Multicast communication in wireless sensor networks with the use of uncontrolled mobile relay node

**(Komunikacja rozgałęźna
w bezprzewodowych sieciach sensorowych
z wykorzystaniem niezależnego mobilnego węzła przekaźnikowego)**

Bartosz Musznicki

Faculty of Computing and Telecommunications
Poznan University of Technology

Doctoral dissertation

Supervisor:
Piotr Zwierzykowski, BEng, MSc, PhD, DSc

Poznań 2023

Abstract

The concept of wireless sensor networks develops and evolves over the last two decades. New research trends appear while more implementations or the elements of these are present in actual environments. Technological development leads both to the emergence of new network structure subcategories, as well as, to the intertwining of those which in are usually studied separately. This trend is visible in particular in urban environment. While researching and designing wireless networks, which are related to measurement functionalities for this environment, one can and should perceive sensor networks as heterogeneous structures composed of increasing numbers of devices of various connectivity, computation, and functional capabilities.

The networks of interest consist of stationary (fixed) and mobile nodes, which location may be known, predicted, or random, with their momentary structure being distributed and fragmented. Network functions performed by mobile devices are merely an addition to their main role with their movement not being controlled by the operator of the network. And yet, they can be used to enable the communication between disjoint areas of a delay tolerant network. Opportunistic node contacts can be used, e.g., in dissemination process of configuration and control messages aimed at a group of destination nodes. The messages are stored in relay nodes to forward them further at a later point in time, while mobile relays also carry them between stationary subareas of the network. The process may happen without the usage of network structure knowledge, with the use of the knowledge of local surroundings of the node, as well as, with global (complete) knowledge. Particular nodes can perform different network roles according to the function assigned to them.

The development of network functionalities is accompanied by the emergence of new sets and sources of data. More and more of them are publicly available in real time. It was leveraged to introduce a method of graph modeling of time-changing network connectivity structures. They are the basis of designed architecture and implemented research environment, which was used to investigate proposed multicast algorithms. Network modeling was performed in four Polish cities based on open data on the location of public transportation vehicles and elements of urban infrastructure. Over 60 thousand graphs were constructed and analyzed. It has been shown that the use of uncontrolled mobile relay enables the construction of space- and time-spanning multicast structures. Their features are determined by the topology of given city area, the distribution of destination nodes, as well as, the number and the routes of mobile relays. Efficacy and efficiency of the algorithms depend on radio range of the nodes, maximum time-span of forwarded messages, and network structure knowledge availability.

Streszczenie

Koncepcja bezprzewodowych sieci sensorowych rozwija się i ewoluuje na przestrzeni ostatnich dwóch dekad. Pojawiają się nowe nurty badawcze, a kolejne wdrożenia lub ich elementy są obecne w rzeczywistych środowiskach. Rozwój technologii prowadzi zarówno do wydzielania się nowych podkategorii struktur sieciowych, jak i do przenikania się tych, które zazwyczaj są badane oddzielnie. Ten trend jest szczególnie zauważalny w środowisku miejskim. Badając i projektując w nim sieci bezprzewodowe związane z funkcjami pomiarowymi można i należy postrzegać sieci sensorowe jako struktury heterogeniczne złożone z rosnącej liczby urządzeń o różnych możliwościach łącznościowych, obliczeniowych i funkcjonalnych.

Rozważane sieci składają się z węzłów stacjonarnych (nieruchomych) i mobilnych, których położenie może być znane, przewidywalne lub losowe, a chwilowa struktura rozproszona i fragmentaryczna. Funkcje sieciowe realizowane przez urządzenia mobilne są jedynie uzupełnieniem ich podstawowej roli, a ich przemieszczanie się niezależne od operatora sieci. Mimo to, mogą być używane w celu umożliwienia łączności między rozłącznymi obszarami sieci niewrażliwej na opóźnienia. Okazjonalne kontakty między węzłami mogą być wykorzystywane, m. in. w procesie rozsyłania wiadomości konfiguracyjnych i sterujących do grupy węzłów odbiorczych. Wiadomości są przechowywane w węzłach przekaźnikowych w celu późniejszego przekazania dalej, a mobilne przekaźniki dodatkowo przenoszą je pomiędzy stacjonarnymi podobszarami sieci. Proces może odbywać się bez wykorzystania wiedzy o strukturze sieci, przy wiedzy o lokalnym otoczeniu węzła, jak też przy wiedzy globalnej (całościowej). Poszczególne typy węzłów mogą pełnić odmienne role, zależnie od przydzielonej im funkcji.

Rozwojowi funkcjonalności sieciowych towarzyszy pojawianie się nowych zbiorów i źródeł danych. Coraz więcej z nich jest publicznie dostępnych w czasie rzeczywistym. Zostało to wykorzystane do opracowania metody grafowego modelowania zmiennych w czasie struktur odpowiadających łączności sieciowej. Są one podstawą zaprojektowanej architektury i zbudowanego środowiska badawczego, w którym analizowano zaproponowane algorytmy komunikacji rozgałęźnej. Modelowanie sieci sensorowych przeprowadzono w obszarach czterech polskich miastach na podstawie otwartych danych o położeniu pojazdów komunikacji publicznej i elementów infrastruktury miejskiej. Skonstruowano i przeanalizowano ponad 60 tysięcy grafów. Wykazano, że wykorzystanie niezależnego mobilnego przekaźnika umożliwia powstawanie rozciągniętych w przestrzeni i czasie struktur komunikacji rozgałęźnej. Ich cechy są warunkowane topologią danego obszaru miejskiego, rozmieszczeniem węzłów docelowych oraz liczbą i trasami mobilnych przekaźników. Skuteczność i efektywność algorytmów zależą od zasięgu radiowego węzłów, maksymalnej rozpiętości czasowej przekazywanych wiadomości i dostępności wiedzy o strukturze sieci.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The objective of this dissertation is to present new algorithms that enable modeling of multicast communication in urban wireless sensor networks in which uncontrolled mobile relay node is used. An important related research task is to introduce a consistent simulation-based investigation procedure which enables the evaluation of the algorithms in urban environment.

The dissertation is organized as follows:

- Chapter 2 presents wireless sensor networks to be currently found in urban environment.
  - Section 2.1 highlights the evolution of sensor networks and the development of related concepts.
  - Section 2.2 discusses the types and characteristics of actual sensor networks existing in urban environment.
  - Section 2.3 gathers key routing research problems in urban sensor networks. Opportunistic routing, data aggregation and offloading matters are followed by the presentation of the fundamental issue of topology control and modeling.
  - Section 2.4 discusses relevant observations.
- Chapter 3 analyses types of node deployment schemes that can be encountered in urban environment. They are illustrated with examples of real systems.
  - Section 3.1 investigates random node deployment.
  - Section 3.2 discusses deterministic distribution of nodes.
  - Section 3.3 presents related practical consequences.
- Chapter 4 introduces innovative method of modeling real-life urban sensor networks based on open data.
  - Section 4.1 investigates the characteristics of urban node location data sources.
  - Section 4.2 introduces data gathering, processing, and network modeling architecture.

- Section 4.3 presents novel network modeling algorithms with network modeling flow diagram and pseudocodes.
- Section 4.4 proves the feasibility of introduced network modeling methodology.
- Section 4.5 summarizes the implications of preliminary verification research.

- Chapter 5 introduces urban delay tolerant multicast framework using uncontrolled mobile relay.
  - Section 5.1 describes key design considerations.
  - Section 5.2 presents new delay tolerant multicast framework. It consists of a family of multicast message-oriented algorithms and procedures designed for opportunistic heterogeneous urban sensor networks with uncontrolled mobile relays.
  - Section 5.3 defines and discusses designed delay tolerant multicast router.
  - Section 5.4 outlines key observations.
- Chapter 6 presents the simulation study of introduced models and algorithms.
  - Section 6.1 covers research environment, simulation scope, architecture and analysis methodology.
  - Section 6.2 discusses space connectivity issues.
  - Section 6.3 analyzes space-time connectivity aspects.
  - Section 6.4 investigates the results of space-time multicast analysis.
  - Section 6.5 highlights simulation study observations.
- Chapter 7 concludes with the summary, lists the main contributions and suggests directions for further research.

In the course of the research, substantial fragments of Chapters 2–4 have been published in:

- B. Musznicki. Empirical Approach in Topology Control of Sensor Networks for Urban Environment. *Journal of Telecommunications and Information Technology*, (1):47–57, 2019 [1];
- B. Musznicki, M. Piechowiak, and P. Zwierzykowski. Modeling Real-Life Urban Sensor Networks Based on Open Data. *Sensors*, 22(23), 2022 [2].

In this dissertation, they have been revised and extended.

Chapter 2

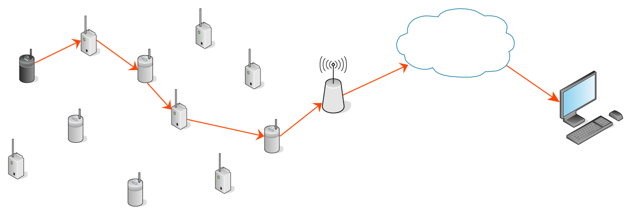
Sensor networks in urban environment

This chapter provides a juxtaposition between theoretical and research-based views on fundamentals of wireless sensor networks on the one hand, and empirical and implementation-based approach, as seen by a network architect with practical experience, on the other. The presentation is based on the author's experience with various network types and his involvement in tests pertaining to those networks. Hence, the evolution and actual types of urban sensor networks are described and characteristics of their nodes are given. Current urban environment routing research problems are identified as well.

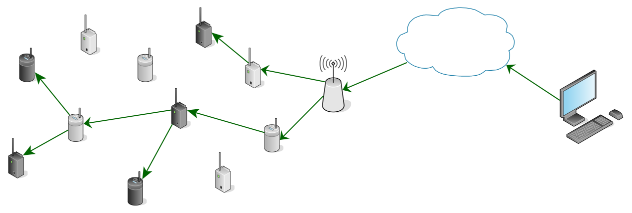
2.1 Evolution of wireless sensor networks

Typical *wireless sensor networks* (WSNs) consist of multiple nodes deployed over a certain area to perform a common sensing-related task [3]. The basic components of such networks are devices equipped with sensors that monitor the variability of physical phenomena and quantities, such as humidity, temperature, pressure, radiation, sound, motion, the degree of particulate air pollution, etc. Typical actual applications of sensors in urban networks include, but are not limited to, coordination of specialized vehicles (ambulances, emergency vehicles), public transport logistics (to optimize use of institutional resources), traffic management, monitoring of environmental parameters determining air and water quality, and monitoring of urban rental vehicles (electric cars, scooters, bicycles).

WSNs were and are focused on the optimization of wireless communication by implementing efficient algorithms and routing methods to save energy, among others. Such networks show the capacity to self-organize, resist single node damages, and apply radio transmission error correction and avoidance mechanisms. Every WSN node is equipped with measurement sensors, but also with its own power supply, wireless communication module, a microcontroller or microprocessor, memory, etc. [4], i.e., such a node is a specialized computer and router that continuously processes measurements and routes data. In addition, other components defining the applications area, e.g., a GPS signal receiver or relays that make the control of external actuators possible [5], may be required as well. Objects such as these may be combined to form an integrated system and are capable of cooperating to complete more complex and context-related tasks [6, 7]. They are often linked with additional analytical tools and distributed resources provided by cloud computing [8].



(a) Unicast



(b) Multicast

Figure 2.1 Multi-hop communication in wireless sensor networks

Therefore, designing compact and energy-efficient network nodes is a challenge. With reference to WSN, a *sensor* is typically understood not only as a component that performs measurements, but also as an entire small-size network node. Similarly, a WSN is often simply called a *sensor network*.

Initially, work on sensor networks involved the individual authors' own hardware designs, mainly due to the lack of commercial availability of dedicated products. As recently as 5–10 years ago, sensor platforms belonging to the MICA2, MICAz and TelosB families were considered to be the most advanced and were most widely accepted by researchers. Over the past few years, general-purpose embedded platforms, such as the Arduino UNO, equipped with the ZigBee module, or the Raspberry Pi 4 Model B which provides IEEE 802.11b/g/n/ac, Bluetooth 5.0 and Bluetooth Low Energy wireless connectivity, seem to be used increasingly frequently. Because of their easy availability, affordable pricing and numerous configuration options, they enable researchers, enthusiasts and innovators alike to design and deploy sensor networks or sensor-like networks [9–13].

