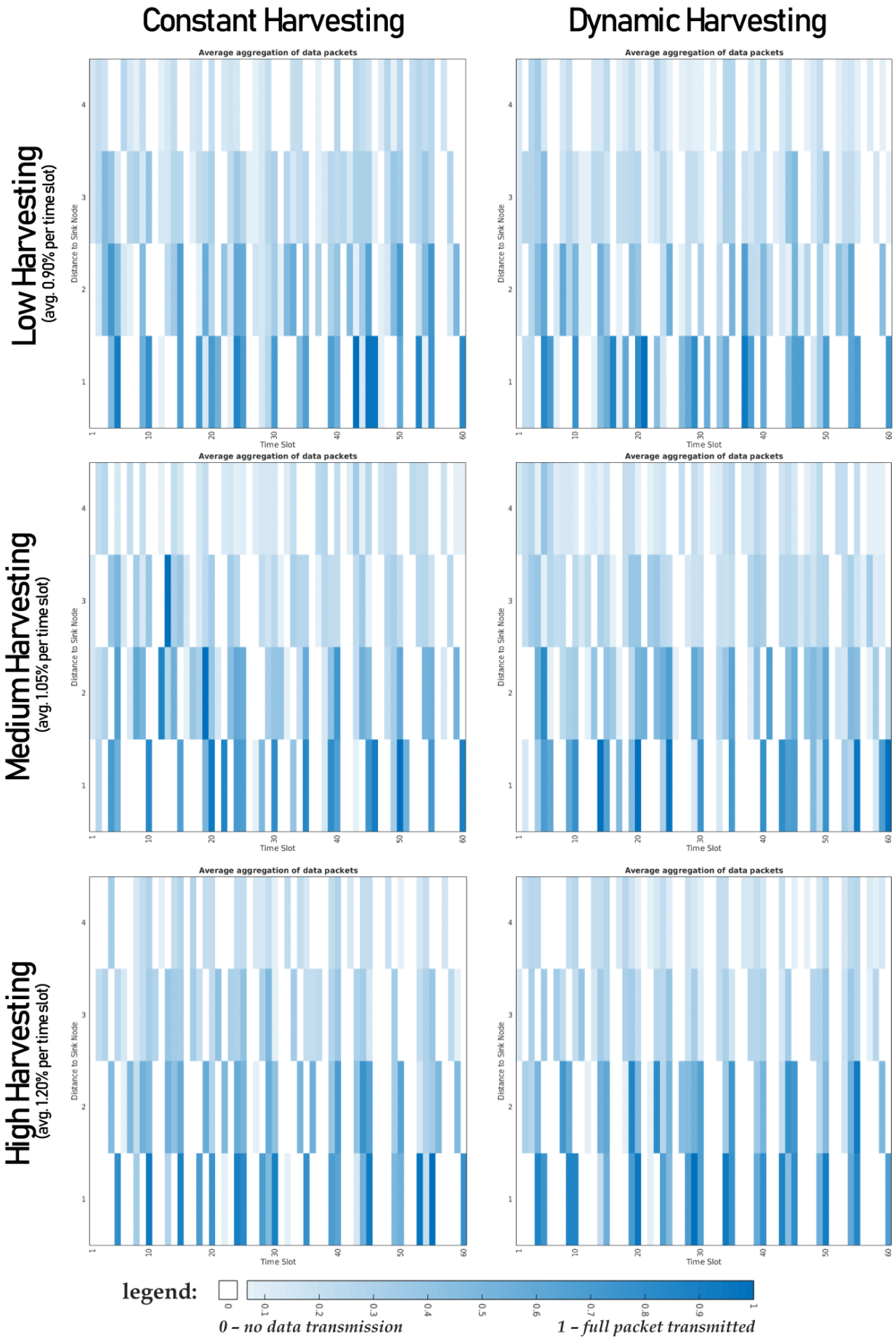


Figure 6.3: Average data aggregation per time slot: harvesting profiles comparison