In the canonical wireless sensor network model, the objective and focus are on monitoring environmental parameters and efficient transmission of data collected in the monitored area (space) using low-emission, low-power wireless communication technology. The information is then relayed, through intermediary nodes, to an endpoint (designated as a controller or monitor) that processes it locally, as well as conveyed further through a portal (gateway) to various systems and networks, e.g., the Internet. Nodes can be fixed (stationary) or mobile and have a defined role and purpose. They tend to be envisaged to operate within a defined framework and, at least to some extent, a homogeneous and well-defined topology.

A typical WSN is a multi-node multi-hop network. Therefore, the *unicast* presented in Figure 2.1a is usually the most frequent form of transmission. Single *source* node initiates such transmission addressed to a single *destinaton* node. The intermediate *relay* nodes send it further to make it reach the *sink* which is the gateway of the sensor network leading to external destination. A quite different relationship occurs in case of *multicast*, where a single node initiates the transmission to reach multiple destination nodes, i.e., a multicast group, as in Figure 2.1b. A special case of multicast, the *broadcast*, is the process of transporting the transmission from a single node to all other nodes in the network.

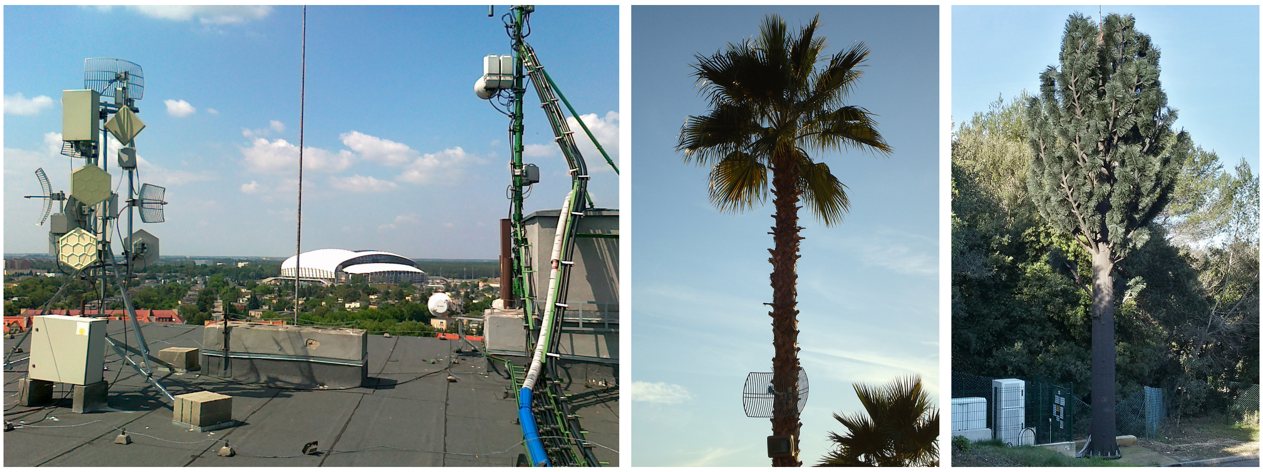
The actual structure of connections and the way they are used depend on the environment and the conditions in which the network is operating, as well as on the assumptions made and the tasks that are to be performed. It may be stated, based on the author's research, that the implementation phase is of a dual nature. It is sometimes preceded by a long-term scientific study (lasting for years), but more often simply follows the product development stage. This leads to a certain dissonance between research and actual implementation practices. On the one hand, the results of novel research related to optimization methods can be relied upon [14]. On the other, however, in response to changeable business needs, well-established technologies and solutions are continuously used, while the very vision of the sensor network is either being simplified or modified so that it meets the requirement of quick execution of ideas and commercial adaptation of the product and services to the needs of the market and society.

2.2 Actual sensor networks in urban environment

At present, one can observe an interesting evolution of sensor networks and the penetration of research areas in the study of these networks. The continuous development of radio technologies requires a holistic view of heterogeneous networks of interconnected sensors, such as urban networks, where sensors operating in different wired and wireless technologies generate huge amounts of data. Moreover, different types of control and maintenance messages are propagated. In such networks, measurements and data access are fundamental.

What becomes more and more apparent is the fact that objects which at first glance are a far cry from simple sensor nodes, here understood as devices with limited computational capabilities and battery-based power supply, are now being equipped with integrated sensor capabilities. In real-life applications, sensor functions are performed, ever more frequently, by vehicles and consumer devices, such as residential water meters, mobile phones and sports watches. They are often called smart objects and are considered to be capable of providing additional functionalities.

The factor that supports the development and implementation of urban WSNs is the increasing coverage ensured by different wireless access networks [15–18] that can be used to transfer data. Moreover, *low-power wide-area network* (LPWAN) technologies that support long-range low bit rate and energy efficient communication in sub-1 GHz frequencies are being developed (e.g., by LoRa Alliance, SigFox and Weightless SIG). In urban environments, such network infrastructure may be found, most frequently, on top of high-rise structures (see Figure 2.2), i.e., mounted on masts or placed on rooftops of buildings, as well as at more unusual locations such as, for instance, on the branches of an artificial tree on the slope of a hill or in palm trees. Furthermore, components providing access to different local (short-range) wireless networks are common inside office and apartment buildings as well.


(a) Rooftop cellular and IEEE 802.11 networks base station

Poznań, Poland
July 2011

(b) Wireless station on a palm tree

Athens, Greece
October 2013

(c) Artificial tree as a base station on the hillside

Sophia Antipolis, France
March 2016

Figure 2.2 Different types of urban wireless networks base stations

2.2.1 Types of urban sensor networks

The ever-increasing potential in terms of the range of applications for sensor-based or sensor-like devices, which are perceived as one of the components of the *internet of things* (IoT), is followed, within the domains of research and product marketing alike, by the need for their further differentiation. The concept of IoT extends the idea of WSNs and creates an ecosystem that gathers measurement data and transmits them over different types of networks, and includes a range of elements such as cloud-based aggregation services, big data analysis, and network management tools. At the same time, more subcategories with clearly defined functions and purposes are emerging [19, 20], such as, home automation, smart factory, or smart city. However, they all have one thing in common—they measure and rely on wireless communication [21]. Thus, the boundary between a typical sensor network understood as a distributed measurement and communication system using often homogeneous resources with common radio technology and other studied types of wireless networks is blurred, especially in 5G networks [18]. Therefore, in a more general manner, all such heterogeneous measurement nodes can be called *connected sensors* [22–24].

Wireless ad hoc networks (WANETs) are usually not bound to a strict framework or infrastructure. Therefore, they include numerous sub-types such as *wireless mesh networks* (WMNs), *mobile ad hoc networks* (MANETs), *vehicular ad hoc networks* (VANETs), and *vehicular sensor networks* (VSNs). The protocols used in MANETs are more complex due to mobility of nodes, and, thus, high network topology dynamics. At any time, network nodes can move spontaneously and in ways that are difficult to predict. Thus, the topology of a MANET can change rapidly and randomly at unpredictable times, which makes the design and implementation of routing methods a demanding task, hence the great popularity and a number of research works on MANETs and VANETs [25–27]. MANETs, as peer-to-peer multi-hop networks, assume the existence of mobile interconnected nodes which do not rely on any additional network infrastructure, such as base stations. As a result, there is no fixed infrastructure and the only limitation is the radio range of each network node.

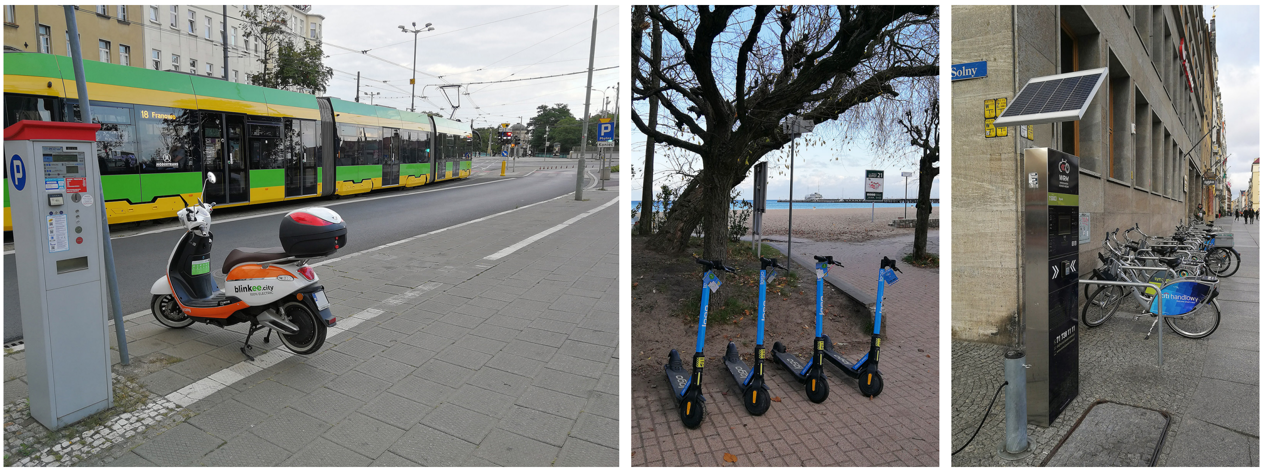
VANETs are similar to MANETs in the sense that they do not require any infrastructure for data transmission. However, VANETs place more emphasis on responding quickly to changes in network topology, directly related to varying road traffic structure and density. They also support higher transmission speeds as compared to MANETs. Their optimization problems take into account the traffic patterns of mobile devices (vehicles) and accurate positioning (radar, nearby devices sensing, satellite positioning, etc.). In this sense, they are better suited to urban environments. They can also play an important role in safe driving, smart navigation, rescue, and entertainment applications. Therefore, VANET-based applications are widely used in urban environments. Due to the large amounts of data generated, the concepts of bandwidth-limiting, discarding redundant data, and prioritizing users during high-traffic scenarios were introduced [28]. A broader term, *vehicle to everything* (V2X) is used in relation to the technologies that enable vehicles to communicate with their surroundings, e.g., other vehicles, street infrastructure, and pedestrians [29].

The picture of the structural complexity of different types of urban wireless networks that have sensing capabilities may be completed by investigating various specialized and targeted applications. An example, presented by [30], is a *wireless body area network* (WBAN) that connects independent nodes (e.g., sensors and actuators) that are placed in clothing, on the body, or under the skin of a person. The network typically extends over the entire human body, and the nodes are connected via a close-range wireless communication channel. Depending on the implementation, the nodes are arranged in a star or multiwire topology [31]. Such a *body area network* (BAN) offers many promising new applications in the medical field: remote health monitoring, healthcare, multimedia, sports, and many others, all of which enable free movement of a BAN user. Moreover, the applications can be related to everyday and leisure activities when based on data gathered by smart watches and smart phones. Such networks are yet another source of the vast amount of wireless data filling up shared urban wireless transmission medium.

2.2.2 Characteristics of urban sensor nodes

The possibilities for generating and processing data of various types by different types of nodes in heterogeneous urban environments are becoming essentially limitless. Most of those nodes perform, or could perform, some kind of sensing activity. Parking meters, trams and scooters in Poznań, electric kick scooters in Sopot, and bike rental stations in Wrocław are only a few examples of such connected devices already deployed in Polish cities, as presented in Figure 2.3. What can be pointed out, though, is the essential network and topological functions of a sensing node. In terms of routing, a single node can act as a source (originator), destination (sink), or relay (router). More complex relays are often termed (mobile) agents and are capable of performing advanced functions, such as data aggregation or buffering.

For instance, the paper [32] proposes a mechanism for node redundancy by introducing mobile agents that communicate opportunistically with a large field of sensors. The addition of mobile agents shifts computationally complex tasks from simple and power-constrained sensors to more advanced and efficient mobile agents. An increase in energy efficiency was achieved by adding an



(a) Parking meter, tram, and scooter
Poznań
August 2019

(b) Electric kick scooters
Sopot
November 2019

(c) Bike rental station
Wrocław
December 2019

Figure 2.3 Examples of urban connected devices in cities in Poland

agent and modifying radio transmission at the physical layer. Not only dedicated sensor nodes can be mobile agents—so can other wireless devices with different energy and data processing properties carried or installed on vehicles (e.g., smart phones, sensor nodes on private vehicles and public transportation, laptops, and even sensor-equipped animals).