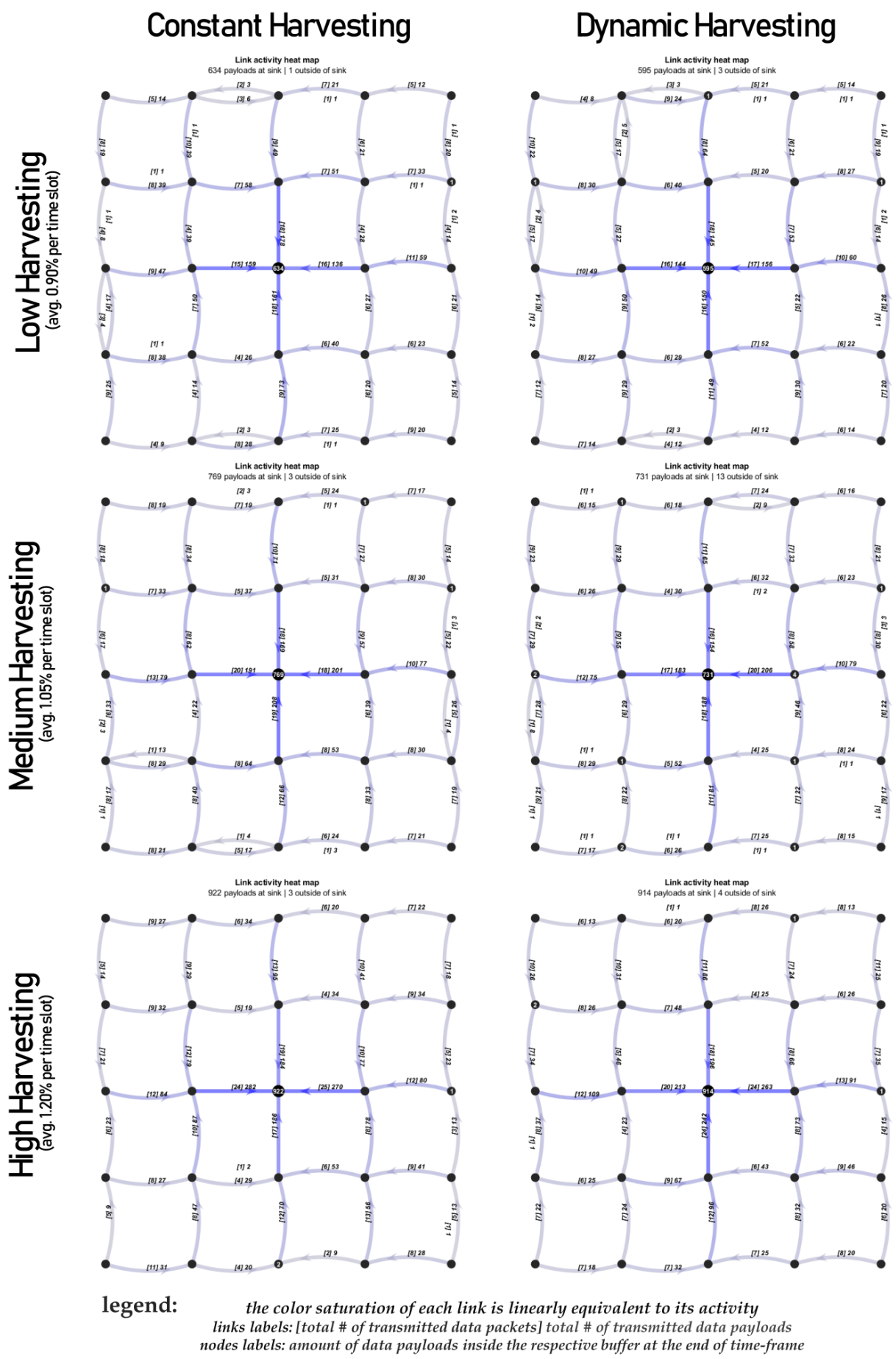


Figure 6.4: Link activity heat map: harvesting profiles comparison

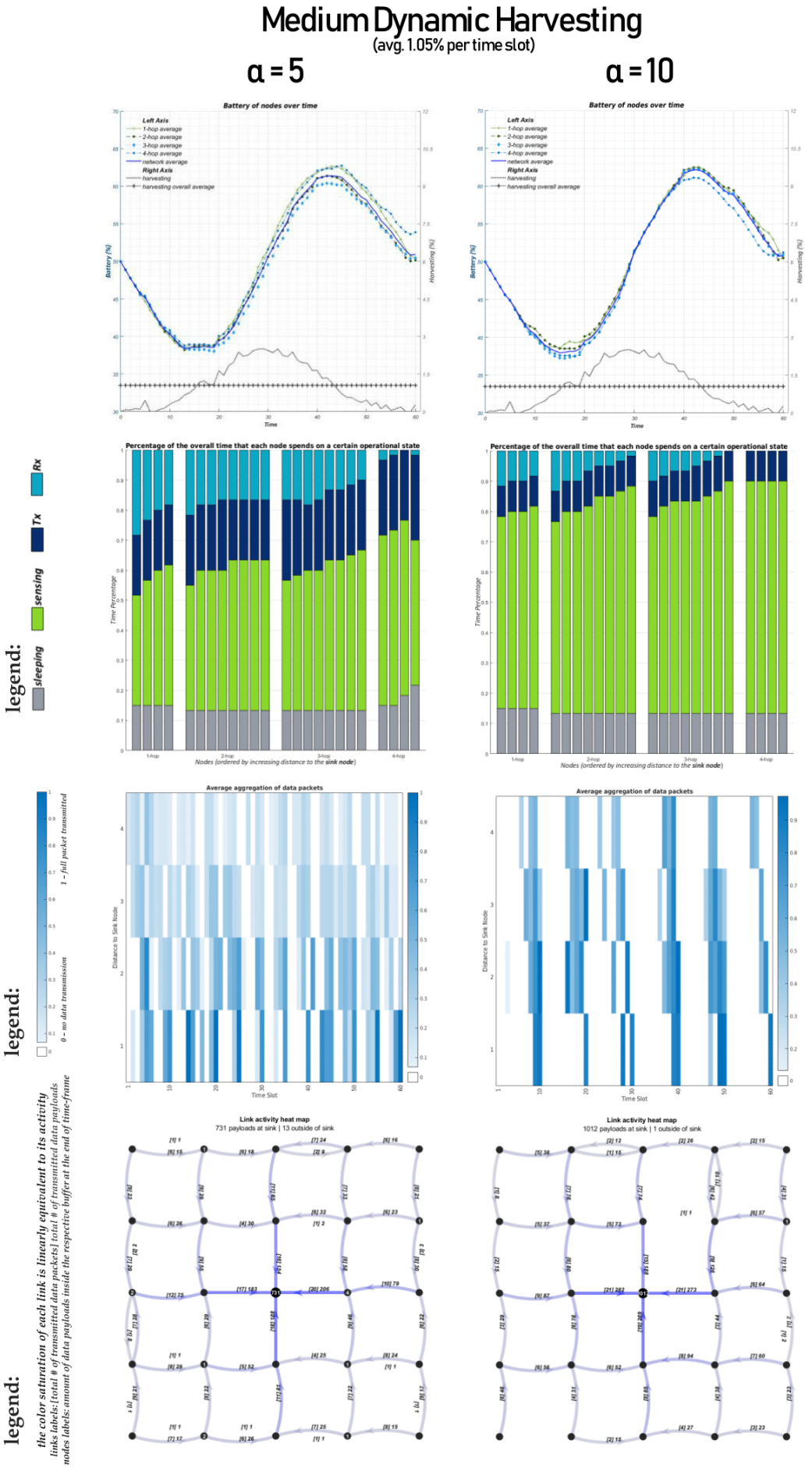


Figure 6.5: Alpha value enforcement: metrics comparison

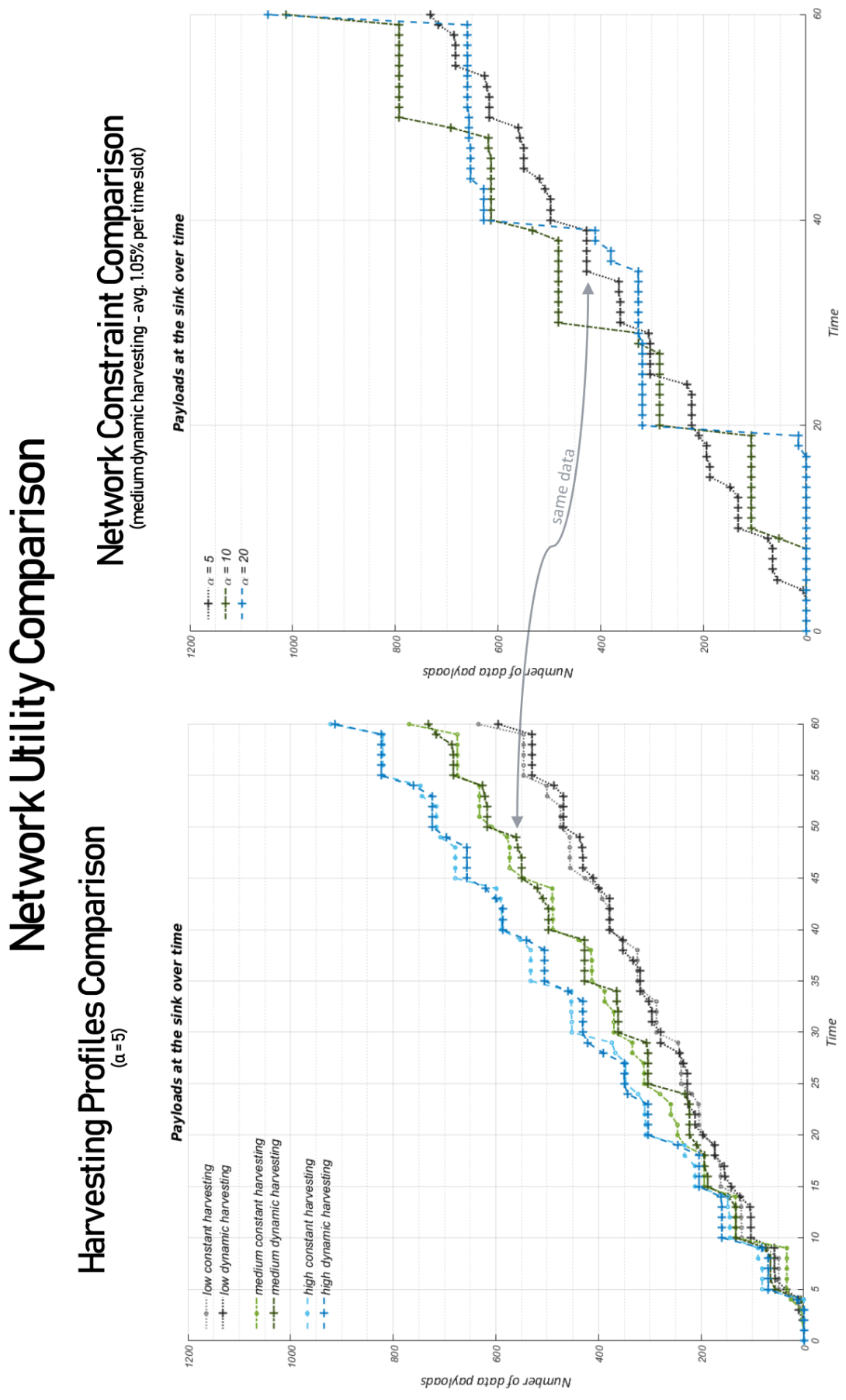


Figure 6.6: Network utility comparison: simulation settings

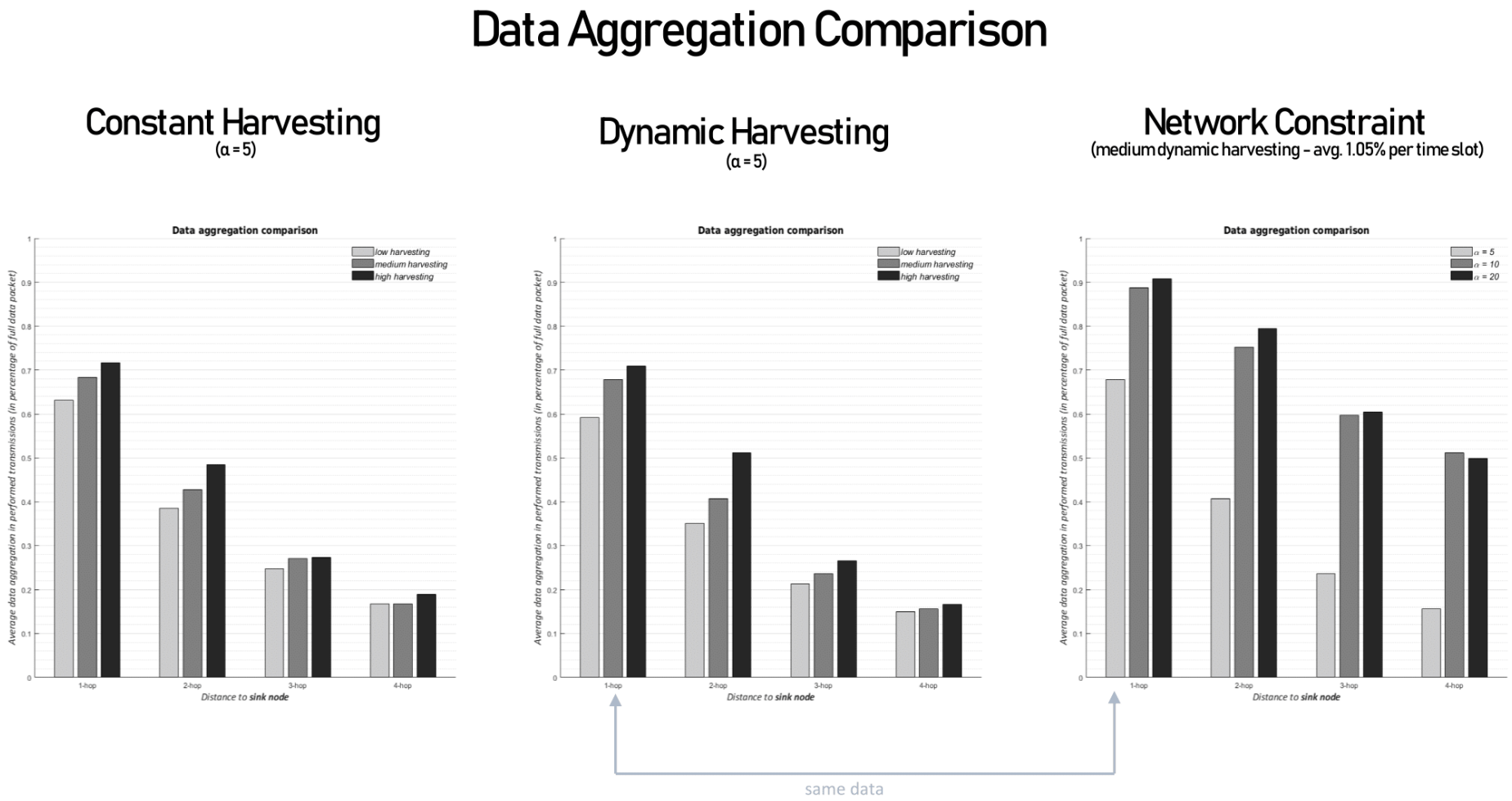


Figure 6.7: Data aggregation comparison: simulation settings

6.4 Results Discussion

The proposed model goals in Section 6.1 lead to the regarded evaluation metrics in Section 6.2.3. In turn, the plots were designed the way they are presented in the foregoing section to reflect on the appointed metrics. So now, in this section, we scrutinize the given plots to verify if the deducted results from them meet the dissertation targets stated in Chapter 2.3 and the model objectives in Section 6.1.

Energy Neutral Operation

The network maintains a neutral energy operation, which can be seen in all the demonstrations in Figure 6.1, although the harvesting profiles vary, both in profile and in average value. We verify ENO when the average power consumption of the network is less or equal than the average power harvested from the environment. Every node starts with 50% of battery and ends with more battery than the one with which it starts in all the considered cases. This behavior allows that the network nodes can always begin with an adequate battery level in following operational *time-frames*. If employing a similar energetic performance in those (which is guaranteed once again by the ENO constraint (4.16)), then it is guaranteed that the network can operate perpetually.

It is important to note that during the simulation there are nodes whose battery drops below the starting value – for instance, it can be seen in the 1-hop nodes in the first plot and in all the nodes in the dynamic harvesting plots. However, they can recover throughout the simulation and end up with as much energy as when it started. Note that we could increase the amount of data that can reach the *sink* bypassing the ENO constraint (4.16), but it would prevent the ability to keep the network running continually.

We can verify that, as might be expected, the lower the given average power, the more difficulties will have in getting nodes with a satisfactory battery margin above the initial value of 50%. Moreover, the network global average energy (thick dark-blue lines in the plots) tends to finish with even more surplus energy as the level of harvesting increases.

In general, the nodes that have more battery, on average, are the 4-hops nodes, characteristic that can be seen with more prominence as the harvesting level increases, being particularly notorious in *high constant harvesting*, where those nodes have 3% more battery than the rest, on average. This reaction is expected since the 4-hops nodes are the ones further away from the network center. Therefore they will not make many data receptions,

since sending data through these will lengthen the data path, which may not be beneficial in most operational conditions.

The analogous charts in Figure 6.2 confirm the deduction above. The 4-hops nodes enjoy a meager rate of Rx in comparison with all other nodes. While the remaining nodes are busy performing energy-expensive operations, 4-hops ones can decrease their power usage by running states that require less power such as sensing and sleeping. This advantage makes them accumulate more energy over time, and therefore they will always be the most robust nodes in cases of power failure. ENO can be easy to achieve for those nodes, contrarily to the internal ones, which may struggle to keep above the ENO line. Despite critical battery situations in the inner nodes may arise, the 4-hops nodes will almost never be used to route information – as verified in the links in their direction in Figure 6.4.

Network Utility

When comparing network utility in the simulations where constant harvesting was provided versus the dynamic harvesting ones, we can report that there is no practical difference between the two situations, even if the battery profiles over time are entirely unrelated. The left plot in Figure 6.6 helps to prove this statement, where it can be seen that every pair of lines with the same harvesting level follow a very similar network utility scheme.

Once again, an ENO is verified: the network optimizes the entirety of its operation by efficiently allocating the given resources. Let us analyze the dynamic harvesting profiles. The network is optimized by always using the right amount of energy, without operationally abusing the nodes when it has the most energy availability in the present to prevent future power outages; and operating with caution in cases of less access to energy, knowing that it can recover later when it is more accessible.

Here, of course, the model works because it is deterministic: it knows beforehand what is the expected power availability and its profile at all times and can plan accordingly. A real-time online algorithm would find the task of adjusting the network operation to the available power much more difficult because it cannot accurately predict the upcoming amount of usable energy.

As harvesting increases, nodes can further decrease their energy level (the curves fall below 37% in the bottom right plot of Figure 6.1), since they know they will have next more energy to recover.

Further similarities between constant and dynamic harvesting simulations are found in the

operational bar charts of Figure 6.2: in each row, the matching distances present analogous operational ratios – the most discernible cases are the sensing ratios.

However, when evaluating Figure 6.2 row by row, an apparent operational pattern emerges: less average power availability equals to a decreased sensing ratio across all the distances. This action is the network response to conserve its energy, as long as it maintains the minimal services imposed by the combined α, $\mathtt{s}_{min}$, and $\mathtt{Tx}_{min}$ constraints ((4.10) to (4.13)).

The network operational dependence in the amount of harvested energy per time slot is notably significant, as an increase of just 0.03% of average harvesting per time slot results in a drop of almost 22% in the network sleeping average. This sleeping reduction results in an equivalent sensing increase since the `Tx` and `Rx` remain practically at the same levels when looking at each row. Therefore, the nodes create more *data payloads* when the energy availability is higher but keep the same communication ratio when transmitting the supplementary created data, meaning that the data aggregation level in communications has to increase.

This is confirmed by the left and middle bar graphs in Figure 6.7. The trend for the global aggregation level is to increase each harvesting level, being this more significant in the 1-hop and 2-hops nodes than the most distant ones.

Multi-hop Potential

Regarding the performance of data reception, it is clear that the inner nodes carry out more often this type of operation than any other, as they act as the only routes between the periphery and the network center, which in this case is the *sink*. Therefore these are the most critical nodes in the network and will be the first to fail in case of energy shortage. While more distant ones will be able to use other communication channels if any of the neighbors fail, there will be repercussions on all the utility of the network if any of the core nodes cease to function, since all the information will have to be routed by the remaining 1-hop nodes. This situation will, in turn, make these nodes extraordinarily vulnerable and to likely fail too. The darker arrows in Figure 6.4 graphs represent this vulnerability. Thus, in a real network setting, it is reasonable that the number of nodes with direct access to the *sink* should always be more significant to make the network more robust.

The fact that the network has multi-hop capabilities is convenient in some cases where a node needs to find an alternative route to send the data, although this topology is not the best one to evaluate these parameters because the symmetry of the network does not allow

to extract meaningful patterns. However, there are some cases in which the same link was used at different times to pass information in opposite directions. The model optimization objective is rewarded when data is pushed closer to the *sink* (constraint (4.13)). Still, the opposite happens in some cases, but the amount of data transmitted on these occasions is always minimal, which suggests that the nodes take this critical decision when they have no other choice, i.e., they have to satisfy some operational constraint such as (4.11).

The multi-hop feature is more interesting when analyzing the corresponding link activity graphs in Figure 6.5, as the $\alpha = 10$ related graph shows distinctly that the network deprecates two of their links. Is interesting to note that increasing the operational degrees of freedom of the network by rising the α values also may provide increased flexibility in the possible placement of the nodes in the field. That is, as some links are expected not to be used, the nodes that had those possible links are no more constrained by their existence and can be moved to a new placement provided that the existing connectivity is maintained.

Data Aggregation

Aggregation rates increase when transmissions are performed nearby the *sink node*. Several plots support this declaration: Figure 6.7 left-most bar plots, the lower darker lines of the charts in Figure 6.3, and even deducing from the operational distributions in Figure 6.2. However, it is difficult to correlate the harvesting profiles (constant versus dynamic) with the data aggregation patterns of Figure 6.3. It seems that the aggregation behavior is equally distributed over time. Nonetheless, a hidden pattern can be deduced. In most of the time periods given by α, the nodes wait for the final time slots of those periods to start the `Tx` operations in a cascade effect from the outside to the inside. By operating the nodes in such controlled way, the network will make the most use of the aggregation energy savings by increasingly accumulating *payloads* in closer nodes to *sink* over the time period duration. After all, this pattern becomes apparent when evaluating the data aggregation chart over time for the simulation with $\alpha = 10$ (Figure 6.5) – dark blue diagonal stripes appear across all the ten time slots period.

The statement above further suggests that increasing α also increases the overall network data aggregation rates. And it is true; in the right-most plot of Figure 6.7, we extract a value for this: *data packets* are, in average 30% more filled when doubling α from 5 to 10. However, further increasing the value of α will not benefit significantly data aggregation rates, since the network has limited physical resources, i.e., number of nodes.

When choosing a value for α, a trade-off must be considered. Increasing α to a certain extent will bring further data communication efficiencies due to the higher rates of data aggregation. Moreover, the network utility will also increase – the right plot of Figure 6.6 shows that 300 more *data payloads* are received when doubling the α value from 5 to 10. Decreasing α will tend to lower those metrics but the lifetime of the received *data payloads* in the *sink* is decreased, that is, they will reach the *sink* at faster rates.

In brief:

- Similar average harvesting levels, but with different profiles, do not result in very different behaviors in our tests: the network optimizes its resources at any moment taking into account the expected global value of energy and results in very similar utility (see Figure 6.6, left plot).
- Higher values of harvesting result in more sensing and data aggregation opportunities and therefore higher network utility (see Figure 6.6, left plot).
- When varying α, there is a trade-off between communication efficiency through data aggregation plus global network utility versus granularity in receiving *payloads* at the *sink* (see Figure 6.6, left plot, and Figure 6.7, left plot). However, increasing the α value beyond a certain threshold will not bring increased additional benefits.
- The network takes advantage of its multi-hop ability, but only uses longer routes for *data payloads* when it is unavoidable.
- Events are guaranteed to be captured by at least one node, even that this is not displayed in any graphical result. The hardcoded constraint (4.14) ensures that this happens.

These results show that all the all the proposed goals have been met: the network model achieves validity through testing and its overall behavior is studied.

7 Conclusions and Future Work

The fulfillment of this dissertation allowed to kick-start in the LCT the study of EH-IoTNs that embrace the following compelling IoT concepts: multi-hop connectivity, Energy Neutral Operation, data aggregation mechanisms, and event capture. The combined evaluation of such referred notions into a single work is something that had not been analyzed yet, although there are works in the area that operate around similar topics such as data aggregation and ENO. When validating the proposed optimization model constructed to maximize aggregated data utility in an EH-IoTN, we obtained results that ensure continuity of the process of disclosure of behavioral patterns in such a network.

However, this model also presents certain simplifications of what would be an actual operation of sensor networks of this type, such as:

- The aggregation mechanism is implicitly regarded as an existing underlying functional concept;
- It is assumed that no communication faults occur and the data is flawlessly transmitted whenever required;
- The model employs deterministic data, the solutions are computed offline and not in real-time; the model is centralized, that is, it knows all the network nodes status at every moment.

Still, a possible solution for tackling the downsides stated above is composed as follows:

- Continue to perform simulation tests with the MILP model along the **NetVis** framework, varying the experimental settings and extracting refined metrics;
- Obtain a more inclusive and expressive portrayal of the EH-IoTN behavior;
- Suggest a sub-optimal heuristic that mirrors the previously studied behaviors and guarantees that it will work in a reactive distributed manner.

Finally, build up a physical test setting that applies the developed algorithmic heuristic in order to prove that the proposed models are viable in a real environment.

8 Bibliography

[1] V. Barchetti, M. Senigalliesi, O. Vermesan, R. Bahr, T. Macchia, A. Gluhak, S. Hubert, S. Vallet Chevillard, and F. Clari, "Analysis on IoT Platforms Adoption Activities," *H2020 Work Programme - UNIFY-IoT Project - ICT-30-2015: Internet of Things and Platforms for Connected Smart Objects*, p. 54, 2017.

[2] M. Prauzek, J. Konecny, M. Borova, K. Janosova, J. Hlavica, and P. Musilek, "Energy Harvesting Sources, Storage Devices and System Topologies for Environmental Wireless Sensor Networks: A Review," *Sensors*, vol. 18, no. 8, p. 2446, 2018.

[3] U. Pešović, J. Mohorko, K. Benkic, and Z. Cucej, "Single-hop vs. Multi-hop – Energy efficiency analysis in wireless sensor networks," in *Telekomunikacioni forum TELFOR 2010*, Beograd, Srbija, 2010.

[4] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power Management in Energy Harvesting Sensor Networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. V, no. 4, pp. 1–35, 2007.

[5] A. Cerpa, J. Elson, M. Hamilton, J. Zhao, D. Estrin, and L. Girod, "Habitat Monitoring: Application Driver for Wireless Communications Technology," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 2, pp. 20–41, 2001.

[6] G. Tolle, D. Gay, W. Hong, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, and P. Buonadonna, "A Macroscope in the Redwoods," in *Proceedings of the 3rd international conference on Embedded networked sensor systems - SenSys '05*, 2005, p. 51.

[7] S. Chalasani and J. M. Conrad, "A Survey of Energy Harvesting Sources for Embedded Systems," in *IEEE SoutheastCon 2008*, 2008, pp. 442–447.

[8] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and Yield in

a Volcano Monitoring Sensor Network," in *OSDI '06 - The 7th symposium on Operating systems design and implementation*, 2006, pp. 381–396.

[9] M. Suzuki, S. Saruwatari, N. Kurata, and H. Morikawa, "A high-density earthquake monitoring system using wireless sensor networks," in *Proceedings of the 5th international conference on Embedded networked sensor systems - SenSys '07*, 2007, p. 373.

[10] L. Yu, N. Wang, and X. Meng, "Real-time Forest Fire Detection with Wireless Sensor Networks," *Proceedings - 2005 International Conference on Wireless Communications, Networking and Mobile Computing*, vol. 2, pp. 1214–1217, 2005.

[11] S. Sudevalayam and P. Kulkarni, "Energy Harvesting Sensor Nodes: Survey and Implications," *IEEE Communications Surveys and Tutorials*, vol. 13, no. 3, pp. 443–461, 2011.

[12] R. Cardell-Oliver, "ROPE: A Reactive, Opportunistic Protocol for Environment Monitoring Sensor Networks," in *Second IEEE Workshop on Embedded Networked Sensors, EmNetS-II*, vol. 2005, no. July, 2005, pp. 63–70.

[13] A. Gaglione, D. Rodenas-Herraiz, Y. Jia, S. Nawaz, E. Arroyo, C. Mascolo, K. Soga, and A. A. Seshia, "Energy Neutral Operation of Vibration Energy-Harvesting Sensor Networks for Bridge Applications," in *International Conference on Embedded Wireless Systems and Networks (EWSN)*, 2018, pp. 1–12.

[14] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A Wireless Sensor Network For Structural Monitoring," in *Proceedings of the 2nd international conference on Embedded networked sensor systems - SenSys '04*, 2004, p. 13.

[15] J. Paek, K. Chintalapudi, R. Govindan, J. Caffrey, and S. Masri, "A Wireless Sensor Network for Structural Health Monitoring: Performance and Experience," in *The Second IEEE Workshop on Embedded Networked Sensors, 2005. EmNetS-II.*, 2005, pp. 1–10.

[16] Y. Luo, L. Pu, and Y. Zhao, "RF Energy Harvesting Sensor Networks for Healthcare of Animals: Opportunities and Challenges," *arXiv preprint arXiv:1803.00106*, 2018.

[17] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications - WSNA '02*, 2002, p. 88.