The mobility model should mimic the changes occurring in the actual network. A number of mobility models for ad hoc networks have been defined in the literature, along with analytical approaches to single node and group mobility. The paper [33] shows that in order to incorporate network dynamics into a mobility model, for example, the gradient descent method can be used in the optimization process (instead of the popular Newton's law of motion known from classical dynamics). Synthetic networks with mobility models are used in the literature to test new routing algorithms and protocols for sensor networks [34].

The current state of research also takes into account the benefits of an architecture of wireless sensor networks, consisting of several mobile nodes of high performance and resources, accompanied by a large number of simple static nodes [35]. Mobile agents can act as mobile relays or mobile sinks. The performance of these two options and the trade-offs associated with them are being studied. Above all, the agent's mobility for network discovery and efficient data collection from static nodes is being planned. Clustering schemes of hierarchical WSN architectures that use mobile relay nodes to achieve energy savings and extend network lifetime are also analyzed. Relay nodes are clustered again when failures are detected [36].

Even more challenges arise when researching solutions for distributed opportunistic wireless networks that are designed to be disruption- and delay-tolerant or resistant. Such networks, commonly termed *delay-tolerant networks* (DTNs), with (a number of) nodes having not only complex processing capabilities, but also buffering (caching) functionalities, are composed both of fixed (infrastructure) relays that perform *store-and-forward* functions, and of mobile nodes (agents) that perform *store-carry-and-forward* functions. Such mobile agents are frequently called data *mobile ubiquitous LAN*

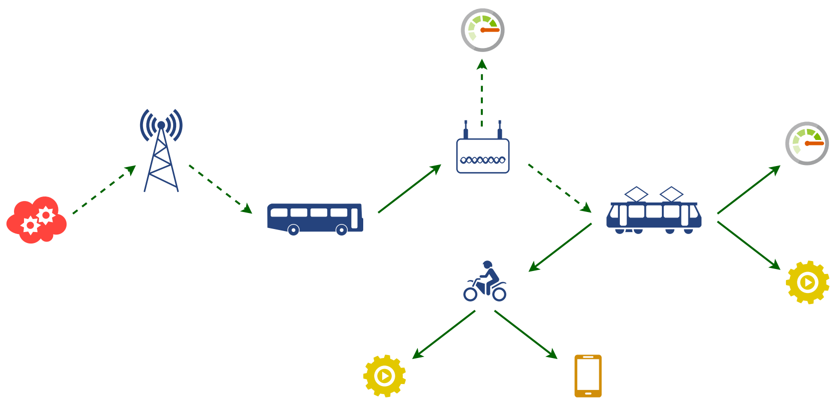


Figure 2.4 Urban delay-tolerant multicast sensor network

extensions (MULEs) [37]. In this way, network discontinuities may be bridged by delaying forward (relay) operation to a more suitable point in time, i.e., when new connectivity possibilities are provided due to the movement of the agent. In the case of city and traffic monitoring applications, this role can be played by vehicles (cars, buses, taxis, etc.) equipped with onboard network nodes and sensors. It is assumed that MULEs are capable of short-range wireless communication and can exchange data with a nearby sensing device or network gateway as they move past it. In this way, advanced mobile relays can receive data from sensors, buffer them, and send them to wired access points when they are in their vicinity. They can also operate in the opposite direction, i.e., disseminate data bundles, such as control messages from the source, to selected sensor nodes, as presented in Figure 2.4. The movement of some agents, such as drones, may be controlled, scheduled, or programmed to support their main functions, i.e., data routing. Other mobile relays, such as animals, humans, or vehicles, move autonomously, regardless of their routing role and perform their network functions opportunistically. Movement patterns of some agents that follow preplanned routes and schedules can be predicted though, at least to some extent (e.g., for public buses and garbage trucks).

2.3 Routing research problems in urban sensor networks

The complexity of data transmission issues in sensor networks is the source of many research problems. Those that relate to open (street) urban environment most are problems related to opportunistic routing, data aggregation, data offloading, and topology control and modeling, as discussed in the next subsections.

2.3.1 Opportunistic routing

Routing protocols designed for traditional multi-hop networks are designed for structures that do not change that continually, frequently, and rapidly as heterogeneous and highly mobile sensor networks. Therefore, the process of designing efficient routing methods for WSNs is a particularly difficult task. Unicast, multicast, and broadcast algorithms and protocols need to be tailored to large-scale and frequently changing networks of a varying node number and connectivity. Moreover, when designing methods for sensor networks, the requirement for energy conservation, especially when autonomously powered, as well as proper management of node resources, must be taken into account. As a result, the choice of the best data transmission path is more complicated than just a simple selection of one of neighboring successive nodes. Where a path is not always available, which cannot be known in advance, the network is expected to be partitioned and change frequently, with limited routing information available, while opportunistic routing may be implemented. Multicast approach may be used, for example, to distribute software updates and control or emergency messages to selected class of devices dispersed throughout the urban area.

Opportunistic networks are an interesting step in the evolution of wireless networks. In opportunistic networks, source and destination might be able to communicate even if no time-continuous (uninterrupted) multi-hop path exists between them. Opportunistic routing benefits from the changes in the structure, usually related to node movement, and aims at bridging connectivity gaps by radio coverage extension or by message buffering.

One of the techniques currently used is to take advantage of the broadcasting nature of the wireless transmission medium. It can be assumed that omnidirectional radio transmission of one node can be overheard by multiple neighboring nodes simultaneously. Unlike popular routing mechanisms that select the next node before sending data, based on fixed parameters and network structure, opportunistic routing selects the next node or nodes dynamically once communication opportunities arise. In this way, forwarding of data may be then performed by the neighbor closest to the destination. Additionally, multipath and multiple-copy (*n-copy*) or epidemic routing may be deployed to increase the probability of successful delivery. Opportunistic networks can even operate without, or based on scarce, routing information, implementing simple flooding-like methods [38] or more sophisticated ones, such as *beacon-less routing* [39], where no node presence messages (beacons) are periodically broadcast to make neighboring nodes aware of the existence (and often location) of other nodes. Opportunistic routing has been shown to achieve better performance than traditional routing under a variety of demanding conditions. One of the key tasks outlined in [40, 41] may be the selection of forwarding nodes and the prioritization of nodes in this set.

The concept of opportunistic networks originated from the research on delay-tolerant networks, which led to the development of a DTN architecture [42]. It typically consists of independent network partitions where there are only occasional opportunities for communication between them, sometimes known and scheduled in time, and sometimes completely random. The disconnected and dispersed networks form DTN regions, and the agent and gateway system are responsible for enabling connections between them. This model fits the characteristics of those urban sensing solutions which do not require fixed cell-like wireless network infrastructure or complete area-wide coverage. They may also be of use in emergency situations and conditions, such as natural disasters, grid power outage, or in a war zone. In such scenarios, single devices and separate islands of fixed (e.g., air quality meters) or mobile (e.g., humans with smartphones) nodes equipped with different sensors and radio interfaces are able to receive messages and transmit collected telemetry data and their location when in contact with a mobile agent or passing by a fixed gateway equipped with long-range network connection (e.g., a bus stop) [41]. Content distribution may also be performed in a similar way [43, 44]. Opportunistic routing can also support emergency services in everyday operations.

The work of [45] analyzed *global positioning system* (GPS) traces generated by fire service vehicles. The results reveal the characteristics of such networks formed by devices following this type of mobility with different radio communication ranges. Formed heterogeneous networks are scattered and fragmented, but there are delay-resistant routes connecting the areas. These results can be used in the design and implementation of solutions from the physical layer to the application layer.

The construction of a sample opportunistic multicast graph is discussed in Section 4.4.2.

2.3.2 Data aggregation

Smart city solutions aim at increasing sensing coverage, diversity, and quality of data obtained from various sensors to provide better services. The data are usually transmitted in a star-like structure, at least locally, to designated fixed nodes, i.e., base stations or gateways. The resources of such nodes can far exceed, in terms of computing power, available memory and storage, power supply, and connectivity, those of simpler devices. Therefore, they can act as transmission aggregators, local buffers, and relays to higher layers of the network using dedicated connections (i.e., other transmission media and technologies or radio links of longer transmission range). However, covering entire metropolitan area with static sensors and providing continuous network access can be unfeasible or expensive. As a result, a few studies consider using a public transportation system as a mobile platform for sensor nodes with networked bus stops acting as data sinks [46]. Structure of this type can improve sensor network coverage and take advantage of opportunistic communication. This solution might be used in latency-constrained applications, and hence an algorithm was proposed to select which bus stops should act as sinks to minimize the maximum delay of message delivery. Experiments show that when using only 16% of bus stops as receivers, a nearly 10% increase in the maximum network delay can be achieved without significantly losing the spatial coverage.

Another example is the concept of the *internet of bikes* (IoB), i.e., a sensor network based on an urban bicycle system [47]. An IoB-DTN routing protocol based on data aggregation is proposed, which applies the DTN paradigm to *internet of things* applications. Data read by bicycles are transmitted in store-and-forward mode and aggregated by a set of dedicated receivers. The IoB-DTN protocol can be viewed as a simplified version of various n-copy DTN protocols, optimized for IoT devices (including some routing functions that are of no use in DTNs being removed).

Exemplary data aggregation related network is modeled in Section 4.4.1.

2.3.3 Data offloading

Yet another research aspect is the problem of the huge amount of data generated by sensors in WSNs, especially in urban environments. Traffic generated by *machine-type communication* (MTC) between devices reached 49 exabytes between 2016 and 2020 [48]. WSNs are typically said to include a large number of devices deployed randomly in a highly dynamic environment. The types of sensing capabilities, data gathering, and communications range of the nodes are typically fixed. When no DTN or opportunistic routing approaches are involved, high device density is used to maintain preferred level of network coverage to ensure the reliability of data collection. Oftentimes, the devices are event-based systems that may attempt to report occurring events or perform measurements and transmit data at the same time. Therefore, attention was placed on optimizing the amount of data transfer in such wireless networks. In [49], the authors proposed a low-latency, low-power *medium access control* (MAC) protocol for hierarchical wireless networks. The protocol involves the transmission of data from end nodes to the sink node via a cluster head. However, this approach applies to large, hierarchical but homogeneous networks, while, as outlined earlier, in real-life urban networks, the boundaries between node types and transmission technologies are blurred.

Bonola et al. [50] conducted data dissemination and gathering scenarios analysis based on position traces of about 320 taxi cabs in the center of Rome, Italy, in a six-month period in 2013–2014. The area of 8 km by 8 km was characterized by congested narrow roads of high traffic and low speeds. It was divided into a 200 by 200 cell grid, where each cell covered the area of 40 m by 40 m. A *store-carry-and-forward* approach was investigated, in which taxis incidentally passing stationary nodes, such as trash bins and street lights, exchanged data with them and performed the roles of data MULEs. The results of the study indicated that with 120 vehicles on average, 80% coverage can be achieved in less than 24 h. A one-month portion of gathered taxi mobility traces is publicly available at *CRAWDAD* repository [51].

Similarly, Dias et al. [52] investigated the feasibility of a delay-tolerant vehicle network in the city of Rio de Janeiro, Brazil, using public transportation system data. The performance of such a network was evaluated by analyzing a large dataset of high mobility data—12,456 buses and 5833 taxi cabs recorded over a 24 h period based on their GPS positions. The presented results indicate the validity and feasibility of the use of the public transportation system as a delay-tolerant data network that provides significant coverage of the city. In the study, a clustering algorithm was used to group nearby vehicles into cells. Then, those clusters were modeled as nodes of a weighted directed graph with edges representing vehicle travels between clusters, and a number of metrics were analyzed. It is worth mentioning that the data were collected by the authors in October 2014 from the source available through open data portal *Data.Rio* [53]. The data were then shared by the authors and can be downloaded from *CRAWDAD* repository [54]. They contain recorded date and time, identifier, line, latitude, longitude, and speed of each bus.

A method for modeling a data offloading related graph is presented in Section 4.4.2.

2.3.4 Topology control and modeling

Network topology is typically understood to be a model that describes the structure of connections between the elements within a given network [55]. This notation may refer to both physical relations (relative arrangement of nodes and connections between them, directly stemming from the properties of the transmission medium applied) and logical relations (an operational configuration based method for transmitting data via the network from the starting point to the end point, between the elements of the network's infrastructure) [56]. As a result, physical and logical topologies may be distinguished [57]. *Topology control* is a related notion and in general encompasses different aspects related to planning, maintenance and adaptation of the system of connections within a given network [58]. *Topology management* is an alternative term used on some occasions [59, 60].

In its broadest and the most diverse scope, topology control is closely connected with wireless networks, not only due to the variable character of radio communications, but also due to the particular features of devices that rely on wireless transmission. Topology control is distinctively illustrated with regard to the ad hoc networks [61–63], and becomes of special significance within the WSN context. As the development of sensor networks progresses, this issue is gaining in importance and reflects the increasingly more detail-oriented and extensive scrutiny of each of the aspects influencing the network structure.

Santi states that "*topology control is the art of coordinating nodes' decisions regarding their transmitting ranges, in order to generate a network with the desired properties, e.g., connectivity, while reducing node energy consumption and/or increasing network capacity*"[58]. Labrador and Wightman point out, more broadly, that "*topology control is the reorganization and management of node parameters and modes of operation from time to time to modify the topology of the network with the goal of extending its lifetime while preserving important characteristics, such as network and sensing connectivity and coverage*" [64]. At the same time, they emphasize that the above definition refers not only to the control of the transmitting power of sensors, but also to turning on and shutting off nodes depending on current needs. Aziz et al. provide the following definition "*topology control is a technique that uses any controlled network parameter to generate and maintain a topology for the benefits of reducing energy consumption and achieving a desired property for an entire network*" [65]. Li et al. present, in turn, a view that topology control is a fundamental benchmark "*which characterizes how well a sensing field is monitored and how well each pair of sensors is mutually connected in WSNs*" [66]. Consequently, studies related to topology control may include investigations into operational management and transmitting power control of radio modules [67, 68], energy-harvesting [69], interference prediction [70], as well as sensor placement [71], network coverage [72], logical network structure and message routing [73, 74], node functional diversification and hierarchy (e.g., chaining [75], and clustering [76]).

Having closely examined the development of this particular domain and taking advantage of the definitions presented above, it can be generalized that the term topology control covers all activities intended to influence the physical or logical structure of a network in order to optimize the way the network executes its tasks while retaining the expected properties.

The related terms *graph* and *network* are often used interchangeably throughout this dissertation, but it is necessary to point out that a graph here is a mathematical structure used to model the topology of a communications network. A graph is defined as a pair $G = (V, E)$, where V is the set of vertices (nodess) and E is the set of edges (links) between vertices [77]. Corresponding elements of a modeled network are devices and wireless connections between those devices. Each node and edge may be labeled with more than one parameter (attribute), such as identifier, occurrence time, or cost.

Research on topology control of WSNs is focused on modeling and analyzing methods that can potentially be used to optimize the interconnection structure (e.g., to reduce power consumption). However, in practice, the ideas and concepts proposed by the research community are rarely used by network designers, and sensor systems that have already been implemented or are under development in urban environments rarely take advantage of the available research models developed and presented in the literature. Moreover, the widespread access to a variety of wireless technologies, which enable the empirical development and popularization of new solutions, especially in urban environments, as discussed in Chapter 3, also encourages the consolidation of this particular trend. Analyzing the topology of such networks, designing new methods, and diagnosing problems should therefore be based on data obtained from real heterogeneous networks.

The above observation partly coincides with the opinion, as presented in [78], that "*although new topology control algorithms are presented on a regular basis, topology control has never made the breakthrough in real-world deployments*", and may be accompanied by a statement that the obstacles faced include the following: unrealistic assumptions, unsuitable graph structures, application agnosticism, unclear role in the stack and insufficient framework support. The arguments presented included insufficient utilization of graph-based methods for optimization of the structures of typical WSNs.

Sensor networks are often modeled using graphs in a similar way to modeling wired and wireless telecommunications and data communications networks. Most relevant WSN studies conducted hitherto have been mainly based on synthetic models and involved simulations [3] or the results obtained in experimental testbeds [79]. A number of other studies used historical data obtained from transportation operators. Fortunately, new data sources have started to arise in the last few years. Many of those are open (publicly available) and provide real-time open data on the location of public transportation vehicles and different elements of urban infrastructure as well as the readings of the measurements they perform. Usually, those *application programming interfaces* (APIs) are well documented and available to use free of charge, as discussed in Section 4.1. The introduced open-data graph-based network modeling architecture and algorithms are defined in Section 4.2 and Section 4.3 with network modeling proof of concept presented in Section 4.4.

2.4 Observations

Sensor networks continue to raise the interest of researchers who pursue further improvements to complex problems and propose potential innovative applications. Network technologies and hardware platforms are being developed in parallel, while more and more types of devices are equipped with sensor components. By striving to optimize the operation of WSNs, it is possible to make use of or draw inspiration from new ingenious concepts and products.

In spite of sensor network products and services available today remaining at different, frequently early stages of development, they allow given assumptions to be verified and lines of action to be corrected, so that they would meet the expected needs in the best possible way. This might be one of the factors facilitating the introduction of sensor or sensor-like networks into common use. This situation brilliantly illustrates the often overlooked significance of an appropriate and fruitful combination of research activities and market operability. In the area of sensor networks, a number of research investigations outpace, by decades, the current market needs or the implementation capabilities available. As a result, despite their research excellence, they might never be adopted in practice in their full and extensive forms. Innovative business enterprises may not be able to undertake a risk or accept costs of an implementation of too complex and seemingly expendable ideas. No wonder then that they tend to choose ready-made and easily available components and proven solutions that make it possible for them to focus on functionality issues rather than on details of all-technical aspects. This, however, is often done at the cost of getting attached to license-fee paying technologies that are not fully open and, more importantly, are developed by some other companies.

Business enterprises chose to provide *minimum viable product* (MVP) as quickly as possible, i.e., such a product that would in the most favorable way satisfy the expectations of the first group of users and would allow the product to be further developed [80]. It is important then that scientists in their research efforts are able to follow the market developments and try to understand current and future needs of prospective users and, wherever possible, check and streamline their new ideas in close cooperation with operators of already existing networks and systems [81]. Moreover, socially-oriented and valuable results may be achieved when research investigations provide opportunities to transform them into real systems. Then, by gaining practical relevance, they will have a chance to enter the mainstream and be appreciated by standardization organizations, thus, at least to a certain degree, be in a position to shape the way sensor networks are implemented in the future.

This approach can and should be applied to current routing-related problems that pertain to urban sensor networks. The research on opportunistic routing, data aggregation and offloading, as well as, on topology control and modeling has to be conducted in close relation to technological advancement. Therefore, modeling and analysis methods introduced in the next chapters combine practical observations, scientific regime, and the usage of state-of-the-art tools and data sources.

Chapter 3

Sensor nodes deployment in urban environment

Node deployment is the basic element that influences the way the topology of a sensor network is controlled [38, 82]. Moreover, the empirical practices related to topology control in implementation-oriented ventures, as discussed in the following sections, seem to focus, first and foremost, on the distribution of nodes.

In real applications, depending on particular requirements or environmental conditions in which a given sensor network operates, random [72] or deterministic [83] distribution of network nodes may be distinguished. This distribution may be predicted at the designing stage, or can be partly or totally random in a dynamically changing working environment. In many real applications it is difficult or even impossible to assign a given type of node distribution within a network (or a part thereof) to just one of two categories. One should not forget that each sensor structure is characterized by a certain degree of determinism (therefore also a degree of randomness) that will vary along with changes in the external environment, though to an extent that can be neglected in some applications.

The following sections discuss both types of seemingly obvious node deployment schemes that may be encountered in urban environments, i.e., random deployment and deterministic deployment, and provide examples of their implementations.

3.1 Random deployment

Since the very beginning of work on WSNs, a general view has prevailed in the literature of the subject that random deployment of nodes [72] is the fundamental approach, serves as the point of departure while constructing WSNs and is typical of this group. One of the basic areas of application for such a network is monitoring the parameters of the natural environment (e.g., temperature or pressure) [84]. Much attention has been then given to methods for random deployment of sensors, while one of the most frequently mentioned examples illustrating the above would be a situation in which sensor devices are dropped from an aircraft over the area to be monitored [85].

The sheer multitude of potential applications of sensor networks that has been identified over nearly two past decades has led to numerous complex concepts related to the construction of the networks' physical topologies. Many of them depart from randomness of node deployment, understood in the direct and unconditional manner, by introducing some kind of order. In the author's opinion, one of the most interesting scenarios is a network in which the sensors are deployed in an unknown working environment, with the process carried out according to a predefined algorithm and based on information obtained during actual deployment [86]. Other networks that should also be noted within this context include VSNs, in which a sensor network typically covers the intended area—a road and its closest surroundings—and may be spread over tens or even hundreds of kilometers. In the case of such networks, one may speak of a combination of randomness and determinism, i.e., a certain portion of the sensors are deployed permanently alongside the road, while the sensors that communicate with them are those that may be deployed in vehicles. Their distribution is random in such a case and they frequently remain beyond the control of sensor networks' operators [87]. Vehicles may be then viewed as mobile agents that perform not only tasks assigned to them, but also additional functions that relate to stationary sensors [88]. Currently, it is rare that nodes in such systems communicate between one another, because typically the exchange of information is performed with a central point only (e.g., a control center or a data collection software). Furthermore, in the case of VSN, it is common to omit issues related to limitations of energy sources or computational power—so important in traditional WSNs [89]. This is why the largest number of implemented examples of sensor-type and sensor-like networks may be found in the group of systems related to transport and presented in the subsequent subsections. It is worthwhile noticing, at this point, that the largest systems of this type, based on smartphones that serve as mobile agents, form today's most widely used sensor networks and that their number and scope of functionalities are depicted by a continuous upward trend.

3.1.1 Mobile measurement agent within INEA network

Following the research studies initiated by the author and carried out in June–July 2016 together with the associates from INEA, a regional Polish telecommunications operator, it was possible to perform measurements related to the operation of radio networks that are based on the IEEE 802.11 family of communication standards [90]. A detailed description of the tests and an analysis of the results obtained in the mobile and residential *wireless local area network* (WLAN) are presented in [17], while the conclusions from this work are presented below.

The first group of experiments was performed using access points providing wireless Internet access, with the use of the 2.4 GHz band, to the passengers of 330 public transport vehicles in Poznań and Konin, two cities in the Greater Poland region. The movement of those agents was beyond any control of the telecommunications network operator because it was the transport operator that decided about the movement of the vehicles involved, at the same time impacting the topology of the network. By relying on devices known as RouterBoard RB751U, deployed in trams and buses and equipped with a 2.5 dBi antenna and a 4G cellular network modem-based uplink (providing Internet connection), a collection of samples was performed in each vehicle once every 15 minute. The important parameters included noise floor level (background noise), expressed in dBm, and

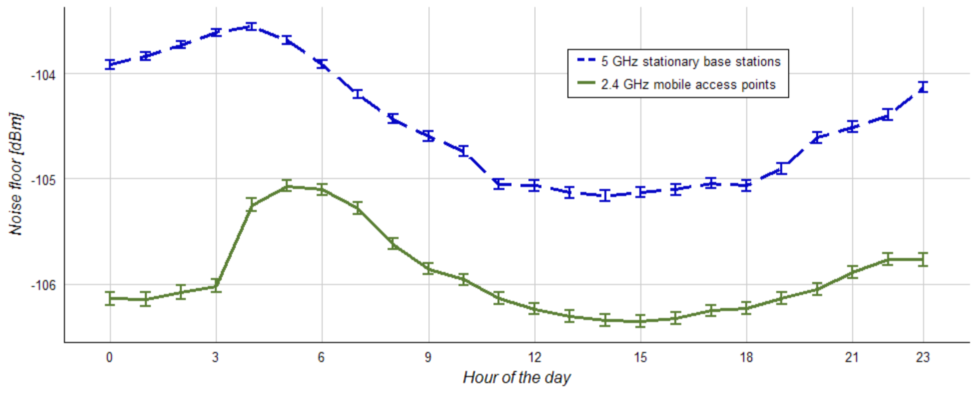


Figure 3.1 Average noise floor observed with INEA's IEEE 802.11 access points

Source: [17]

the values of the *received signal strength indicator* (RSSI) related to each of the connected users were recorded. In this way, reliable around-the-clock distributions for the urban environment were obtained, thus allowing a technical and a business analysis useful for INEA to be performed.