[18] P. A. Gargano, D. H. Gilmore, F. A. Pace, and L. Weinstein, "Personal tracking and recovery system," *US Patent 5,629,678*, jan 1997.

[19] T. R. Halford and K. M. Chuggy, "Barrage Relay Networks," in *2010 Information Theory and Applications Workshop (ITA)*, 2010, pp. 153–160.

[20] J. Huang, Z. Chang, M. Atiquzzaman, Z. Han, and W. Saad, "Guest Editorial Special Issue on Wireless Energy Harvesting for Internet of Things," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2580–2584, 2018.

[21] EnOcean GmbH, "The True Cost of Batteries – why energy harvesting is the best power solution for wireless sensors," Tech. Rep. August, 2015. [Online]. Available: https://www.enocean.com/en/technology/white-papers/

[22] World Energy Council, "World Energy Resources: Waste to Energy 2016," Tech. Rep., 2016. [Online]. Available: https://www.worldenergy.org/wp-content/uploads/2017/03/WEResources{_}Waste{_}to{_}Energy{_}2016.pdf

[23] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An Analysis of a Large Scale Habitat Monitoring Application," in *Proceedings of the 2nd international conference on Embedded networked sensor systems - SenSys '04*, 2004, p. 214.

[24] S. Basagni, V. D. Valerio, D. Informatica, R. La, D. Informatica, R. La, D. Informatica, and R. La, "Harnessing HyDRO: Harvesting-aware Data ROuting for Underwater Wireless Sensor Networks," in *Mobihoc '18 - Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2018, pp. 271–279.

[25] A. H. Dehwah, J. S. Shamma, and C. G. Claudel, "A distributed routing scheme for energy management in solar powered sensor networks," *Ad Hoc Networks*, vol. 67, pp. 11–23, 2017.

[26] G. Jackson, S. Ciocoiu, and J. A. McCann, "Solar Energy Harvesting Optimization for Wireless Sensor Networks," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017.

[27] C. Wang, J. Li, Y. Yang, and F. Ye, "Combining Solar Energy Harvesting with Wireless Charging for Hybrid Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 3, pp. 560–576, 2018.

[28] P. D. Mitcheson, E. M. Yeatman, G. K. Rao, A. S. Holmes, and T. C. Green, "Energy Harvesting From Human and MachineMotion for Wireless Electronic Devices," *Proceedings of the IEEE*, vol. 96, no. 9, pp. 1457–1486, 2008.

[29] M. A. Abdelkareem, L. Xu, M. K. A. Ali, A. Elagouz, J. Mi, S. Guo, Y. Liu, and L. Zuo, "Vibration energy harvesting in automotive suspension system: A detailed review," *Applied Energy*, 2018.

[30] S. Kosunalp, "An energy prediction algorithm for wind-powered wireless sensor networks with energy harvesting," *Energy*, vol. 139, pp. 1275–1280, 2017.

[31] L. Mateu, C. Codrea, N. Lucas, M. Pollak, and P. Spies, "Energy Harvesting for Wireless Communication Systems Using Thermogenerators," in *21st Conference on Design of Circuits and Integrated Systems*, 2006.

[32] W. K. Seah, A. E. Zhi, and H. P. Tan, "Wireless Sensor Networks Powered by Ambient Energy Harvesting (WSN-HEAP) - Survey and challenges," *Proceedings of the 2009 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace and Electronic Systems Technology (Wireless VITAE 2009)*, no. June, pp. 1–5, 2009.

[33] F. K. Shaikh and S. Zeadally, "Energy harvesting in wireless sensor networks: A comprehensive review," *Renewable and Sustainable Energy Reviews*, vol. 55, pp. 1041–1054, 2016.

[34] P. S. Chindhi, H. P. Rajani, and G. B. Kalkhambkar, "A Review on Radio Frequency[RF] Energy Harvesting Systems," *JASC: Journal of Applied Science and Computations*, vol. 5, no. 8, pp. 211 – 222, 2018.

[35] T. Sanislav, S. Zeadally, G. D. Mois, and S. C. Folea, "Wireless energy harvesting: Empirical results and practical considerations for Internet of Things," *Journal of Network and Computer Applications*, vol. 121, pp. 149–158, 2018.

[36] K. S. Adu-manu, N. Adam, C. Tapparello, H. Ayatollahi, and W. Heinzelman, "Energy-Harvesting Wireless Sensor Networks (EH-WSNs): A Review," *ACM Transactions on Sensor Networks*, vol. 14, no. 2, p. 50, 2018.

[37] F. Engmann, F. A. Katsriku, J.-D. Abdulai, K. S. Adu-manu, and F. K. Banaseka, "Prolonging the Lifetime of Wireless Sensor Networks: A Review of Current Techniques," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–38, 2018.

[38] S. Yang, X. Yang, J. A. McCann, T. Zhang, G. Liu, and Z. Liu, "Distributed Networking in Autonomic Solar Powered Wireless Sensor Networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 750–761, 2013.

[39] Z. A. Eu, H.-P. Tan, and W. K. Seah, "Design and performance analysis of MAC schemes for Wireless Sensor Networks Powered by Ambient Energy Harvesting," *Ad Hoc Networks*, vol. 9, no. 3, pp. 300–323, 2011.

[40] L. Lin, N. B. Shroff, and R. Srikant, "Asymptotically Optimal Energy-Aware Routing for Multihop Wireless Networks with Renewable Energy Sources," *IEEE/ACM Transactions on Networking*, vol. 15, no. 5, pp. 1021–1034, 2007.

[41] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network Aggregation Techniques for Wireless Sensor Networks: A Survey," *IEEE Wireless Communications*, vol. 14, no. 2, pp. 70–87, 2007.

[42] E. F. Nakamura, A. A. F. Loureiro, and A. C. Frery, "Information fusion for wireless sensor networks," *ACM Computing Surveys*, vol. 39, no. 3, pp. 9–es, 2007.

[43] M. Amarlingam, P. K. Mishra, P. Rajalakshmi, S. S. Channappayya, and C. S. Sastry, "Novel Light Weight Compressed Data Aggregation using sparse measurements for IoT networks," *Journal of Network and Computer Applications*, vol. 121, pp. 119–134, 2018.

[44] A. Riker, E. Cerqueira, M. Curado, and E. Monteiro, "Data Aggregation for Machine-to-Machine Communication with Energy Harvesting," in *2015 IEEE International Workshop on Measurements & Networking (M&N)*, 2015, pp. 76–81.

[45] S. Guo, Y. Shi, Y. Yang, and B. Xiao, "Energy Efficiency Maximization in Mobile Wireless Energy Harvesting Sensor Networks," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2017.

[46] J. Pelegri-Sebastia, M. Gasulla, and G. Boggia, "Energy harvesting and management for distributed sensor networks," *International Journal of Distributed Sensor Networks*, 2017.

[47] L. Lin, N. B. Shroff, and R. Srikant, "Energy-Aware Routing in Sensor Networks: A Large Systems Approach," in *WONS 2006 : Third Annual Conference on Wireless On-demand Network Systems and Services*, 2006, pp. 159–169.

[48] S. Sarkar, M. H. R. Khouzani, and K. Kar, "Optimal Routing and Scheduling in Multi-hop Wireless Renewable Energy Networks." *IEEE Transactions on Automatic Control*, vol. 58, no. 7, pp. 1792–1798, 2013.

[49] S. Manfredi and E. Di Tucci, "Decentralized Control Algorithm for Fast Monitoring and Efficient Energy Consumption in Energy Harvesting Wireless Sensor Networks," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1513–1520, aug 2017.

[50] Y. Yi and S. Shakkottai, "Hop-by-Hop Congestion Control Over a Wireless Multi-Hop Network," *IEEE/ACM Transactions on Networking*, vol. 15, no. 1, pp. 133–144, 2007. [Online]. Available: http://users.ece.utexas.edu/{~}shakkott/Pubs/hopbyhop-submit.pdfhttp://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4100728

[51] S. Peng and C. P. Low, "Energy neutral directed diffusion for energy harvesting wireless sensor networks," *Computer Communications*, vol. 63, pp. 40–52, 2015.

[52] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes, "Powertrace : Network-level Power Profiling for Low-power Wireless Networks Low-power Wireless," Tech. Rep., 2011.

[53] M. Zareei, C. Vargas-Rosales, R. Villalpando-Hernandez, L. Azpilicueta, M. H. Anisi, and M. H. Rehmani, "The effects of an Adaptive and Distributed Transmission Power Control on the performance of energy harvesting sensor networks," *Computer Networks*, vol. 137, pp. 69–82, 2018.

[54] G. Jackson, Z. Qin, and J. A. McCann, "Long Term Sensing via Battery Health Adaptation," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 2240–2245.

[55] S. Yang, Y. Tahir, P.-y. Chen, A. Marshall, and J. McCann, "Distributed Optimization in Energy Harvesting Sensor Networks with Dynamic In-network Data Processing," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016, pp. 1–9.

[56] A. Riker, M. Curado, and E. Monteiro, "Neutral Operation of the Minimum Energy

Node in Energy-Harvesting Environments," in *2017 IEEE Symposium on Computers and Communications (ISCC)*, 2017, pp. 477–482.

[57] A. Mehrabi and K. Kim, "General Framework for Network Throughput Maximization in Sink-Based Energy Harvesting Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 7, pp. 1881–1896, 2017.

[58] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, "Cross-layer Congestion Control, Routing and Scheduling Design in Ad Hoc Wireless Networks," in *IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, 2006.

[59] M. Kodialam and T. Nandagopal, "Characterizing Achievable Rates in Multi-hop Wireless Networks: The Joint Routing and Scheduling Problem," in *MobiCom '03*. ACM, 2003, pp. 42–54.

[60] S. He, J. Chen, D. K. Yau, H. Shao, and Y. Sun, "Energy-Efficient Capture of Stochastic Events under Periodic Network Coverage and Coordinated Sleep," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 6, pp. 1090–1102, 2012.

[61] C.-f. H. C.-f. Hsin and M. L. M. Liu, "Network Coverage Using Low Duty-Cycled Sensors: Random & Coordinated Sleep Algorithms," in *Third International Symposium on Information Processing in Sensor Networks 2004 (IPSN 2004)*, 2004, pp. 433–442.

[62] Z. Ren, P. Cheng, J. Chen, D. K. Yau, and Y. Sun, "Dynamic Activation Policies for Event Capture in Rechargeable Sensor Network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3124–3134, 2014.

[63] S. Baghaee, S. Chamanian, H. Ulusan, and O. Zorlu, "WirelessEnergySim: A Discrete Event Simulator for an Energy-Neutral Operation of IoT Nodes," in *2018 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2018, pp. 1 – 5.

[64] X. Zhang, C. Wang, and L. Tao, "An Opportunistic Packet Forwarding for Energy-Harvesting Wireless Sensor Networks With Dynamic and Heterogeneous Duty Cycle," *IEEE Sensors Letters*, vol. 2, no. 3, 2018.

[65] H. Al-Tous and I. Barhumi, "MPC for Online Power Control in Energy Harvesting Sensor Networks," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, 2018.

[66] Atmel Corporation, "Atmel ATmega256/128/64RFR2 Summary Datasheet."

[67] I. Khan and D. Singh, "Energy-balance node-selection algorithm for heterogeneous wireless sensor networks," *ETRI Journal*, vol. 40, no. 4, 2018.

[68] W. contributors, "Closeness centrality," Reviewed in 2016. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Closeness{_}centrality{&}oldid=854898371

[69] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," *RFC 7252, Internet Engineering Task Force*, 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7252.txt

[70] C. Bormann, G. Mulligan, and T. Lemon, "IPv6 over Low power WPAN (6lowpan)," *RFC4944 Internet Engineering Task Force*, 2012. [Online]. Available: https://datatracker.ietf.org/wg/6lowpan/documents/

[71] Digi-Key Electronics, "Microchip Technology ATSAM4S-XPRO." [Online]. Available: https://www.digikey.com/product-detail/en/microchip-technology/ATSAM4S-XPRO/ATSAM4S-XPRO-ND/3906411

[72] A. Cavaco, H. Silva, P. Canhoto, S. Neves, J. Neto, and M. C. Pereira, "Annual Average Value of Solar Radiation and its Variability in Portugal," in *Workshop On Earth Sciences 2016 / Workshop em Ciências da Terra 2016*, 2016, p. 5.

[73] T. D. Lee and A. U. Ebong, "A review of thin film solar cell technologies and challenges," *Renewable and Sustainable Energy Reviews*, vol. 70, no. December 2016, pp. 1286–1297, 2017.

[74] Texas Instruments Inc., "eZ430-RF2500-SEH Solar Energy Harvesting Development Tool User's Guide," Tech. Rep., 2009. [Online]. Available: www.ti.com

Appendix A

Calculations for the Energy Input Data

A.1 Battery

The battery that powers the Atmel ATmega256RFR2 Xplained Pro board (similar as the one in Figure A.1a) is a rechargeable Lithium-Ion one rated to **3V/60mAh** (Figure A.3). Each *time slot* has 300s ($\mathbf{\Delta = 300s}$).

Since an `Amp Hour` (**Ah**) is a measure of charge (measured in `Coulombs` (**C**)) whereas a `Joule` (**J**) is a measure of energy, it is only possible to convert **mAh** to `Joules` by using the Voltage (`Volts` (**V**)) which the charge is transferred. So, by combining

$$\textbf{Charge}\,(\textbf{C}) \times \textbf{Voltage}\,(\textbf{V}) = \textbf{Energy}\,(\textbf{J})$$

and

$$\textbf{Current}\,(\textbf{A}) \times \textbf{time}\,(\textbf{seconds}) = \textbf{Charge}\,(\textbf{C})$$

we obtain

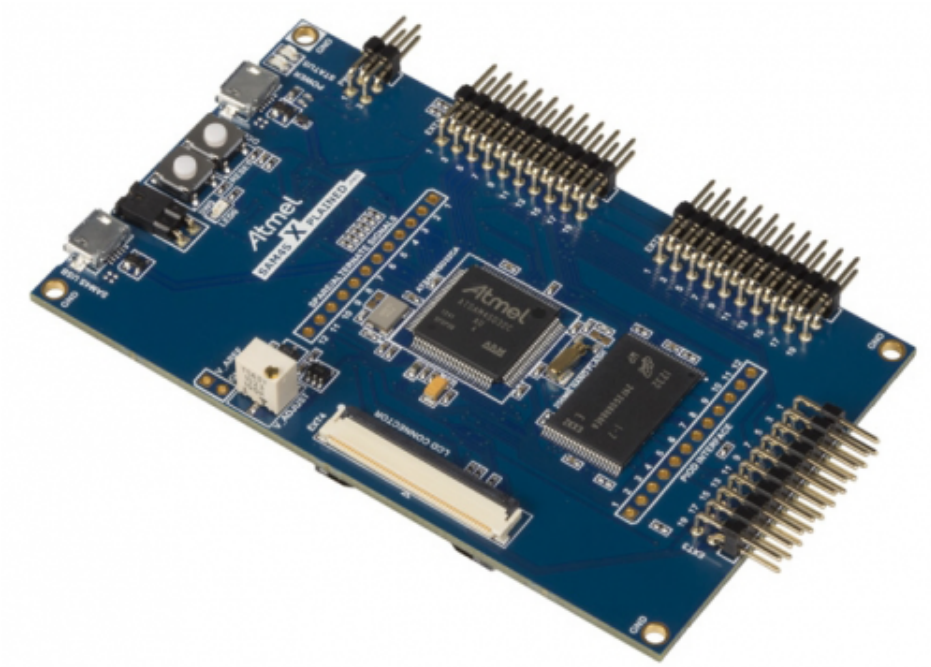
$$\textbf{Energy}\,(\textbf{J}) = \textbf{Current}\,(\textbf{A}) \times \textbf{time}\,(\textbf{seconds}) \times \textbf{Voltage}\,(\textbf{V})$$

The **60mA** delivered by the battery over 1h (**3600s**) at **3V** provide

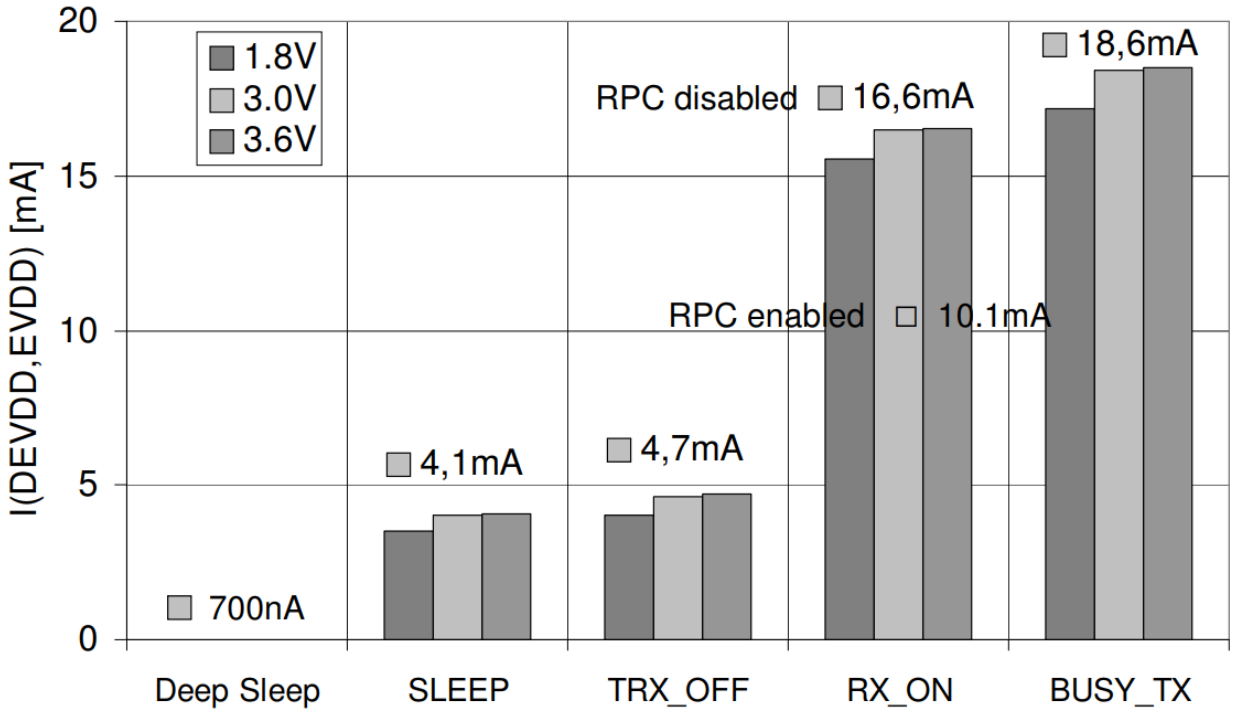
$$0.06\,\textbf{A} \times 3600\,\textbf{s} \times 3\,\textbf{V} = 648\,\textbf{J}$$

of energy. In other words, the battery can supply **648 J** before running out of energy. We assume that the battery performs ideally and in an adequate operating environment. Also, it is considered that the battery does not wear over time. Those 648 J of energy are the full battery, that is, 100% of the battery:

$$B_{max} = 648\,\textbf{J}$$



(a) The Atmel XPlained Pro evaluation board [71].