The around-the-clock distribution of the average noise floor that occurred over the period of one month was particularly interesting. In order to verify the distributions observed, they were compared with the results obtained with the help of 10 stationary INEA access points operating in the 5 GHz band, located on masts and on rooftops. It turned out that both trends were nearly identical, which is clearly visible in Figure 3.1. Each point in the graph corresponds to the average value from a given month, whereas confidence intervals correspond to standard deviation. Because the comparison involved a mobile network and a stationary network, both operating in different environmental conditions and with the application of antennas with a much higher gain and on other frequencies, the conclusion drawn is the changes in noise floor were mainly caused by external factors and were not typical of human activity. Further studies were conducted in 2017 with the use of a stationary 3.5 GHz IEEE 802.16e WiMAX network, yielding similar characteristics and suggesting that they may result, in addition to other factors that are yet to be identified, from ambient temperature changes [91].

The other group of experiments involved issues related to the operation of more than 20 thousand fixed residential Wi-Fi hotspots, i.e., IEEE 802.11 access points that were located at INEA customers' homes. In this service model, each INEA subscriber takes advantage of a *community WLAN* service based on home routers connected to the INEA HotSpot WiFi network.

The experience of mobile users was verified using a smartphone equipped with an IEEE 802.11a/b/g/n/ac (2x2 MIMO) radio module, GPS and GLONASS navigation receiver, as well as measurement and communication software. The test to be carried out involved measurements of signal parameters and extraction of technical information for each *basic service set identifier* (BSSID) observed. During the test, the user was moving, with the smartphone, on the sidewalk, between multi-story apartment buildings in council housing estates built in the 1970s. It was observed that 313 out of 1874 BSSIDs used the *service set identifier* (SSID) with the name INEA HotSpot WiFi. The tests performed indicate that 59% of INEA residential hotspots could have been used for conversations with the use of *voice over IP* (VoIP), provided that the user standing on a sidewalk was connected to the access point and that the strength of the signal received was not lower than -75 dBm.

3.1.2 Yanosik driving assistant and notiOne location beacon

The development of mobile radio-location systems, as well as increasingly common packet data transmission has led to widespread use, among drivers, of sensor systems based on mobile agents. No statistical comparisons are available that would illustrate the popularity of particular solutions, but, based on market observations, one may come to a conclusion that the Yanosik driving assistant, operated by a Poznań-based company named Neptis, is the leading platform among Polish drivers. On the global scale, such applications as Google Maps, HERE WeGo and Waze are based on similar concepts.

The Yanosik driving assistant is primarily a system used for exchanging information and warnings between drivers, relying on dedicated devices and smartphones (a special free app has to be installed). Both devices need to be wirelessly connected to the Internet (usually via a cellular network) to act as mobile agents. Vehicles (devices) do not communicate directly with other road users because the entirety of the exchange of information is performed on the central operator's host platform. This network has a dynamically variable random topology of the logical star type, in which the location of nodes depends on the decisions of the drivers and situation on the road.

Drivers who report events, such as road accidents, road works or a police patrol, are the source of information. Along with a report, additional information is forwarded on the user's location. A smartphone with the Yanosik app may be also used as a navigation system with real-time traffic service. As indicated in a release issued to the author by Neptis in April 2017, at its peak times, the system is used by over 150 thousand concurrent users, which means that it comprises the same number of sensor nodes. The further processing of data makes it possible to develop real time traffic intensity maps (Figure 3.2). In addition, the author was shown the results of an investigation that demonstrated the use of information gathered from sensors embedded in smartphones (accelerometers and gyroscopes) in order to evaluate the quality of roads and driving comfort. This enabled data on

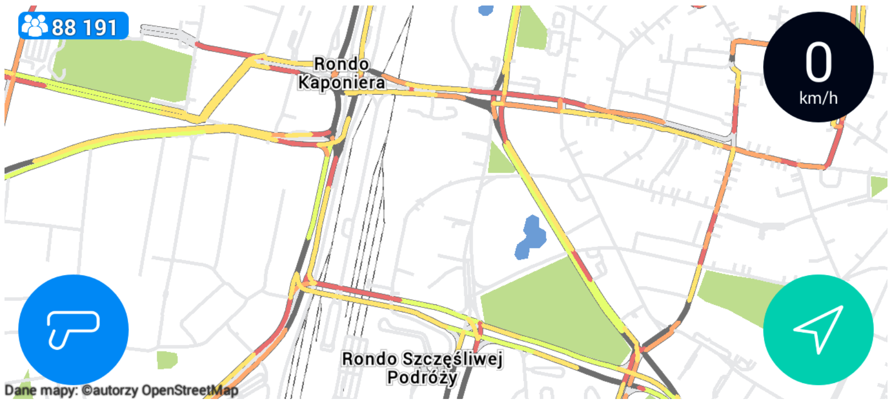


Figure 3.2 Poznań city center road traffic map presented by Yanosik (ver. 3.1.1.1)
9 May 2017

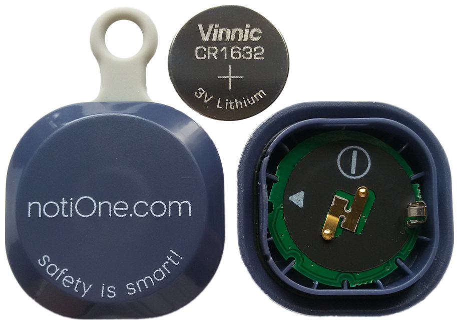


Figure 3.3 Opened notiOne beacon with a CR1632 button cell battery.

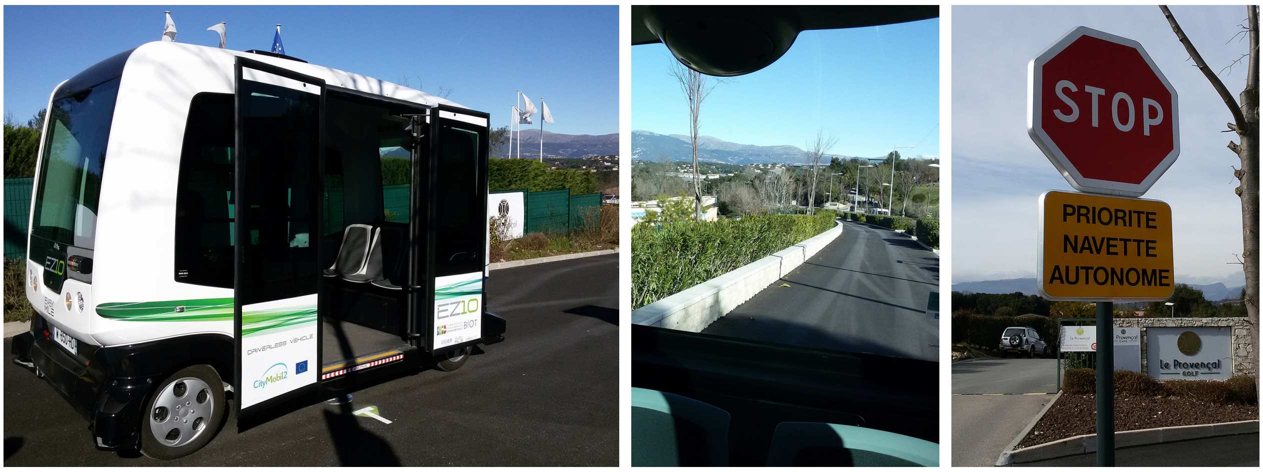
vibrations experienced by drivers and vehicles, while in motion, to be analyzed. The smoothness of traffic flow and the average speed were studied as well.

In more complex network topologies, smartphones offer wireless interfaces of different types and make it possible to connect with devices of other types to execute additional agent functions. This provides a basis for the operation of devices known as notiOne, which are the so-called hardware beacons (see Figure 3.3). These simple and small-scale mobile transmitters broadcast signal that includes a device identifier and may be received by smartphones that happen to be in the vicinity. In this way, the location of the beacon is approximated based on the accurate location of the smartphone that received the beacon's signal. Effectiveness and accuracy of such geolocation depends on the number of nearby smartphones on which software co-operating with the system has been installed.

The beacon may be attached to a key pendant or a dog collar, so that if lost, it makes it possible to easily obtain information on its location by means of a dedicated app. The location highlighted on the map will indicate the spot where the beacon's signal was received for the last time. The notiOne device is powered by a button cell battery that allows the device to work for nearly one year, with the range up to 90 m using Bluetooth 4.0 Low Energy connectivity.

3.1.3 The Veniam system

The activity of Veniam serves as a good example of a successful implementation of research work on mobile sensor networks in a commercial product. The company's founders, Barros, Cardote, and Sargento, were previously involved in research on mobile and sensor networks, among others [92–95]. In 2003, as a result of their work, commissioned by city authorities, Veniam launched in Porto, Portugal, a mobile wireless network targeting to collect the results of measurements performed in the urban environment and improving the operational efficacy of the city'stransport and service utility vehicles. In 2016, the network comprised nearly 350 vehicles (buses, police cars, garbage trucks, taxi cabs) [96]. These vehicles are equipped with a device that serves as an access point (with a network interface, etc.) known as NetRider. To provide connectivity, radio base stations are used, operating in the bandwidth of 5.9 GHz according to the IEEE 802.11p standard for the mobile environment, in particular for the so-called *intelligent transport systems* (ITS) [97]. Each access point serves as an IEEE 802.11g Wi-Fi hotspot and makes the Internet available to passengers. Sensors are placed inside vehicles to make environmental measurements and to monitor the fleet. The vehicles may communicate with one another and serve as mobile transceivers (mobile relays), making indirect,



(a) EasyMile EZ10 (b) Dedicated bus lane (c) Autonomous vehicle precedence sign

Figure 3.4 Autonomous vehicle demonstration route in CityMobil2 project
Sophia Antipolis, France. March 2016

real time communication between the control system and the out of range vehicles possible. In turn, when communication in real time is not possible or suffers from delays, data MULE function, i.e., data extraction and temporary buffering, is performed. If a vehicle that passed near a stationary sensor located at the edge of the road is just outside the base station coverage, the reading is taken (receiving a portion of data using IEEE 802.11 or Bluetooth) by using the local memory of the mobile agent, with data forwarded to the central repository once the connection with the base station has been reestablished [98]. The system employs a complex and variable topology of connections that is successfully created by means of different models and methods for wireless communication.

The author has had a chance to examine the Veniam network's control and management panel. The virtual environment software makes it possible to monitor the network in real time and to store and analyze the data extracted. The location of any vehicle is presented, just as are the estimated range of the hotspot and the live traffic intensity map for the area covered.

3.1.4 CityMobil2 automated road passenger transport

Transport automation is another area of sensor applications that is currently under development. One of the leading European projects in this area was CityMobil2, launched in 2012 and concluded in August 2016, co-financed under the European Union's Seventh Framework Programme [99]. The goal was to create a pilot automated passenger road transport platform with automated and autonomous vehicles, and to carry out tests in a number of European urban environments [100]. The platform was made up of electric vehicles, i.e., mini buses equipped with wireless interfaces for communication with the control center, as well as with sensors necessary for the unmanned vehicles to operate. The entire system used a central controller that gathered data transmitted by the vehicles and controlled the vehicle movement, hence the physical network topology. In addition, the personnel was capable of override the control system.

In 2016, the author visited a demonstration route used by autonomous vehicles in the French Sophia-Antipolis technology park, and took a ride in the EasyMile EZ10 autonomous vehicle shown in Figure 3.4. The one-kilometer test lane with five stops was used by three autonomous shuttle service vehicles, each with the capacity of 9 passengers. By using a GPS receiver, proximity detectors and accelerometers, the vehicles were capable of adjusting the driving speed to other road users and avoided obstacles or, alternatively, stopped before them if avoiding the obstacle was impossible. When this was the case, the vehicles were sending information to the control center, requesting operator's intervention.

The primary reason for the operator's presence in the autonomous vehicle was, according to the operator, the restriction imposed by applicable French legal regulations that do not allow vehicles to be admitted to streets and roads without a person authorized to drive them on board. The other reason was the occurrence of a potentially dangerous situation, due to the pilot stage of the project. In fact, this turned out to be necessary when, for example, the board computer crashed, when a situation occurred on the road that had not been foreseen in the control algorithms applied, or when an uncontrollable panic attack took place among the passengers.