(b) Atmel ATmega256RFR2 Radio Transceiver and microcontroller (16MHz) supply current – from the datasheet [66]. Considering only the values referring to **3.0V**.

Figure A.1: The Atmel ATmega256RFR2 Xplained Pro evaluation kit is a hardware platform to evaluate the ATmega256RFR2 microcontroller and features a temperature sensor and a Low Power 2.4GHz IEEE 802.15.4 compliant Radio Transceiver.

A.2 Consumptions

The typical supply current of the Atmel 8-bit AVR Microcontroller ATmega256RFR2 with a Low Power 2.4GHz IEEE 802.15.4 compliant Radio Transceiver with CPU clock set to 16MHz is shown in Figure A.1b. Consider only the values regarding **3.0V**.

From it, it is possible to expose that in the idlest state, `Deep Sleep`, which we regard as the sleeping state in our model, an entire *time slot* (300s) consumes $C^{\mathtt{z}} = 700\,\mathbf{nA} \times 300\,\mathbf{s} \times 3\,\mathbf{V} = 0.63\,\mathbf{mJ}$. In other words, $0.63 \times 10^{-3}/B_{max} = 0.00009722\%$ of the battery is consumed per *time slot* when a node is sleeping.

The remaining of the datasheet values seem valuable but we performed experimental tests to achieve coherent values for these parameters, recurring to Powertrace [52], a network-level power profiling tool of low-power wireless systems. We found that in our setting, the consumption values differ from the nominal ones:

- When in sensing, we consider that the microcontroller is performing CPU operations, i.e., not idle, but not has the radio transceiver turned on. Also, in the test setting, performing sensing operations at a rate of 1 measurement per second, the microcontroller draws an average current of 8.52 **mA**. Then, sensing consumes $C^{\mathtt{s}} = 8.52\ \mathbf{mA} \times 300\ \mathbf{s} \times 3\ \mathbf{V} = 7.668\ \mathbf{J}$ of the battery energy during a *time slot*, which corresponds to $7.668/B_{max} = 1.18333333\%$ of the battery per *time slot*.

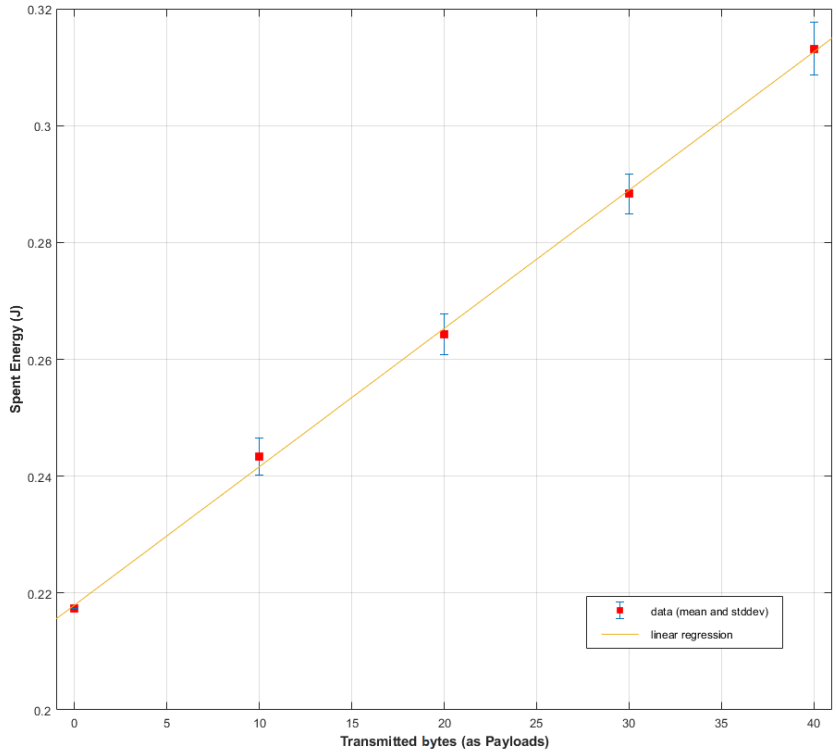
- For the operation states where there is data flowing, we first determined the average current that the board consumed when operating with the transmitter on. This value stands at 8.4757 **mA**. Therefore, there is a fixed consumption that is given by $C^{\mathtt{Tx}} = C^{\mathtt{Rx}} = 8.4757\ \mathbf{mA} \times 300\ \mathbf{s} \times 3\ \mathbf{V} = 7.62813\ \mathbf{J}$ or $7.62813/B_{max} = 1.17718056\%$ of the battery during a *time slot* in which the board is in the `Tx` and `Rx` states.

- Our model also discriminates the energy cost per sent/received data payload and data packet. Again, recurring to Powertrace, we were capable of isolating the power consumptions spikes that happened when the board sent and received fixed amounts of data: no data payloads (0 bytes), 10 bytes, 20 bytes, 30 bytes, and 40 bytes. The results are exhibited in Figure A.2. By performing linear regressions on the results, we determine that in practice:

  - The discrete energy consumption for transmitting 1 data packet (just the header) is $C_h^{\mathtt{Tx}} = 217.96\ \mathbf{mJ}$ (0.03363578% of B_{max}). This is the value of the linear

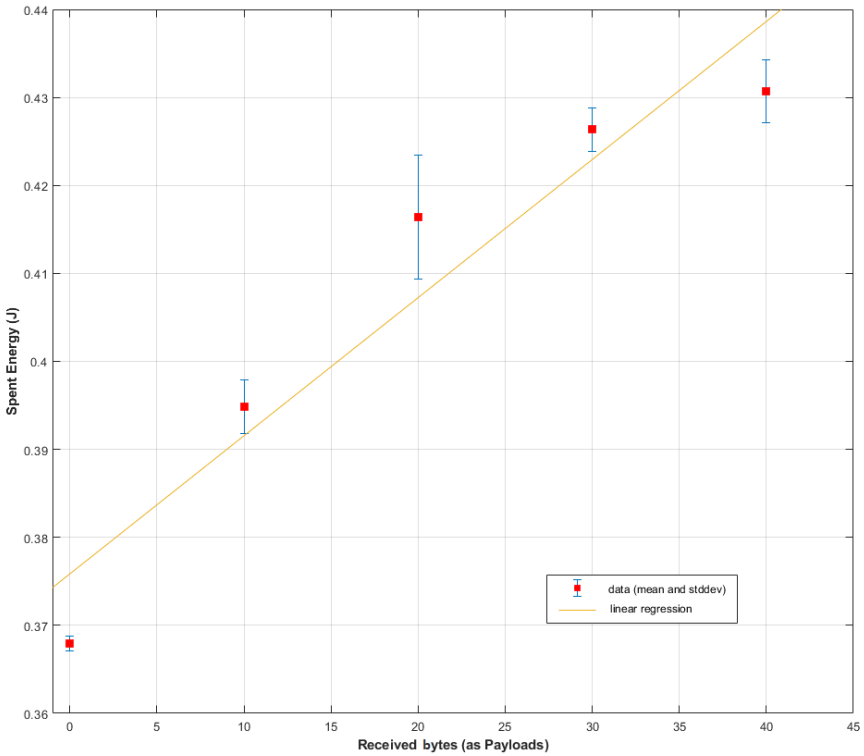
regression in Figure A.2a when we consider 0 bytes.

- A data payload has 4 bytes. The corresponding linear regression value from Figure A.2a is 227.42 **mJ**. Therefore, each additional data payload that is carried on a packet consumes $C_p^{\texttt{Tx}} = 227.42 - C_h^{\texttt{Tx}} = 9.46$ **mJ** (0.00146047% of B_{max}) on the sender node.
- The discrete energy consumption for receiving 1 data packet (just the header) is $C_h^{\texttt{Rx}} = 375.85$ **mJ** (0.05800218% of B_{max}). This is the value of the linear regression in Figure A.2b when we consider 0 bytes. Although the experimental values (Figure A.2b) are best fitted by a quadratic regression, our data packets are limited to 15 data payloads, meaning that we have more interest in approximating the results by a linear regression.
- A data payload has 4 bytes. The corresponding linear regression value from Figure A.2b is 382.13 **mJ**. Therefore, each additional data payload that is carried on a packet consumes $C_p^{\texttt{Rx}} = 382.13 - C_h^{\texttt{Rx}} = 6.28$ **mJ** (0.00096885% of B_{max}) on the receiver node.



(a) Values for transmitting data.

Linear regression: $\textbf{energy}\ (\textbf{J}) = 0.002366 \times \textbf{bytes} + 0.21796$



(b) Values for receiving data.

Linear regression: $\textbf{energy}\ (\textbf{J}) = 0.0015696 \times \textbf{bytes} + 0.37585$

Figure A.2: Experimental consumptions results with Powertrace [52] in the Atmel ATmega256RFR2 Xplained Pro evaluation kit from Figure A.1.

Figure A.3: Typical *Lithium-Ion* rechargeable battery. This one has a rated voltage of 3.7V.

A.3 Harvesting

The images in Figure A.4 are taken from the 2016 *Annual Average Value of Solar Radiation and its Variability in Portugal* [72]. From the values in the lower right-hand corner of Figure A.4a we observe that the Average Solar Irradiation for Coimbra through the year is 1600 $\mathbf{kWh/m^2}$. This unit of measurement – $\mathbf{kWh/m^2}$ – represents the amount of energy (measured in `kW-hours`) accumulated over 1 square meter in a year period. Of course, there is no sunlight each evening, so the values correspond to the energy absorbed during daylight hours.

We can expect an average of $1600/365 = 4.38\,\mathbf{kWh}$ of energy falling on an area of $1m^2$ over the period of one day. This is equivalent to $4380\,\mathbf{W} \times 3600\,\mathbf{seconds/hour} = 15.78\,\mathbf{MJ}$ per square meter per day. We can calculate how much of that energy can be converted to electrical form using the following equation:

$$E_{electrical} = E_{solar} \times \mathbf{Effective\ Area} \times PV_{efficiency}$$

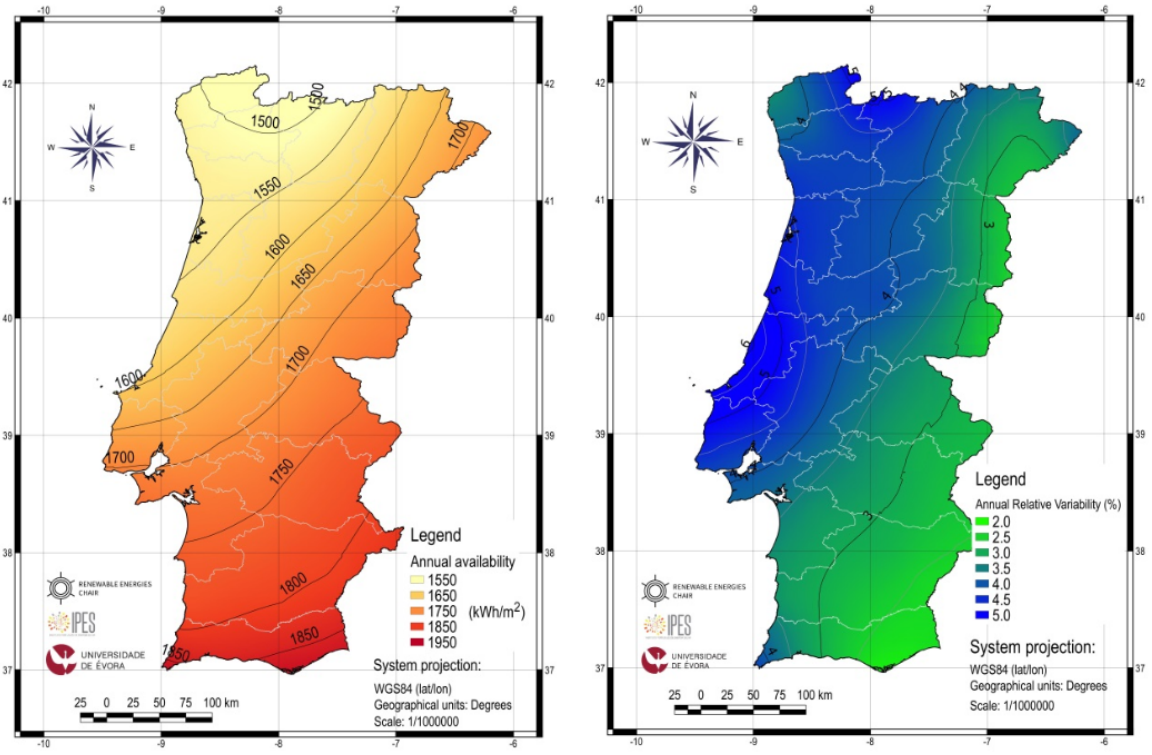
The average power available from the sun is

$$E_{solar} = \frac{15.78\,\mathbf{MJ}}{86400\,\mathbf{seconds/day}} = 183\,\mathbf{J/m^2/second}$$

`Effective area` is the surface area in the solar cell actually absorbing photons and is measured in m^2. $PV_{efficiency}$ quantifies the *Photo-Voltaic* cell's ability to convert light energy to current.

Thin film photo-voltaic cells (similar to Figure A.5) are usually cheap and offer some mechanical flexibility. The latter feature can be handy for applications requiring the cells to cover a curved surface. Nowadays, typical efficiencies for these cells go from 8% to more than 20% [73]. We will assume a very low efficiency value (typical for a decade ago) for the purposes of this work: a 6% solar to electric power conversion.

The effective area of a small *thin film photo-voltaic* cell is around $50\,\mathbf{mm} \times 50\,\mathbf{mm} = 0.0025\,\mathbf{m^2}$.



(a) GHI Average Annual Availability (**kWh/m²/year**).

(b) Relative Annual Variability of GHI (%/**year**).

Figure A.4: Average values of Global Horizontal Irradiation (GHI) and its variability in the Portuguese continental territory [72].

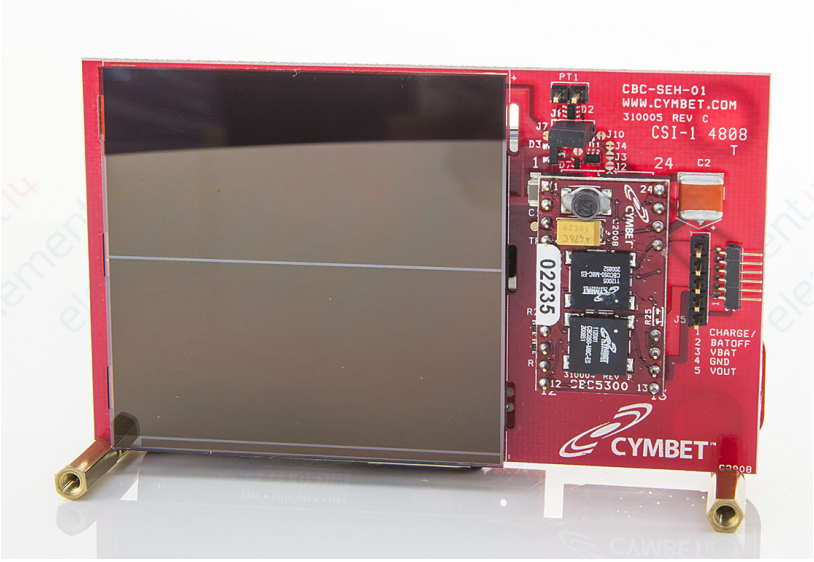


Figure A.5: Thin film *photo-voltaic* cell mounted on an EZ430-RF2500-SHE solar energy harvesting development kit [74].