During the tests, a collision of a car driven by a human and one of the unmanned vehicles occurred at the only crossing of the dedicated bus lane with a general traffic road, and as a result the presence of the operator also turned out to be necessary. Following this accident, to make the test and demonstration route more conspicuous to other road users, it was additionally marked with noticeable posters, and a STOP traffic sign with a visible note "*priorité navette autonome*" (French for "*priority for the autonomous shuttle bus*"), as shown in Figure 3.4c, was installed.

The tests have shown how diversified problems need to be foreseen and predicted while designing autonomous systems operating in a dynamically changing environment. Moreover, more integrated and complex information is to be extracted by means of different sensors, enabling a fast and reliable interpretation of the road traffic situation.

3.2 Deterministic deployment

Deterministic deployment of nodes in WSNs is the second category related to the process of creating physical topologies. For example, some of industrial WSNs are capable of using fixed or controllable node deployment (distribution) schemes. This type of a network would be tasked, for example, with monitoring the vibration signatures to predict maintenance needs [101]. Manually deployed sensor networks with cameras and microphones [102] are also being considered for implementation. In addition, in various research projects connected with natural environment monitoring, sensors are deployed manually. This makes it possible to adjust network topology to the nature of phenomena observed and to the assumptions based on which the experiments are to be carried out. As often as not the location of nodes remains deterministic (in most cases it changes slowly or remains fixed) during their service life (e.g., this is the case with investigations concerning volcanic phenomena [82]). The deployment of nodes or the range of their relocation makes it possible to prolong the operating time of a network. This can be also achieved, for example, by securing such distances between sensors that would enable the routing mechanisms applied to remain operable in the most effective way, without the need to manipulate the power of transceivers [38, 71].

In consumer applications, home automation systems (otherwise known as smart home systems) become increasingly popular. Usually, they have the features of a small-scale sensor network and take advantage of wireless communication protocols, such as Bluetooth Low Energy, ZigBee, Z-Wave and 6LoWPAN [103]. Their physical topology is usually determined at the installation stage due to the operational range being limited to just one property, whereas network communication takes place predominantly directly between the node and the base station (control center), and only occasionally with the use of intermediary network nodes [104]. In multi-family houses, radio-enabled electricity meters [105] or water meters [106] may also be found, often placed inconspicuously but effective in performing their measuring functions.

3.2.1 LoRaWAN metering infrastructure

As presented by the author and associates in [107], in case of LoRaWAN-based sensor networks, the deployment of gateways is of key importance. In the discussed scenario, the distribution of electricity meters is strictly related to the location of power consumers, e.g., private households, institutions, and enterprises. Therefore, sensor network designers have no direct influence on the distribution of those sensing nodes. The aspects they can determine are, though, the number and location of network gateways (base stations), as well as, the operational parameters of the network. In this way, the topology can be controlled in a manner which will ensure desired coverage, reliability and wireless medium utilization. As a result, star-like LoRaWAN connectivity, please see Figure 3.5, can be used to gather, process, store and visualize the data of interest.

In network planing process, to achieve optimum coverage and avoid unnecessary interference, machine learning and reliable radio loss models were used. Figure 3.6 shows a random network topology obtained when *COST-231 Hata* propagation model [108] and *K-means* clustering algorithm [109] were used. It was assumed that each node had the same radio range and operated with $SF = 7$ (*Spreading Factor*). As a result, nodes were partitioned into k clusters centered at k gateways. In Figure 3.6a it is visible that with $K = 3$ there is a number of outliers which are not

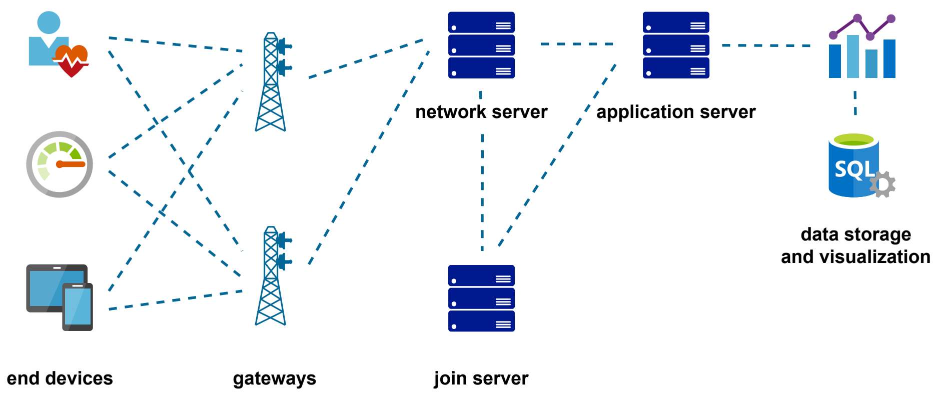
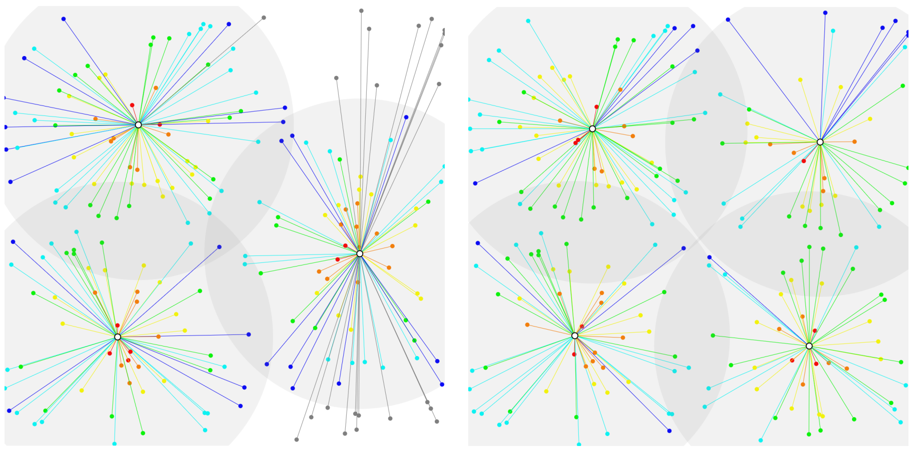


Figure 3.5 LoRaWAN-based sensing topology and system architecture

Source: [107]



(a) $K = 3$ (b) $K = 4$

Figure 3.6 Exemplary LoRaWAN gateway deployment planed with K-means algorithm
Source: [107]

able co communicate with a gateway. When the number of clusters gets increased and $K = 4$, one additional gateway is to be deployed. By this extension, all meters will be located within the radio range of at least one base station depicted in Figure 3.6b. This approach has been discussed and verified in synthetic scenarios, as well as, in the actual network structure modeled in Bydgoszcz, the city with around 355 thousand inhabitants, located in northern Poland.

3.2.2 Sensor network in Cisco openBerlin Innovation Center

The sensor network launched in Berlin, Germany, at the Cisco openBerlin Innovation Center, is an example of the deterministic deployment scheme. The network is used both as a backbone of a smart building system (Figure 3.7a) and as a testbed on which research projects of companies affiliated with openBerlin are evaluated [110].

(a) Cisco openBerlin Innovation Center (b) Bosch XDK110 based sensor network

Figure 3.7 Sensor network in Cisco openBerlin Innovation Center
Berlin, Germany. February 2016

The author examined the components of the test and demonstration set, as well as verified the topology of the network deployed within the building that consisted of several hundred nodes. The node hardware is based on the Bosch XDK platform. The devices are located at different places, including cable support ceiling systems in the office section or near ceiling joists in the recreational section, as shown in Figure 3.7b, and are marked with dotted circles. The Bosch XDK110 node is a hardware component equipped with a 32-bit ARM Cortex-M3 microcontroller, 1 MB Flash memory, 128 KB RAM, a Micro SD card reader, Bluetooth 4.0 Low Energy and IEEE 802.11b/g/n modules, 560 mAh rechargeable battery and contains 8 sensors: an accelerometer, a gyroscope, a magnetometer, as well as humidity, pressure, temperature, acoustic and light sensors. The network is used as a source of data for the system developed by an IoT company known as Relayr, and allows temperature and lighting inside the building to be controlled.

3.2.3 Fibaro home automation system

The Fibaro home automation system comprises a host controller that wirelessly manages the attached sensors and actuators (Figure 3.8). The sensors include smoke, flood and motion detectors, door or window opening sensors, as well as a universal device that allows any sensor with a binary output to be added to the system. In addition, such components as a switch-key in the electric wall socket enclosure with an energy consumption measurement functionality, roller and gate shutters, lighting controller or relay switches may also be used. The system is capable of co-operating with a home weather station, wireless speakers or cameras.

It should be stressed that components of the Fibaro systems communicate via the Z-Wave protocol, and not via IEEE 802.15.4 ZigBee [111] that is in common use in research environments. The Z-Wave protocol was initially developed by the ZenSys, and was then largely used by the standardization organization ITU-T for the development of the G.9959 recommendation [112]. Radio communication relying on the Z-Wave protocol uses, in Europe, primarily the 868.42 MHz frequency band, but systems for the 2.4 GHz band are available as well. Data is routed between network nodes by assigning identifiers, whereas the throughput is not more than 40 kbps [103]. For a commercially available device to be capable of making use of the protocol, it has to be equipped by the manufacturer with a Z-wave communication module, sold separately, and then certified for interoperability to comply with the license agreement.

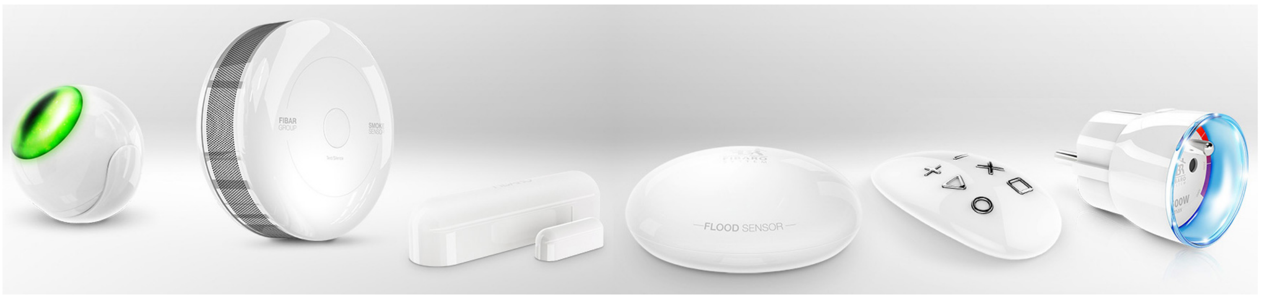


Figure 3.8 Components of FIBARO home automation system

Graphics courtesy of Fibar Group S.A.

3.3 Observations

In today's real-life sensor network deployments that enjoy an established reputation among their users and are successfully implemented in urban environments, one may primarily list only those that treat the idea of WSNs in a rather simplified manner, or those that have been even developed in isolation from elaborate and complex scientific research. This is particularly visible in the area of topology control that can be analyzed and executed as a complex and multifaceted design problem, while it is still hard to find good examples of its implementations in which it would constitute one of the main issues. It seems that, in practice, the prevailing model of implementation is empirical, i.e., draws from experiments, past experience, best practices and intuition. In some implementations, network topology is even unknowingly or deliberately pushed aside from the areas of interest or remains beyond any significant interest of network designers involved in a project—as a component of industry standards or licensed protocols implemented in building blocks relied upon.

Therefore, the primary or exclusive aspect of topology control in urban environment, as presented, is then reduced in its essence to the deployment of nodes. In applications related to transport, randomness and variability of node deployment, and hence the physical topology of the network, still remain the dominant element. In applications encompassing building automation systems, in turn, it is the deterministic and static distribution of nodes established during the installation of each of the components that remains dominant.

Chapter 4

Modeling real-life urban sensor networks based on open data

The main objective of this chapter is to present recent development and usage possibilities of publicly available real-life diversified data sources as the cornerstone of a novel urban WSN-related research approach. This approach is based on actual node data and location, as opposed to relying on synthetic models or limited homogeneous historical data, and paves the way for modeling of realistic space- and time-changing graphs that can be visualized, stored, and analyzed to design and optimize various aspects of the networks. Presented methodology is, to the best knowledge of the author, the first attempt to model and analyze dynamic heterogeneous urban sensor networks as graphs based on real node location data.

4.1 Network modeling data sources characteristics

The variety of node types and routing research problems in sensor networks operating in urban environments is followed by the growing diversity of real-life and real-time data sources. In spite of each source having its own unique features, a number of more general characteristics can be distinguished, as listed and described in the following sections.

4.1.1 Data availability

The most basic division of digital data source types one can currently think of is related to whether the data are available online or offline, as well as whether the data are provided in real time or as an archived dataset.

The largest group of real-time data sources appears to be the one related to public transport. They can be found and used either based on the information provided directly by the transport service providers or in dedicated repositories, such as, Mobility Database which lists open mobility data feeds from around the world [113].

Online and offline repositories of historical data (archives) may be of great value, especially when they provide rich research data. The challenge with such repositories is that they are currently few in number and provide selected datasets, such as the ones in the CRAWDAD repository mentioned in Section 2.3.3. Some enable access only to partial, or not necessarily up-to-date data [114], while others require formal efforts and agreements or do not provide any access to external researchers at all.

The present study suggests that there are no heterogeneous frequently updated high-quality data archives available to urban sensor network researchers. Hence, as long as no such repositories are in existence, the use of publicly available online data, referred to as *open data*, shall be considered. Such sources offer much easier access, usually to real-time or quasi-real-time data, and are more promising in terms of novel research areas and approaches. Moreover, similarly to closed offline archives, there exist numerous systems that could provide real-time online access to valuable *closed data* if they are approved by a responsible authority.

Real-time sources gradually grow in numbers and the data are usually provided based on *REpresentational State Transfer* (REST) [115] APIs. To obtain the data, first an *HyperText Transfer Protocol* (HTTP) GET request needs to be sent to a particular resource address, i.e., a *Uniform Resource Locator* (URL), often called an endpoint. Then, as a result, an HTTP response containing the requested dataset is returned by the server. Sources of this type are of interest in the next sections. Less frequently, the data are provided as more static files that need to be downloaded and unpacked. Gathered data can be used in a real-time application, as well as stored for further usage in a solution-specific custom-made repository (e.g., the one used in the present research and described in Section 4.2).

4.1.2 Data provider

The data are, or can be, provided by a number of entities which maintain and manage measurements gathering and processing systems. They include global communities and commercial enterprises, as well as regional and national authorities, public services providers, etc.

A good example of the above is the Packet Broker Mapper API which enables open access to the data related to the global IoT ecosystem of *The Things Network* (TTN), providing the locations of LoRaWAN gateways, number of their antennas, online status, etc. [116]. Then, there are the open data related to stationary air quality measurement stations in Poland provided by commercial providers Airly [117] and Syngeos [118], as well as the *Chief Inspectorate of Environmental Protection* (GIOŚ). The *Air Quality* portal operated by this national authority provides access to both archived [119] and real-time data API [120]. Example responses of this portal are presented in Figure 4.1. In some areas,

There is also an increasing number of regional authorities that operate their own open data websites and portals related to public services [53, 121, 122]. Very often, the *Comprehensive Knowledge Archive Network* (CKAN) open-source software [123] is used at the core of those portals [114, 124, 125]. This management system allows data sources to be grouped, described, and presented in a user-friendly way. Each dataset can be made available in a number of data formats and enriched with metadata and access methods examples. CKAN operators can use the DataStore extension to automate data update and retrieval processes [126]. Once a resource is updated, its preview page can be automatically refreshed with the Data Explorer extension, to be ready to be presented to the end users. Moreover, DataStore API enables users to search, filter, and fetch the

```json
{
    "id": 944,
    "stationName": "Poznań, ul. Dąbrowskiego",
    "gegrLat": "52.420319",
    "gegrLon": "16.877289",
    "city": {
        "id": 729,
        "name": "Poznań",
        "commune": {
            "communeName": "Poznań",
            "districtName": "Poznań",
            "provinceName": "WIELKOPOLSKIE"
        }
    },
    "addressStreet": "ul. Dąbrowskiego 169"
}
```

(a) Measurement station

```json
{
    "id": 4681,
    "stationId": 729,
    "param": {
        "paramName": "pył zawieszony PM10",
        "paramFormula": "PM10",
        "paramCode": "PM10",
        "idParam": 3
    }
},
{
    "id": 28227,
    "stationId": 729,
    "param": {
        "paramName": "pył zawieszony PM2.5",
        "paramFormula": "PM2.5",
        "paramCode": "PM2.5",
        "idParam": 69
    }
}
```

(b) Sensor characteristics

```json
{
    "key": "PM10",
    "values": [
        {
            "date": "2022-10-06 12:00:00",
            "value": null
        },
        {
            "date": "2022-10-06 11:00:00",
            "value": 23.9628
        },
        {
            "date": "2022-10-06 10:00:00",
            "value": 11.2768
        }
    ]
}
```

(c) Sensor readings

Figure 4.1 Sample data provided by GIOŚ Air Quality portal
6 October 2022

data without having to download the whole dataset. Hence, a PostgreSQL query can be used, for example, to limit the number of returned records, parameters, or even to convert their data types.

It needs to be pointed out that open data are currently a relatively limited source of information, as compared to closed data systems. Such systems, that belong to smartphone manufacturers, navigation software providers, etc., operate based on much larger, diversified, and constantly changing volumes of data related to the location and operation of each device in the network. Mobile nodes, in general, tend to generate larger amounts of measurement data than the stationary ones. A good example of heavily monitored solutions is the segment of vehicles used in the innovative solutions for *shared mobility* (also called car-sharing, bike-sharing, etc.), which is a new, distinct, and evolving category of urban mobility [127]. It includes various types of vehicles for individual and commercial use (e.g., cars, bicycles, scooters) [128–131]. This transportation strategy allows users to gain short-term paid access, on an as-needed basis, to various types of vehicles widely dispersed throughout the city, such as the bikes and scooters presented in Figure 2.3. Once the trip is completed, the vehicle becomes available for subsequent users. Gathered measurements are primarily accessible to and used by the operators of such networks with no open data access provided. Although, based on special agreements, access may be granted to interested third parties, e.g., to the developers of mobile applications, such as take&drive [132], which aggregate and present the data form multiple sources, i.e., related to numerous urban means of transportation. Moreover, other types of car fleets, trucks, and service vehicles (e.g., ambulances, police cars, and garbage trucks), as well as taxis and delivery vehicles, are monitored as well. Depending on local and national regulations, access to closed data related to public infrastructure and service vehicles, in particular to the archived location records, might be possible, upon request, based on the right of a citizen to access public information [133].

```json
{
    "lastUpdate": "2022-10-16T21:01:04Z",
    "vehicles": [
        {
            "generated": "2022-10-16T21:01:01Z",
            "routeShortName": "12",
            "tripId": 221,
            "headsign": "Strzyża PKM",
            "vehicleCode": "1063",
            "vehicleService": "012-08",
            "vehicleId": 145753,
            "speed": 10,
            "direction": 90,
            "delay": 61,
            "scheduledTripStartTime": "2022-10-16T20:41:00Z",
            "lat": 54.34904098510742,
            "lon": 18.61301040649414,
            "gpsQuality": 3
        }
    ]
}
```

(a) Tram in Gdańsk [137]

```json
{
    "header": {
        "gtfsRealtimeVersion": "1.0",
        "incrementality": "FULL_DATASET",
        "timestamp": "1665953154"
    },
    "entity": [
        {
            "id": "6051",
            "vehicle": {
                "trip": {
                    "tripId": "4_1799332^+",
                    "scheduleRelationship": "SCHEDULED",
                    "routeId": "603"
                },
                "position": {
                    "latitude": 52.33697,
                    "longitude": 16.89085,
                    "speed": 19.17
                },
                "currentStopSequence": 18,
                "timestamp": "1665953144",
                "vehicle": {
                    "id": "6051",
                    "label": "603/31"
                }
            }
        }
    ]
}
```

(b) Bus in Poznań [138]

Figure 4.2 Sample public transport vehicles data
16 October 2022

4.1.3 Data format

The most common data export formats used by open sources are textual. Currently, the leading format is the *JavaScript Object Notation* (JSON), which is a data-interchange format (syntax) based on the object literals of JavaScript programming language [134], see Figure 4.1 and Figure 4.2. A far less frequent format is the *Comma-Separated Values* (CSV) [135], which can sometimes be enabled as an alternative to JSON access to CKAN-based data sources or to provide metadata related to a data source.

Binary data exchange formats are also being implemented and used relatively infrequently. The one which is actively developed and increasingly frequently used is the *Protocol Buffers* (protobuf) mechanism [136]. This method for serialization and deserialization of structured data minimizes the size of the message being transmitted (or a *.pb* file), while preserving its full content. The original message is encoded and decoded according to the message type definition stored in a fixed template defined for a given protocol (a *.proto* file). In this way, in contrast to a JSON file, the structure of a binary message is minimized and contains mostly a numbered series of values which correspond to specific parameters defined in the template. Therefore, the *.proto* file needs to be used at the receiving side to decode the message, i.e., to determine field names and types.

4.1.4 Data structure

It can hardly be said today that all sources follow the same well-established data structure. However, there are specifications to which source developers are increasingly turning to in order to standardize the matter. An example of JSON-based solution-specific convention is *GeoJSON* format, which was designed to represent geographic objects together with related attributes [139]. Another one, the *GTFS Realtime*, an extension to *General Transit Feed Specification* (GTFS), specifies the structure in which public transport operators shall provide real-time data related to their services. The specification includes the information on vehicle positions (location and congestion levels), trip updates (delays, cancellations, and route changes), and service alerts (unplanned travel or infrastructure events) [140]. The data are published in *Protocol Buffers*-based format.

Clearly visible, though, is the fact that numerous analyzed sources do not follow any common data structure. The hierarchy of the elements of a response differ, and usually, the meanings, naming conventions, and data types or accuracy of the elements vary. It can be seen in both CSV and JSON-based formats.

First, compare the structures of the public transport vehicle related data in Poland for Gdańsk (received as a JSON-based structure) [137] and Poznań (received as *Protocol Buffers*-based *GTFS Realtime* format and decoded to JSON) [138] presented in Figure 4.2. Gdańsk-related data are also available in the *GTFS Realtime* format, if required [141].

Then, please locate the position within the structure and the format of the timestamps:

- Figure 4.1c — *date* key related to *2022-10-06 12:00:00* reduced local time formatted string.
- Figure 4.2a — *generated* key related to *2022-10-16T21:01:01Z* extended complete UTC (*Coordinated Universal Time*) ISO 8601 formatted string [142].
- Figure 4.2b — *timestamp* key related to *1665953144* string in POSIX (Unix) time format [143].
- Figure 4.4b — no timestamp.

It also shall be noted that the latitude and longitude are not only given as different elements of the structure, but are sometimes of an unusual data type (i.e., a string):

- Figure 4.1a — *gegrLat* key related to *52.420319* string.
- Figure 4.2a — *lat* key related to *54.34904098510742* floating-point number.
- Figure 4.4a — *latitude* key related to *54.409971066405* floating-point number.
- Figure 4.4b — second element of *coordinates* array, i.e., the *52.412023* floating-point number.

Furthermore, as illustrated in Figure 4.4b, the language of the elements of the structure may be mixed. For example, the keys *type* and *street* are in English, whereas the payment methods *bilon* (coin) and *karta* (card) are in Polish, with Boolean values given as strings, i.e., *TAK* (YES) and *NIE* (NO).