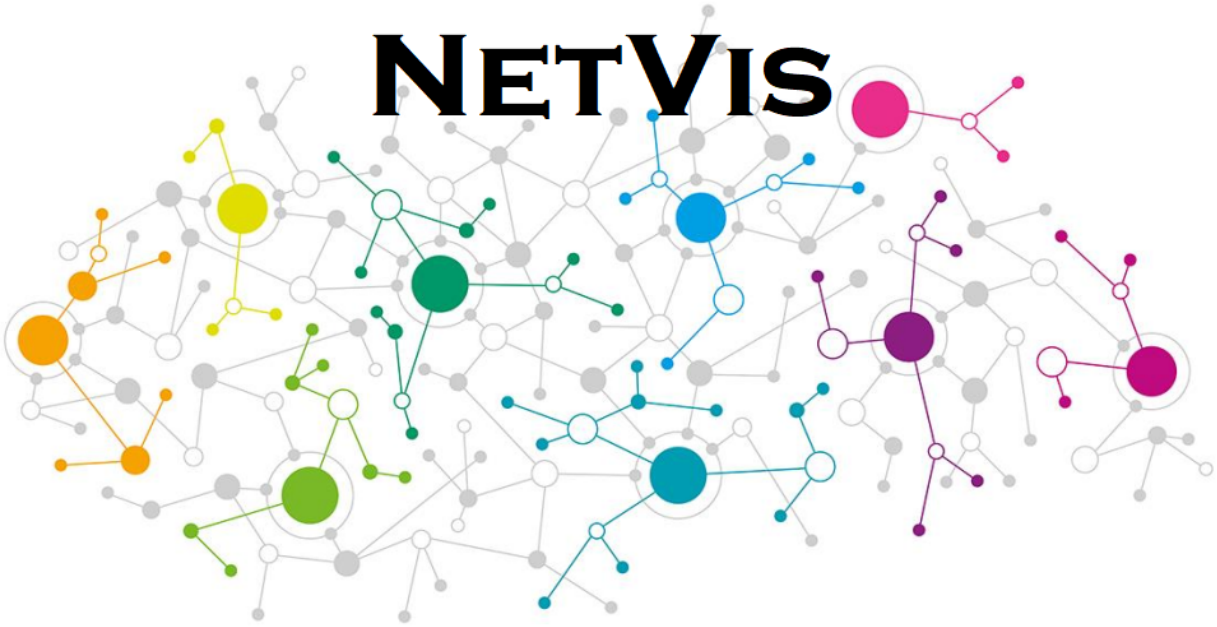
In short, the nodes can capture an average of

$$E_{electrical} = 183\,\mathbf{J/m^2/second} \times 0.0025\,\mathbf{m^2} \times 0.06 = 0.0274\,\mathbf{J/second}$$

During an entire *time slot*, a node can obtain around $0.0274 \times 300 = 8.23$ `Joules` of energy, which corresponds to 1.2684% of B_{max}. This value is an overall average, so the values used in Chapter 6 to test the model are below to this one, as this represents an optimistic approach and we can say it is a "high" value of harvesting.

Appendix B

Network Visualizer User's Guide



B.1 Introduction

Welcome! Thanks for using `NetVis - Network Visualizer Application`!

NetVis is a desktop app created to quickly create, manage and visualize simulations of an IoT networking optimization model developed at the Laboratory of Communications and Telematics of the University of Coimbra.

By using **NetVis**, you can manage the simulation model by creating new input data parameters, saving data models and loading existing ones, watching the results dynamically and extracting graphics from them.

B.2 Warning Note

This User's Guide refers to an alpha release of the **NetVis**. This means that the app may change in the future. Thus, the user is asked to understand that some functionalities can change and others can be added and removed, or both, until the final product release.

At the launch of the final product, an updated version of this User's Guide will also be provided.

B.3 System Requirements

NetVis requires a computer running an updated MATLAB Runtime Engine in any operative system. The system does not require to have the full MATLAB environment installed. The Runtime usually is provided within the installation files. If you do not have any, **NetVis** app will ask to download and install the required files from the internet.

NetVis also requires that the computer has installed the CPLEX engine to perform simulations. You can obtain this engine from here.[1]

Additionally, the computer must allow interaction via mouse or touchpad and keyboard.

B.4 NetVis Operation

B.4.1 First use

To get started with the application, make sure that you have all the referred *software* in the `System Requirements` section.

Subsequently and considering that you are in the directory where you identify the executable `NetVis.mlapp`, you must double-click that same file to run it.

After a few moments, the application window will appear, presenting the `Home Screen`. At the first use, it will display the default network topology: a square network composed of 25 nodes – Figure B.1.

From this point, you can use the software by clicking on the buttons and options of the `Home Screen` or explore the current network topology.

[1] https://www.ibm.com/products/ilog-cplex-optimization-studio

B.4.2 Data types

The **NetVis** use a data structure for storing all the data related to simulations called `NVDat`. This data structure is used in all the operations referred in this guide. It is possible to store it in a regular `.mat` file.

B.4.3 Home Screen

This is the opening screen of **NetVis** (Figure B.1). In here you can see:

- **Orange:** `Main Menu` – Access other app functionalities. This menu is accessible at all times.
- **Blue:** *Home Screen* Main Options
  - `Create New Graph`: Generates and displays in the `Network Representation` area a new network topology that encompasses the options in `Settings Screen`.
  - `Run Simulation`: Sends the current graph options to the CPLEX Engine and waits for the simulation results.
- **Red:** `Network Representation` area – Shows the present working network.

Navigating through the app is possible via the buttons on the `Main Menu`, each one allowing to access other screens in which you can perform further operations (see more in the next sections).

`Network Representation` area

Clicking and dragging with the cursor is possible to move around the represented network. The buttons in the top left can be used to *zoom in* and *zoom out* when necessary. Usually, all the network fits in the screen, but when that is not possible, the default view of the network will be a maximum zoom out to encompass all the network nodes. Use the `Home Button` to reset the view area to the default state.

The `Numbers Button` can show/hide the numeric identifiers of the network nodes.

In the default viewing state, the light blue dots represent the network nodes, the bigger dark blue dot is the *sink node*, and the lines between them are the possible wireless connections between nodes.

Whatever the options regarding graph topology selected in the `Settings Screen`, the app always defines by default the *sink node* of the network as the node with maximum

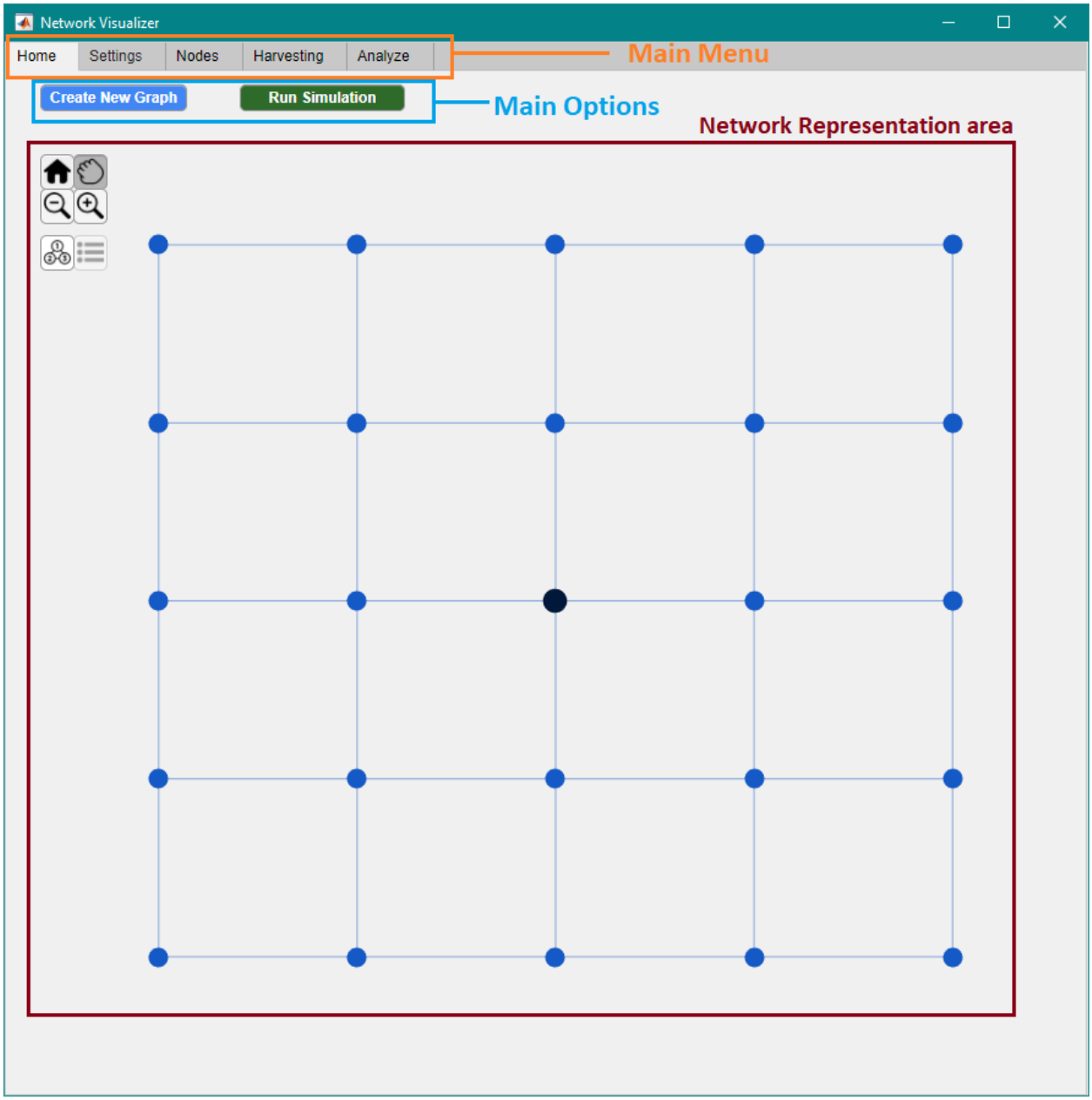


Figure B.1: **NetVis** *Main Screen* showing the default network topology

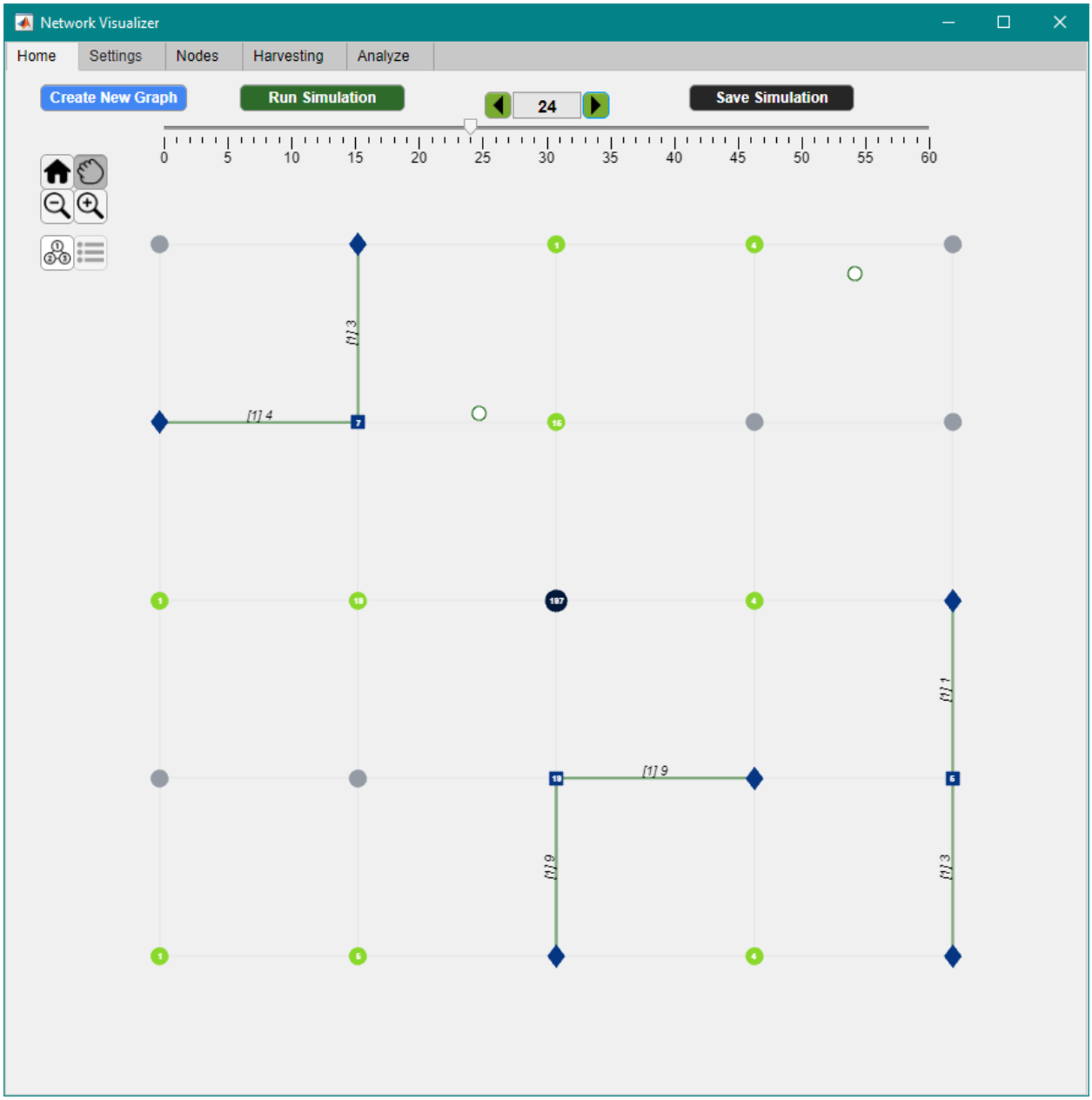


Figure B.2: **NetVis** *Main Screen* showing network state after receiving a simulation result

closeness centrality[2], i.e., the *sink node* stands in the closest position to all other nodes.

Exploring the network states through time after a successful simulation

After running a successful simulation (see `Perform a simulation`) or loading a saved simulation (see `Load and save simulation data`), the app `Network Representation` area will change to allow you to explore the status of the network over time. See Figure B.2. At the top of the `Network Representation` area, it will appear a *number box* showing the currently considered *time slot* and a horizontal slider representing all the possible *time slots* that you can explore. The explorable *time slots* go from 0 (initial network conditions) until the selected value of T for the presented simulation.

There are three possible ways to change the currently displayed *time slot*: click the *number box* with the mouse and write a new value, click on the green arrows buttons at its side to move back and forth one *time slot*, or click and drag the slider to the desired value. The `Network Representation` area will update accordingly.

The network information displayed in the `Network Representation` area has the following meaning:

- A big dark circle represents the *sink node*. In Figure B.2 is the central node.
- A grey circle represents a sensor node in sleeping state (`z`). In Figure B.2, the top right node is in sleeping state, for example.
- A light green circle represents a sensor node in sensing state (`s`). In Figure B.2, the bottom left node is in sensing state, for example.
- A blue diamond represents a sensor node in transmitting state (`Tx`). In Figure B.2, the bottom right node is in transmitting state, for example.
- A blue square represents a sensor node in receiving state (`Rx`). In Figure B.2, the node immediately below the *sink node* is in receiving state, for example.
- A light grey thin line represents a non-active link between two nodes.
- A thick green line represents an active link between two nodes. There is data flowing from the transmitting node (diamond) to the receiving node (square). The numbers above those links show `[Number of data packets] Number of data payloads` being transmitted through them.
- The white number inside the nodes represent the current *data payloads* in the nodes' *buffers*. If no number appears, then the respective node *buffer* is empty.

[2] `https://www.mathworks.com/help/matlab/ref/graph.centrality.html`

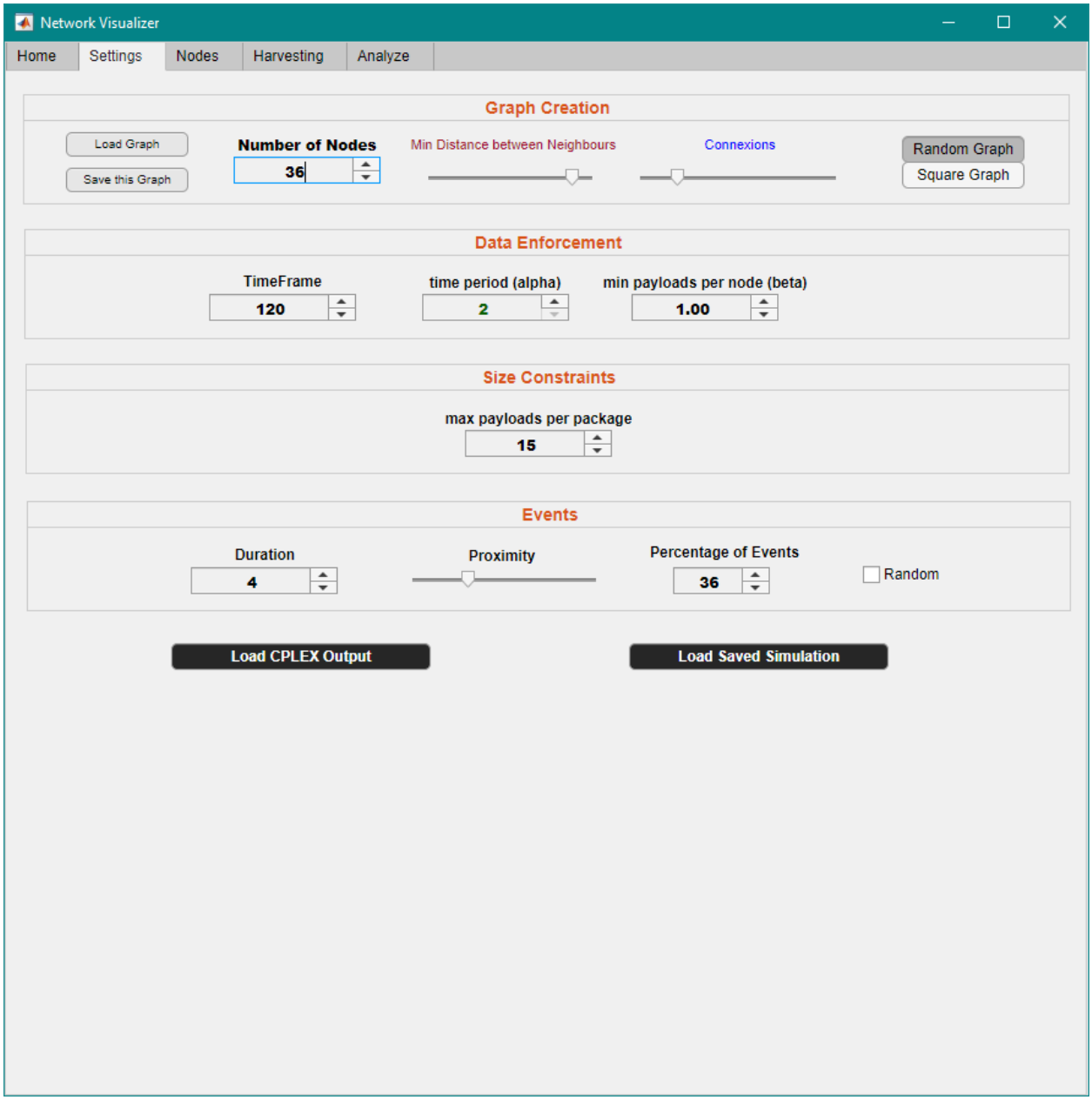


Figure B.3: **NetVis** *Settings Screen* showing the default parameters

- The dark green empty circles represent the location of the current events on the network. These are guaranteed to be captured by a nearby sensing node during their lifetime.

B.4.4 Settings Screen

All the major network settings input are in this screen (Figure B.3). Changing these allows to change some simulation parameters that are given as input to the CPLEX engine. From top to bottom, left to right:

`Load Graph` **button**

Loads a network topology previously saved by the `Save this Graph` Button. The app will switch automatically to the `Home Screen` displaying the loaded topology.

`Save this Graph` **button**

Saves to a file the data about the current network topology. Useful when is intended to repeat various simulations with the same network.

`Number of Nodes` **box**

Click the arrows or click and write to change the number of nodes of the network. This setting refers to the parameter N of the implemented model (Chapter 4). The topology only updates and takes into effect the new number after clicking the `Create New Graph` button in the `Home Screen` *Main Options*. When the *square graph* is selected, this box only allows perfect squares. Changing this parameter may also change some node-related parameters in the `Nodes Screen`.

`Min Distance Between Neighbors` **and** `Connexions` **sliders**

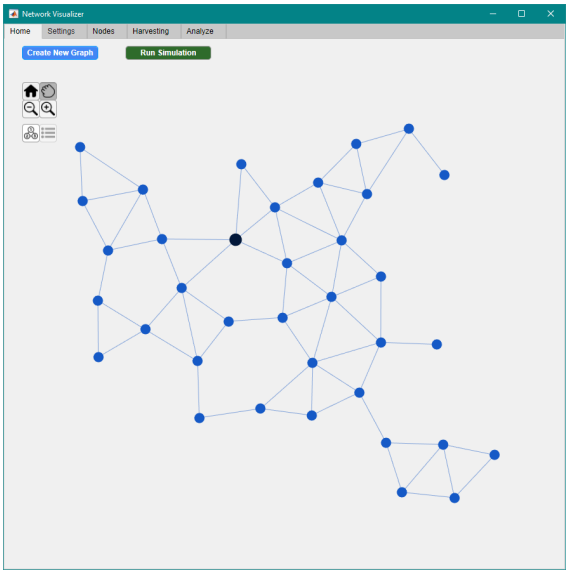
Refer to Figure B.4.

The `Min Distance Between Neighbors` slider is only visible when the *random graph* topology is selected. When creating a random topology, there is an absolute minimum distance around any existent node where no new node can be placed. This slider sets that minimum distance, therefore affects only the nodes spatial distribution in the graph. In any case, how far can they placed from other nodes is only limited by the overall connectivity of the whole network, that is, **NetVis** ensures that every node must be connected to any other node. When the slider advances to the left side, the forbidden radius is augmented, so the nodes tend to have more space between them and consequently tend to be more isolated and less connected. To the right side, the nodes have more probability to be closer to each other, therefore increasing the average number of connections between them.

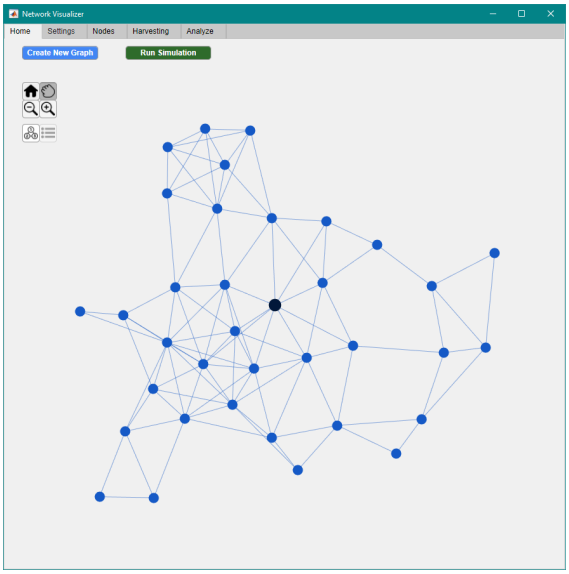
The `Connexions` slider changes the average number of links between the nodes in the network by varying the transmitter range of the nodes. To the left side this radius is diminished until the minimum possible number of links to maintain a viable network. To the right side it is augmented until it connects all the nodes between themselves. This option also works with *square grid* networks.

`Square/Random Graph` **selection**

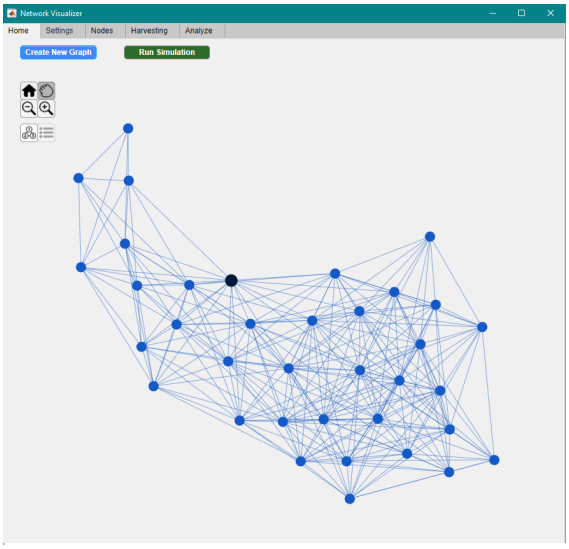
Change the network topology between a perfect square, as in Figure B.1 or a random topology – as in Figure B.4. The topology only updates and takes into effect after clicking the `Create New Graph` button in the `Home Screen` *Main Options*. With *random graph* selected, each press of the `Create New Graph` button creates a new non-repeatable topology.



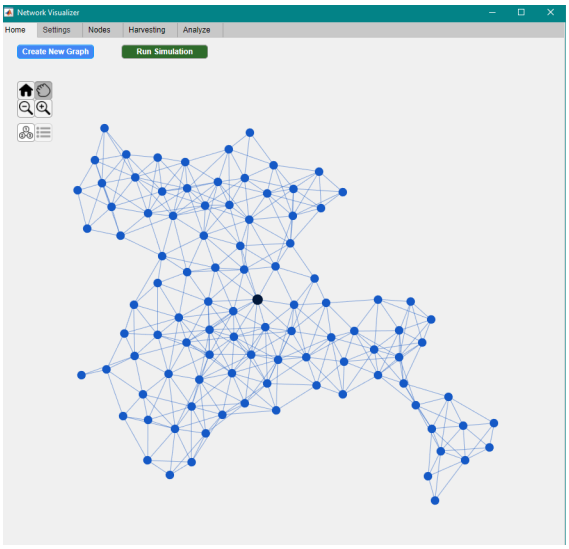
(a) 36 nodes, low connections degree, high distance between neighbors



(b) 36 nodes, low connections degree, low distance between neighbors



(c) 36 nodes, high connections degree, medium distance between neighbors



(d) 100 nodes, medium connections degree, medium distance between neighbors

Figure B.4: **NetVis**: some random network topologies with different parameters. The *sink node* is the dark blue more prominent node.

`TimeFrame` **box**

Click the arrows or click and write to change the *time-frame* of the simulation. This setting refers to the parameter T of the implemented model (Chapter 4). Changing this parameter may also need to manually update some time-related parameters in the `Settings Screen` and `Harvesting Screen`.

`Time period (alpha)` **box**

Click the arrows or click and write to change the time periods of the simulation. This setting refers to the parameter α of the implemented model (Chapter 4). The `TimeFrame` value must be a multiple of this value – the value appears in green if this is true and in red if not.

`min payloads per node (beta)` **box**

Click the arrows or click and write to change the minimum number of *data payloads* per sensor node that must be enforced in the simulation. This setting refers to the parameter β of the implemented model (Chapter 4).

`max payloads per package` **box**

Click the arrows or click and write to change the maximum number of *data payloads* per *data packet* that are used in the simulation. This setting refers to the parameter P_{max} of the implemented model (Chapter 4).

`Duration` **box**

Click the arrows or click and write to change the time duration (in *time slots*) of each of the events that exist in the simulation. This setting refers to the parameter δ of the implemented model (Chapter 4).

`Proximity` **slider**

To the left, the location of the events – randomly created as explained in Section 6.2.2 – will tend to be closer to the sensor nodes. To the right, the location will tend to be more distant. This parameter influences the average number of sensor nodes that can sense an event at the same time – more to the left, less to the right. This will affect the values of $E_i(t)$ of the implemented model (Chapter 4).

`Percentage of Events` **box**

Click the arrows or click and write to change the time percentage of the *time-frame* in which events will be created for the simulation. A number closer to zero will result in fewer events, and a number closer to 100 will result in more events. This parameter will affect the values of $v(t)$ of the implemented model (Chapter 4).

`Random` **checkbox**

Check this box to obtain always different events distributions, even if the other events parameters remain unchanged. A random distribution is useful to approximate the input data to reality.

Uncheck to obtain always the same events if the events parameters are the same. This setting is useful to retry simulations with the same conditions.

B.4.5 Nodes Screen

All the parameters related to the network nodes are on this screen (Figure B.5). Also, the nodes operational states and respective energy consumptions are found here.

Changing these allows changing some simulation parameters that are given to input to the CPLEX engine.

Nodes List

The `Nodes List` shows all the information about the network nodes. From left to right:

- **Node ID**: Each node unique identifier. Not editable. The app creates it automatically.
- **Is Sink?**: Boolean identifying which node is the *sink node*. It can be changed by clicking the respective checkbox. Only one node can be the *sink node*. The *sink node* always has infinite *buffer* size, and no battery related data.
- **Initial Payloads**: The amount of *data payloads* that a node has in its *buffer* at the beginning of simulation. Click each cell and write to change it. This setting can be altered for all the nodes at the same time by using the box above it. This parameter corresponds to the parameter $Q_i(0)$ of the implemented model (Chapter 4). Useful for retrying a simulation with the results data of a previous one.
- **Buffer Size**: The amount of *data payloads* that a node can hold at maximum. Click each cell and write to change it. This setting can be altered for all the nodes at the same time by using the box above it. This parameter corresponds to the parameter Q_{max} of the implemented model (Chapter 4).

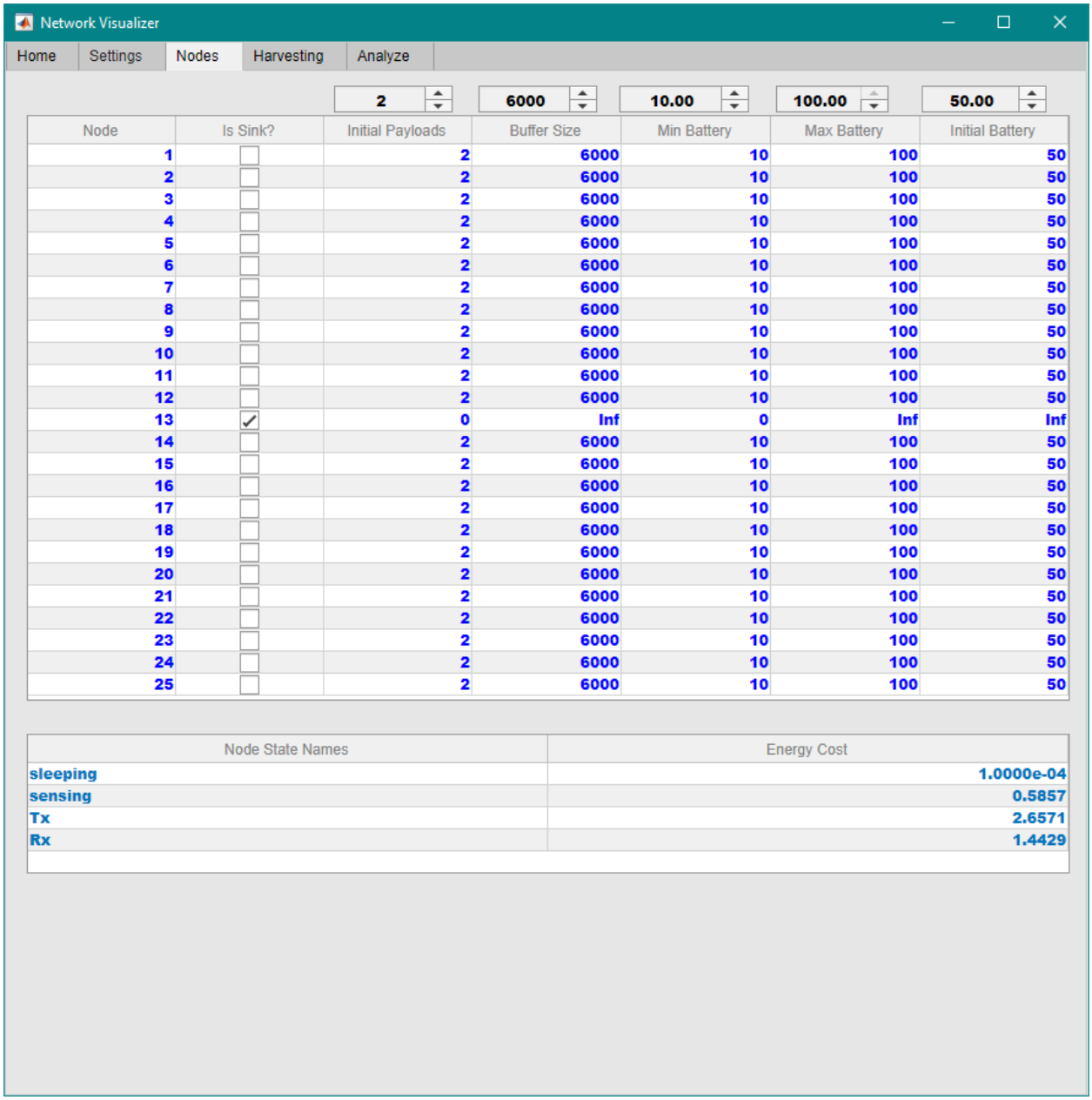


Figure B.5: **NetVis** *Nodes Screen*

- **Min Battery**: The minimum amount of battery that a node can have in the simulation. Click each cell and write to change it. This setting can be altered for all the nodes at the same time by using the box above it. This parameter corresponds to the parameter B_{min} of the implemented model (Chapter 4).
- **Max Battery**: The maximum amount of battery that a node can have in the simulation. Click each cell and write to change it. This setting can be altered for all the nodes at the same time by using the box above it. This parameter corresponds to the parameter B_{max} of the implemented model (Chapter 4).
- **Initial Battery**: The amount of battery that a node starts the simulation with. Click each box and write to change it. This setting can be altered for all the nodes at the same time by using the box above it. This parameter corresponds to the parameter $B_i(0)$ of the implemented model (Chapter 4).

Nodes States List

The Nodes States List shows the operational states that the nodes can have – on the left – and the corresponding energy consumption of the state per *time slot* – on the right. It appears below the `Nodes List`. Click and write to add a new state, to edit an existing one or to delete an existing one.

Warning: keep in mind that changing any of the nodes states in this list needs a manual rework of the CPLEX OPL model to be consistent with the new changes.

B.4.6 Harvesting Screen

In this screen (Figure B.6a) you can add or edit the amount of energy harvesting per *time slot* considered in the simulation. Note that the amount of harvesting is the same for all the network nodes.

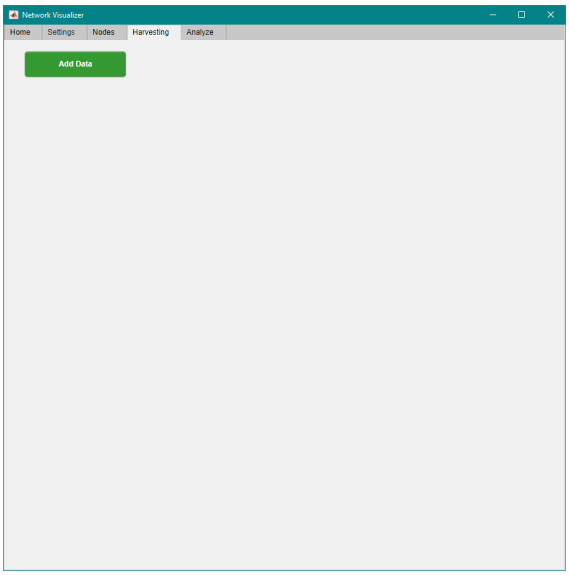
To start, click the green `Add Data` button. A small pop-up window appears (see Figure B.6b): you can choose between adding a constant harvesting value to all of the T *time slots* or to load existing data from a file (useful for retrying simulations with the same data, or to load external harvesting profiles).

If choosing the constant option, a box will appear next to the green button, where you can click and write the desired constant value. If choosing the data load option, a file picker window will appear to select the desired file from which to load the harvesting data.

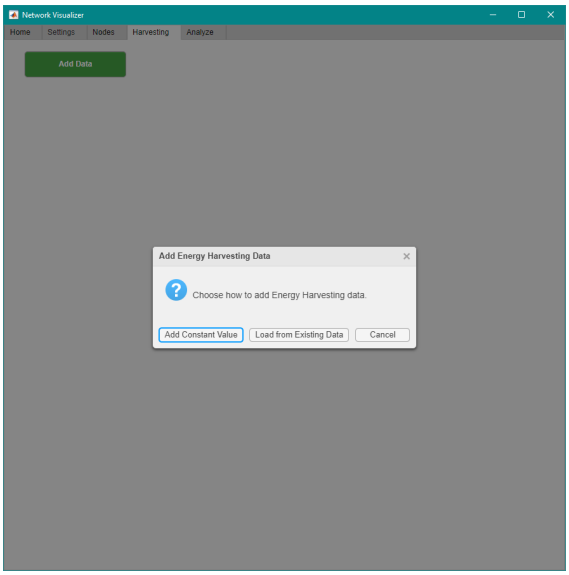
Whichever the selected option, a list will appear with the chosen harvesting values in each *time slot* (see Figure B.6c). It is possible to edit each one of them by clicking and

writing a new value. Dragging the horizontal scroll bar allows access to more *time slots*.

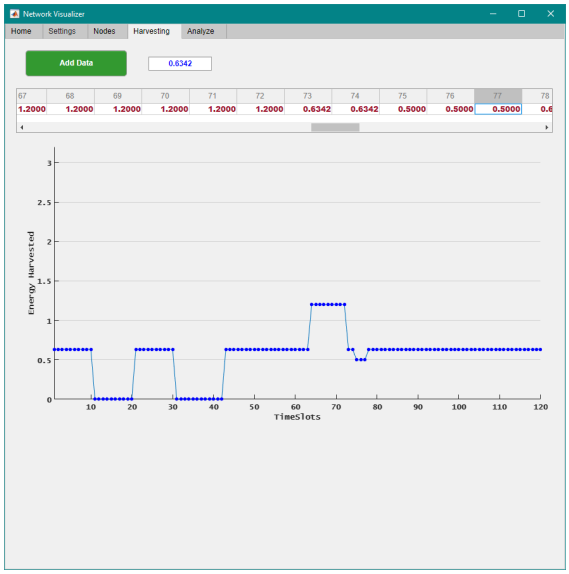
Below this list will appear the graphic representation of the selected values. This visualization is useful to see the chosen harvesting profile.



(a) Without Harvesting values added



(b) Pop-up selection window



(c) Harvesting values added: list and graphic

Figure B.6: **NetVis** *Harvesting Screen* sequence

B.4.7 Analyze Screen

In this screen (Figure B.7) you can obtain graphical representations about data generated by the simulations. The options in this screen only result in graphics after a successful simulation.

Choose an option in the center list with the mouse. A description of the resulting graphic

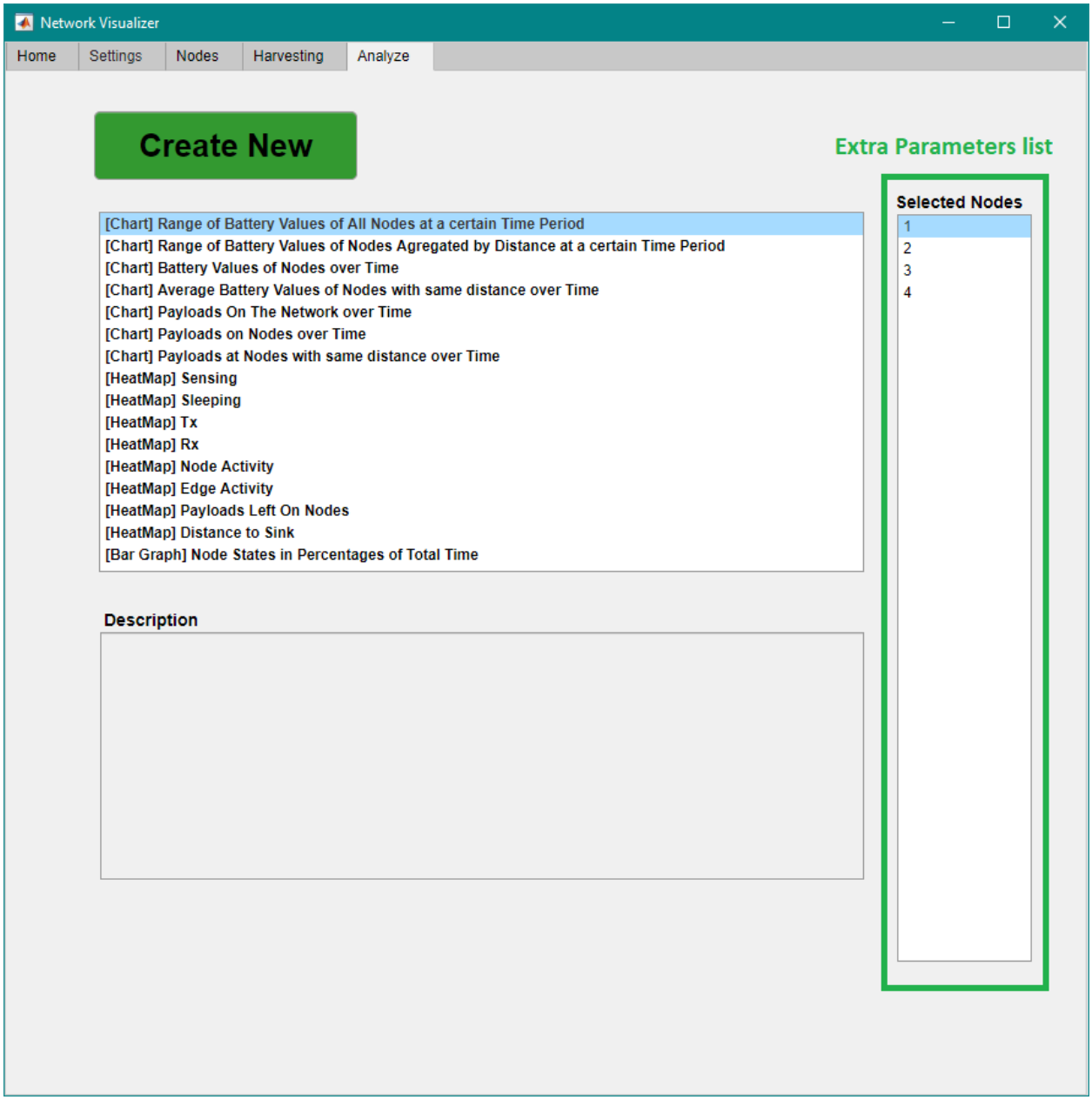


Figure B.7: **NetVis** *Analyze Screen*

appears in the big text-box below. Some options provide extra parameters on the list of the right. When satisfied with the selected options, click on the green `Create New` button to create a new graph that will appear in a new window. You can create as many figures as you want. The charts currently supported by the **NetVis** are:

- **Networks link heat map:** Shows which links are the most used to transmit *data packets*. Darker links are equivalent to more data. See Figure B.8a.
- **Nodes battery over time:** Shows the average battery of the nodes aggregated by the distance to the *sink node* over time. See Figure B.8b.
- **Node states distribution by distance:** Percentage of *time slots* that each node spends on a certain state. See Figure B.8c.
- **Node states allocation over time:** Shows in which operational state each node is at each *time slot*. See Figure B.8d.
- **Average *data packets* aggregation sent by node and *time slot*:** Shows in what *time slot* the nodes made transmissions and the amount of data aggregation in each one of them. See Figure B.8e.
- **Average *data packets* aggregation sent by distance and *time slot*:** Shows in which *time slot* the nodes made transmissions and the amount of data aggregation in each one of them. The nodes are averaged by the distance to the *sink node*. See Figure B.8f.
- **Range of battery values of all nodes at a specific time period:** The horizontal axis contains all the nodes ordered by distance to the *sink node* – left is close to it and right is far from it. The vertical axis has the battery values in percentage. The time period can be specified in the extra parameters list. See Figure B.8g.
- **Range of battery values of nodes aggregated by distance at a specific time period:** The same as above but it aggregates and averages the battery values from nodes with the same distance to the *sink node*. The horizontal axis represents the distance to the *sink node*. The vertical axis represents the range of battery values (in percentage) for the nodes at same distances. The time period can be specified in the extra parameters list. See Figure B.8h.

- **Battery values of a specific node over all the simulation:** The horizontal axis represents the time. The vertical axis represents the battery value in percentage. The selected node can be specified in the extra parameters list. See Figure B.8i.

- ***Data payloads* on the network over time:** It shows where all the *payloads* are during all the simulation. The horizontal axis represents the time. The vertical axis represents the number of *data payloads*. In the extra parameters list, it can be specified: *data payloads* at *sink node*; *data payloads* still outside *sink node* (at sensor nodes); *data payloads* created by sensing; and *data payloads* being transmitted. See Figure B.8j.

- ***Data payloads* on a node over time:** It shows the number of *data payloads* inside a specified node during all the simulation. The horizontal axis represents the time. The vertical axis represents the number of *data payloads* inside the nodes' *buffer*. The selected node can be specified in the extra parameters list. See Figure B.8k.

- ***Data payloads* on node with the same distance to the *sink node* over time:** It shows the number of *data payloads* inside the nodes of a specified distance to the *sink node* during all the simulation. The horizontal axis represents the time. The vertical axis represents the sum of *data payloads* inside the selected nodes' *buffer*. The selected distance can be specified in the extra parameters list. See Figure B.8l.

- **Sensing heat map:** It shows the cumulative sum of number of sensing states of the nodes until a specified *time slot*. The specified *time slot* can be selected in the extra parameters list. If the selected *time slot* is the last one then it shows the cumulative sum of all the simulation. See Figure B.8m.

- **Sleeping heat map:** It shows the cumulative sum of the number of sleeping states of the nodes until a specified *time slot*. The specified *time slot* can be selected in the extra parameters list. If the selected *time slot* is the last one then it shows the cumulative sum of all the simulation. See Figure B.8n.

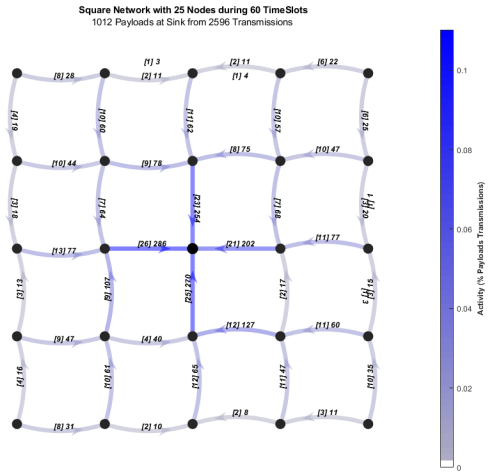
- **Transmitting heat map:** It shows the cumulative sum of the number of transmitting states of the nodes until a specified *time slot*. The specified *time slot* can be selected in the extra parameters list. If the selected *time slot* is the last one then it shows the cumulative sum of all the simulation. See Figure B.8o.

- **Receiving heat map:** It shows the cumulative sum of the number of receiving states of the nodes until a specified *time slot*. The specified *time slot* can be selected in the extra parameters list. If the selected *time slot* is the last one then it shows the cumulative sum of all the simulation. See Figure B.8p.

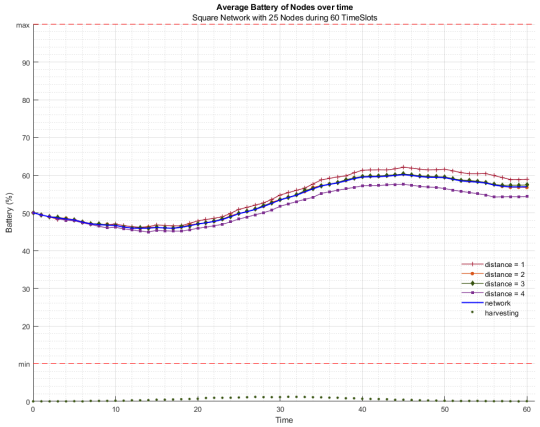
- **Node activity heat map:** It shows the cumulative sum of spent battery of the nodes until a specified *time slot*. The specified *time slot* can be selected in the extra parameters list. If the selected *time slot* is the last one then it shows the cumulative sum of all the simulation. See Figure B.8q.

- ***Data payloads* left inside sensor nodes heat map:** It shows the amount of *data payloads* that are inside each sensor's *buffer* at the end of the simulation. See Figure B.8r.

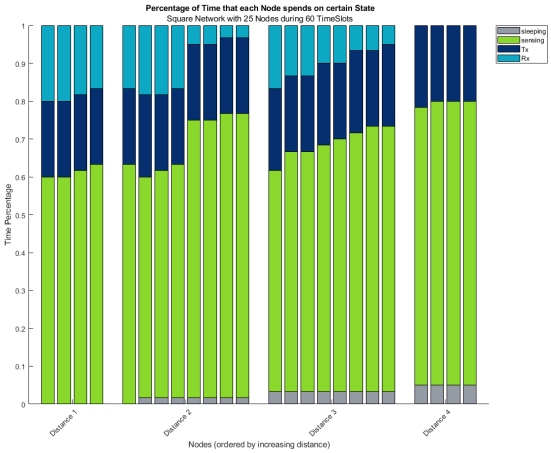
- **Distance to the *sink node* heat map:** It shows the shortest-hops-distance to the *sink node* of the sensor nodes of the network. See Figure B.8s.



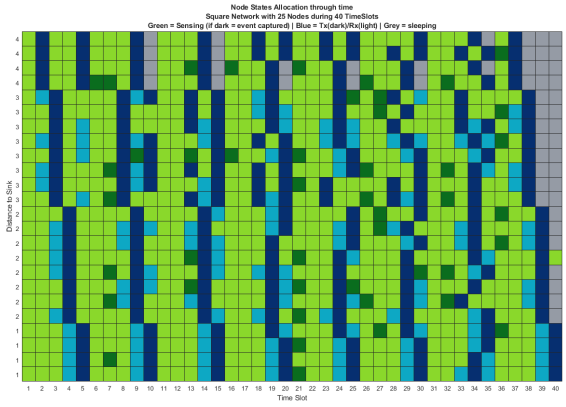
(a) Network links heat map



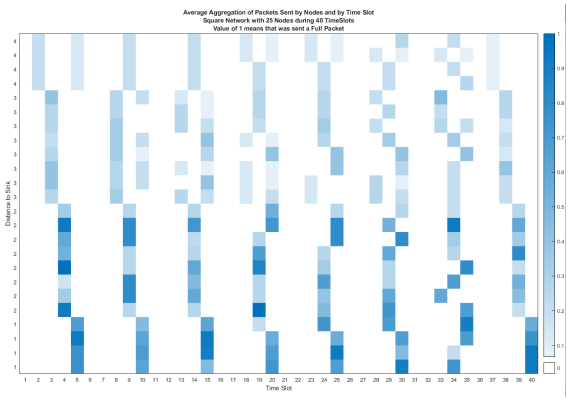
(b) Nodes battery over time



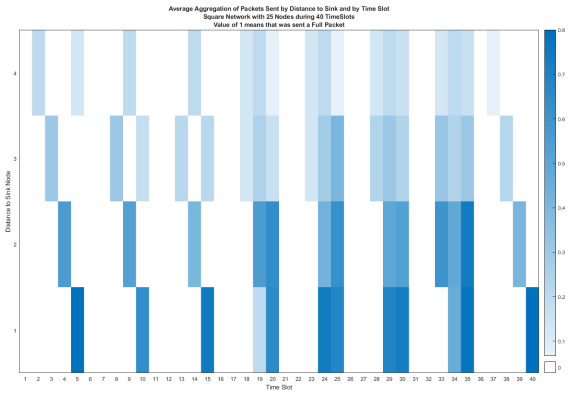
(c) Node states distribution by distance



(d) Node states allocation over time

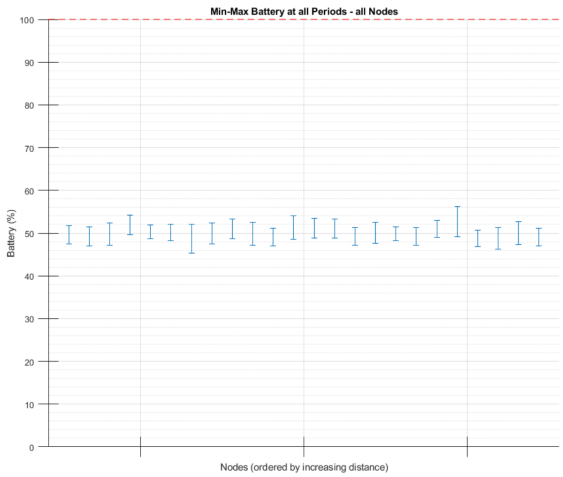


(e) Average *data packets* aggregation sent by node and *time slot*

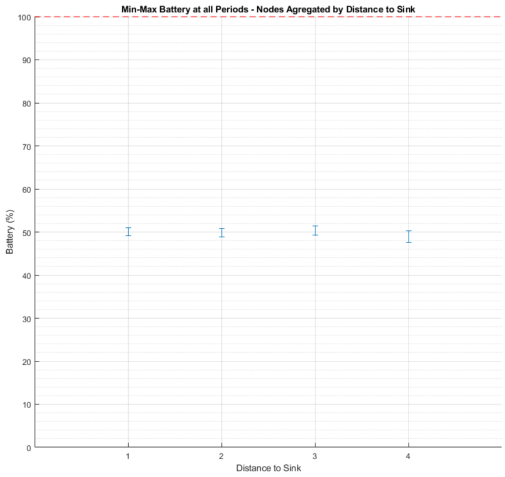


(f) Average *data packets* aggregation sent by distance and *time slot*

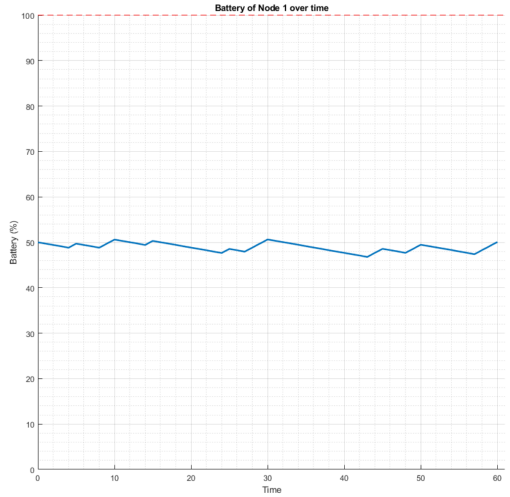
Figure B.8: Examples of types of graphics supported by **NetVis** *Analyze Screen*



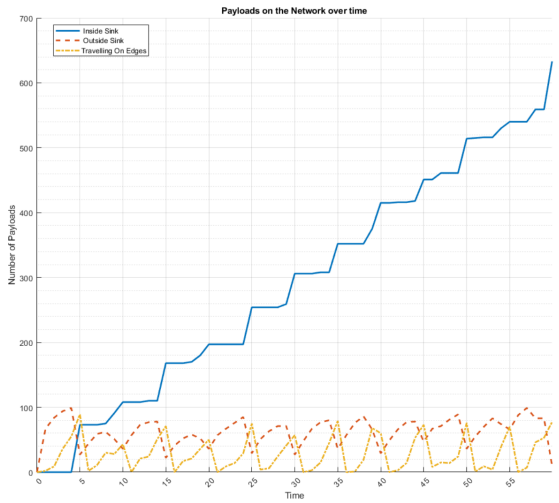
(g) Range of battery values of every node at a specific time period



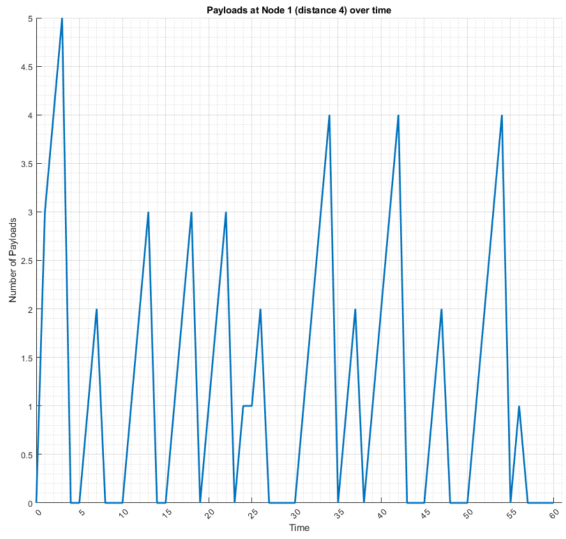
(h) Range of battery values of nodes aggregated by distance at a specific time period



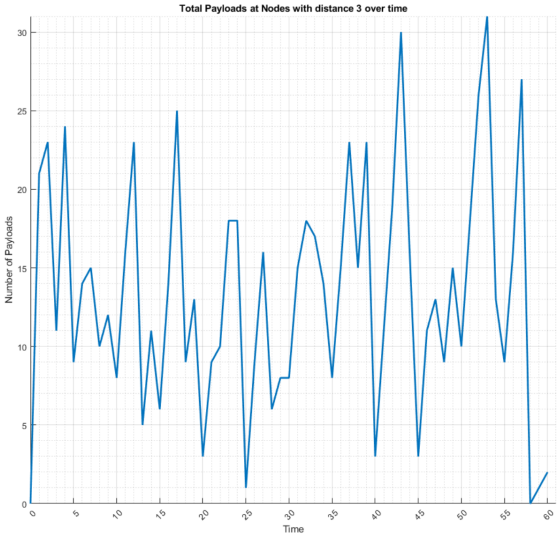
(i) Battery values of a node over time



(j) *Payloads* on the network over time

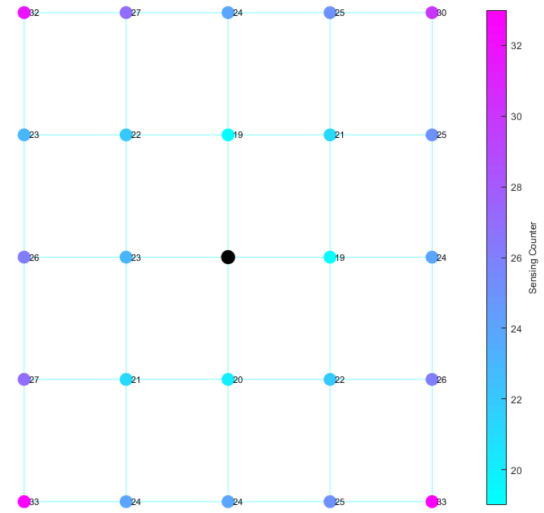


(k) *Payloads* on a node over time

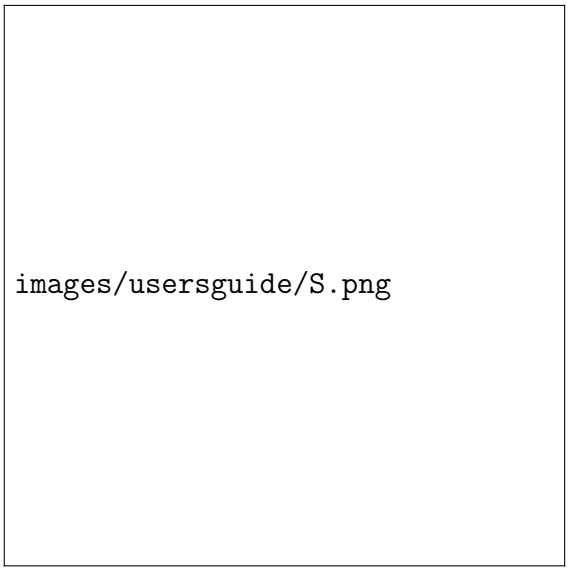


(l) *Payloads* on nodes with same distance over time

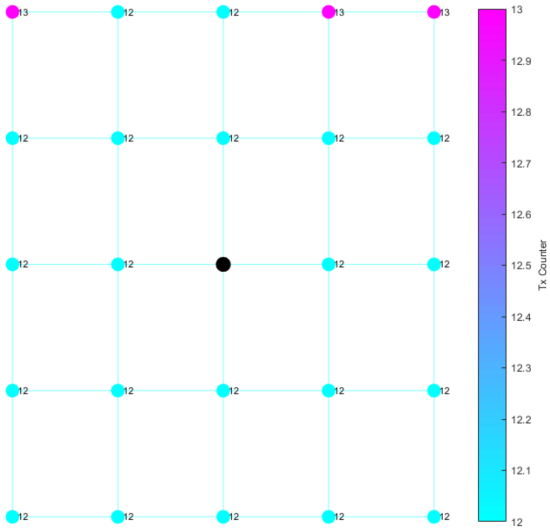
Figure B.8: Examples of types of graphics supported by **NetVis** *Analyze Screen* (cont.)



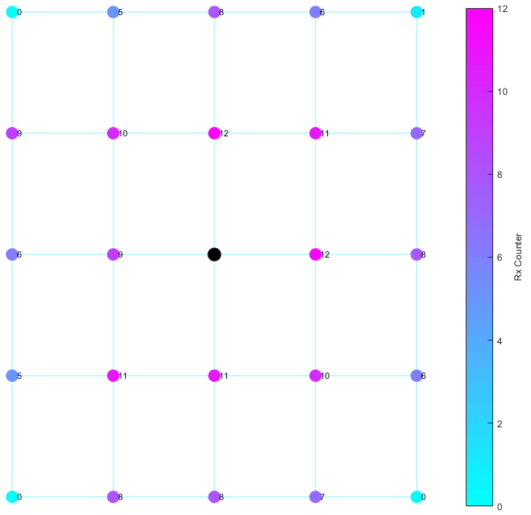
(m) Sensing nodes heat map



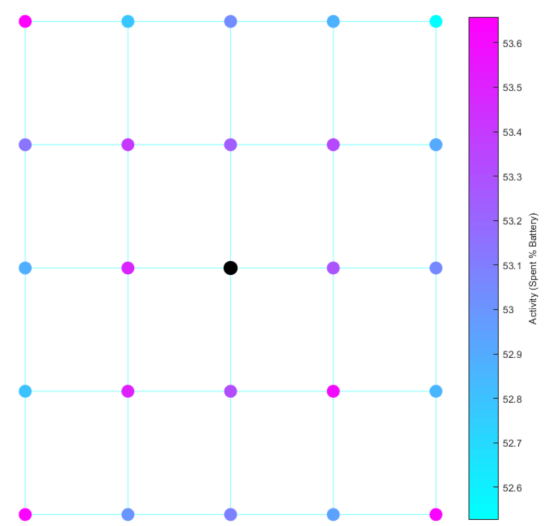
(n) Sleeping nodes heat map



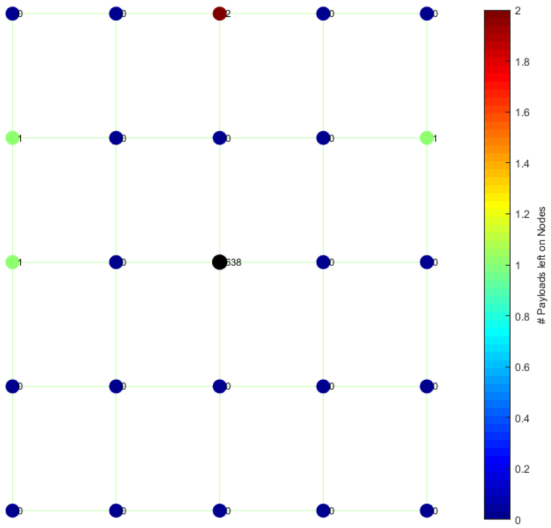
(o) Transmitting nodes heat map



(p) Receiving nodes heat map

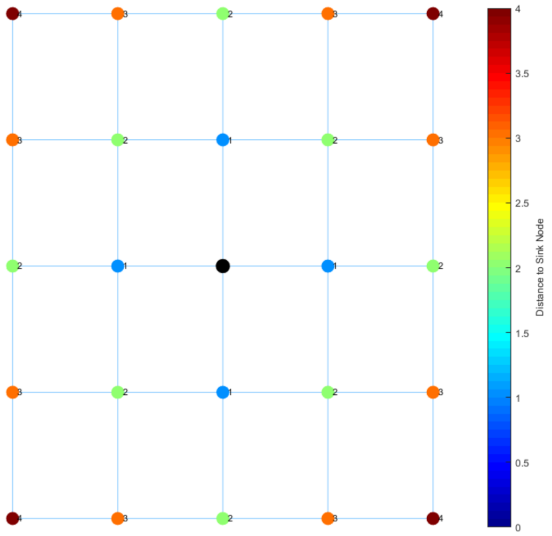


(q) Node activity heat map by spent battery



(r) *Payloads* left on nodes heat map

Figure B.8: Examples of types of graphics supported by **NetVis** *Analyze Screen* (cont.)



(s) Nodes distance to *sink node* heat map

Figure B.8: Examples of types of graphics supported by **NetVis** *Analyze Screen* (cont.)

B.5 Usage Examples

B.5.1 Perform a simulation

The most common use for the **NetVis** app is to obtain simulation results while varying input parameters to the optimization model of Chapter 4. For that purpose, the usual steps that you need to take to obtain results are:

1. Open the **NetVis** app – see `First use`. The `Home Screen` will appear.
2. Chose the desired network parameters by interacting with the boxes and buttons in the `Settings`, `Nodes` and `Harvesting` screens. See the respective sections to know how to change each parameter.
3. When satisfied with the input data, go back again to the `Home Screen` and click the green `Run Simulation` button.
4. If everything is correct, the data will be sent over to the CPLEX Engine and this engine will start the simulation.
5. Depending on the chosen input parameters, the results may take a very long time to be ready.
6. If CPLEX finds a solution for the simulation, the `Network Representation` area at the `Home Screen` will change colors. A slider and a textbox will appear at the top

of the `Home Screen` area. Interacting with those allows to view the network state *time slot* by *time slot*. See Figure B.2 and `Exploring the network states through time after a successful simulation`.

7. It is then possible to save the data in the `Settings Screen` or analyze it in the `Analyze Screen`.

8. To retry a simulation with a new set of parameters, go back to item 2.

Additionally, see Figure 5.1 for a visual depiction of the steps above.

B.5.2 Load and save simulation data

Save a simulation

After obtaining simulation data (check the steps in `Perform a simulation`), a dark `Save Simulation` button will appear in the `Home Screen`. Click it and a folder window will appear. Choose the folder location and the file name in which to save the simulation data.

Load a simulation

Go to the `Settings Screen` and click the dark `Load Simulation` button at the bottom of the window. A folder search window will appear. Find and select a previously saved simulation file. If the chosen file contains simulation data, the app will change automatically to the `Home Screen`. After this, you can explore the network states over time as seen in `Exploring the network states through time after a successful simulation` or analyze the simulation with the help of graphics as seen in the `Analyze Screen`.