Within the context of the presented differences, it is necessary to use a conversion software while gathering and processing data from a number of heterogeneous sources to standardize the data before further usage. To ease this task, the features of some CKAN-based sources may be taken advantage of to tune the structure and scope of the responses to the needs of the user using SQL-based queries. Figure 4.3 presents an example of such a request and response related to the data source [144] for

```
https://www.wroclaw.pl/open-data/api/action/datastore_search_sql?sql=
SELECT DISTINCT "_id" AS "id", "number", "uid" AS "station_id", "lat", "lng" AS "lon"
FROM "42eea6ec-43c3-4d13-aa77-a93394d6165a"
```

(a) SQL request to CKAN-based data source

```
{
    "help": "https://www.wroclaw.pl/open-data/api/3/action/help_show?name=datastore_search_sql",
    "success": true,
    "result": {
        "records": [
            {
                "lat": 51.099713,
                "lon": 17.027905,
                "id": 67,
                "station_id": "12500714",
                "number": "15068"
            },
            {
                "lat": 51.127192,
                "lon": 16.991832,
                "id": 171,
                "station_id": "16339566",
                "number": "15172"
            },
        ],
        "sql": "SELECT DISTINCT \"_id\" AS \"id\", \"number\", \"uid\" AS
        \"station_id\", \"lat\", \"lng\" AS \"lon\" FROM \"42eea6ec-43c3-4d13-
        aa77-a93394d6165a\""
    }
}
```

(b) Fragment of received data [144]

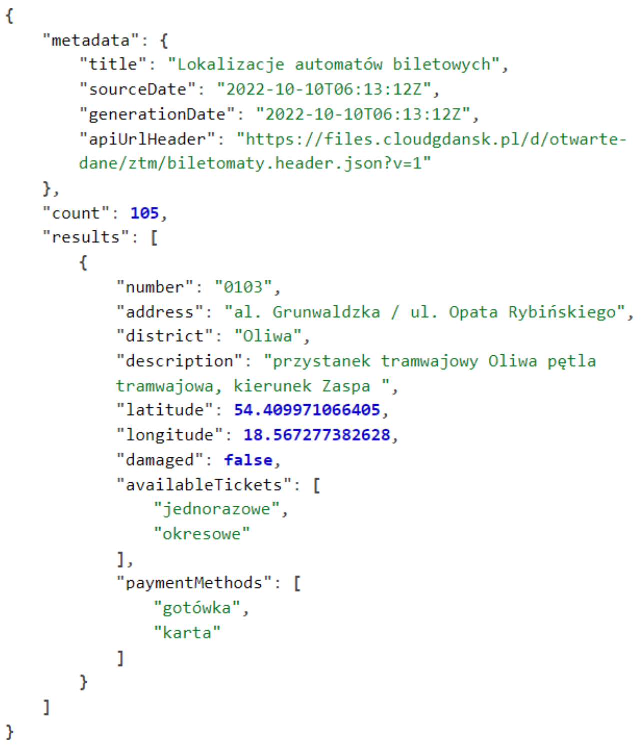
Figure 4.3 Sample server-side filtered and formatted Wrocław City Bike rental stations data
16 October 2022

rental stations of Wrocław City Bike [145]. The real-time information is filtered and shaped as the data consumer sees fit for the purpose. In this particular case, only unique (distinct) records were requested and received, the number of parameters (i.e., the scope of the data) was limited, and three original field names were altered by the server to reduce transmission overhead and computational efforts on the receiving side.

4.1.5 Data scope

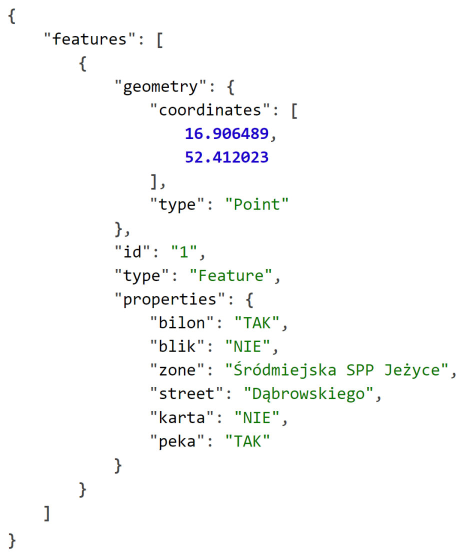
The scope of available data related to the elements of urban devices depends on both the decisions of the operator of the network and the nature and features of the device itself. Although different kinds of data are provided, one is of key importance in this study, i.e., the geographic location of stationary and mobile elements. The fundamental coordinates, latitude and longitude, are expressed in accordance with *World Geodetic System '84* (WGS 84) notation [146]. Other location-related parameters, such as altitude or street address, may be additionally present. Typically, some other application-specific features, such as the identifier, name, type, status, or capabilities of a node, are also parameterized.

In case of public transportation, additional information may include current speed of the vehicle, side number, and route or trip identifiers, as visible in Figure 4.2. Furthermore, correlated information may be available to determine the vehicle type (e.g., low-floor), equipment (e.g., ramp, air conditioning, voice announcement system, ticket machine, and USB chargers), and whether space to carry bicycles is provided [138, 147]. The timetables are commonly available as well [148, 149].



```
{
    "metadata": {
        "title": "Lokalizacje automatów biletowych",
        "sourceDate": "2022-10-10T06:13:12Z",
        "generationDate": "2022-10-10T06:13:12Z",
        "apiUrlHeader": "https://files.cloudgdansk.pl/d/otwarte-
        dane/ztm/biletomaty.header.json?v=1"
    },
    "count": 105,
    "results": [
        {
            "number": "0103",
            "address": "al. Grunwaldzka / ul. Opata Rybińskiego",
            "district": "Oliwa",
            "description": "przystanek tramwajowy Oliwa pętla
            tramwajowa, kierunek Zaspa ",
            "latitude": 54.409971066405,
            "longitude": 18.567277382628,
            "damaged": false,
            "availableTickets": [
                "jednorazowe",
                "okresowe"
            ],
            "paymentMethods": [
                "gotówka",
                "karta"
            ]
        }
    ]
}
```

(a) Ticket machine in Gdańsk [150]



```
{
    "features": [
        {
            "geometry": {
                "coordinates": [
                    16.906489,
                    52.412023
                ],
                "type": "Point"
            },
            "id": "1",
            "type": "Feature",
            "properties": {
                "bilon": "TAK",
                "blik": "NIE",
                "zone": "Śródmiejska SPP Jeżyce",
                "street": "Dąbrowskiego",
                "karta": "NIE",
                "peka": "TAK"
            }
        }
    ]
}
```

(b) Parking meter in Poznań [151]

Figure 4.4 Sample ticket machine and parking meter data

23 October 2022

Ticket machine and parking meter data can cover, for instance, district or zone name and supported payment methods, as in Figure 4.4. The air-quality-related data usually extend over the number of measurement types, such as particulate matter (floating dust) readings PM10 and PM2.5, presented in Figure 4.1.

What is crucial for real-time applications and data analysis is that each received dataset should be unambiguously timestamped—preferably at the source. Frequently, each record of a dataset is also timestamped—when the source distinguishes the moment of each obtained reading.

4.1.6 Data update frequency and quality

Online urban data sources belong to one of the three categories in terms of update frequency—frequent (i.e., almost real time), infrequent, or archival. Some are well documented in this matter, such as the source of frequently updated locations of public transport vehicles in Warsaw, Poland, which is updated each 10 s [152]. The source related to the *intelligent transportation system* (ITS) in Gdańsk captures the data, while the resulting delay is approximately 20 s for each vehicle independently. If a vehicle loses the connection to the data-gathering server, its position is not updated and is discarded after 5 min [137]. Some other sources provide no commentary on the update frequency, and this needs to be concluded from the analysis of the changes of timestamps (time markers) included in the responses. It tends to be no more than 5–20 s. To see these kinds of data and network dynamics in action, i.e., changing and visualized over a city map, BusLive [153], a mobile application, can be used. It presents (almost) real-time locations of buses and trams in a number of Polish metropolitan areas.

Similarly to the information related to mobile nodes, stationary sensing node parameters are also frequently changing, as the measurements progress in time. Some of those sources impose though free-access frequency limitations. In the case of Airly [117] and Syngeos [118] air quality measurements networks, those are set to the maximum of 100 requests per day.

Information on fixed infrastructure elements, such as bus stops, parking meters, and ticket machines, is less frequently updated. One can usually expect these data to be updated at least once every 24 h [154]. It seems that few archival data sources group related data by year [119] or by day [149].

Due to the open data nature of the sources of interest, there are usually no *service level agreements* (SLAs) that would guarantee the quality and continuity of the data. This results in some sources producing, from time to time, corrupted or outdated data. Moreover, because of technical problems, some reading series may be missing when the service is not (fully) operational. In spite of data providers aiming at sharing data of required quality and continuity, each researcher needs to take into consideration the fact that quite often there is no quality guarantee and, as a result, countermeasures need to be put in place to cope with broken or missing data.

4.2 Network modeling architecture

Due to the lack of ready-to-use heterogeneous historical node data, as well as no real-life data-based urban sensor network modeling architectures existing, as the result of the analysis in Section 4.1, the generalized architecture presented in Figure 4.5 was proposed. It adresses the research problem defined in Section 2.3.4 and the data is obtained, processed, and a network is modeled in the following steps:

1. Data gathering:
   (a) Query each data source;
2. Data processing:
   (a) Extract and clean received data;
   (b) Preprocess, integrate, and store the data to local archive (data storage);
3. Network modeling:
   (a) Retrieve the data of interest from local archive;
   (b) Model network topology and connectivity as a graph based on given modeling parameters and node attributes (e.g., position and type);
   (c) Solve network optimization problem (e.g., find a tree);
   (d) Calculate the metrics (attributes) of the resulting network (graph);
   (e) Save the graph in the archive for further use;
   (f) Visualize the network (with or without background city map).

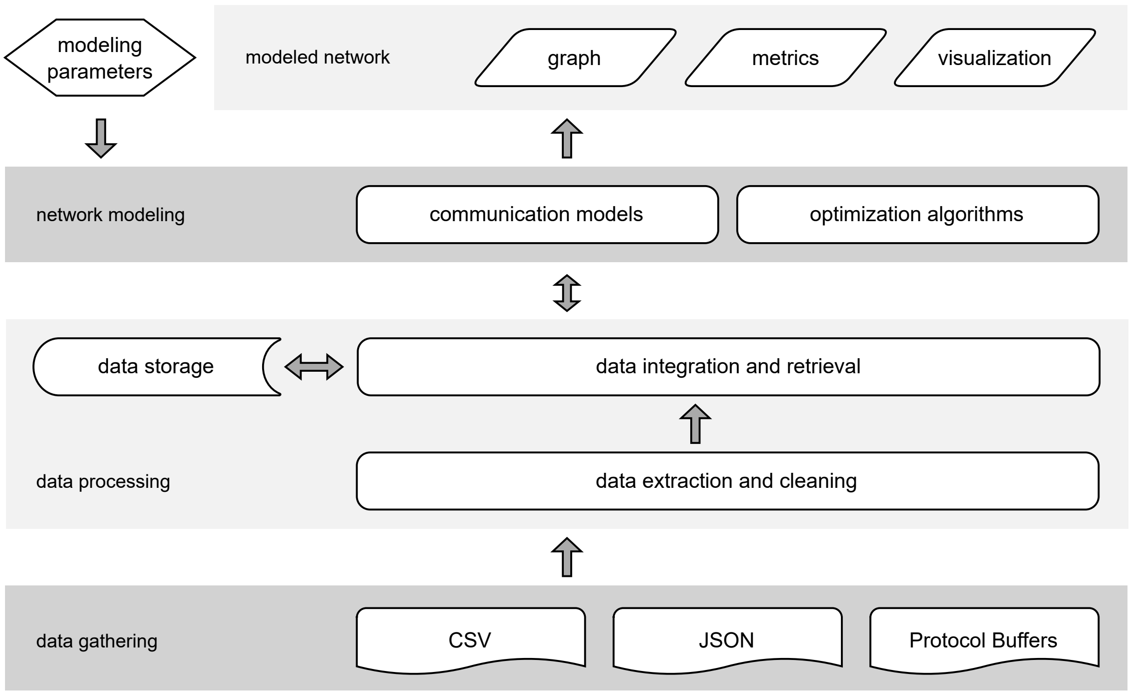


Figure 4.5 Open-data-based network modeling architecture

To implement this approach, Linux, PostgreSQL, and Python-based data-gathering software was developed to automate the process of continuous data gathering from multiple open data sources, as well as extraction, cleaning, integration, and storage in a local archive. Then, a graph representation and analysis environment was built on top of NetworkX—a network analysis library which provides basic data structures for graphs and implements numerous standard operations and algorithms [155]. Finally, static and dynamic (time-changing) graph modeling methods were implemented and a visualization based on the map data from OpenStreetMap [156] was introduced. Presented research environment was implemented based on these software frameworks both due to their availability free of charge to any interested user, and due to the fact that they provide numerous fundamental functionalities used by graph researchers, as well as, because of their high popularity and proven value for data scientists. The performance of such software depends on both the computing capabilities of the local platform and on the response times and transmission delays of open data sources. Should the need arise to, for example, assess the robustness of different available computing platforms, a simple execution-time-based comparison may be used that compares the time needed to construct a given graph or a series of graphs from locally stored data.

4.3 Network modeling algorithms

The generalized node data processing and network modeling algorithms are the key elements of network modeling architecture presented in Section 4.2. Figure 4.6 shows the complete modeling flow composed of algorithms introduced in the next subsections. As a result, both static (space) and

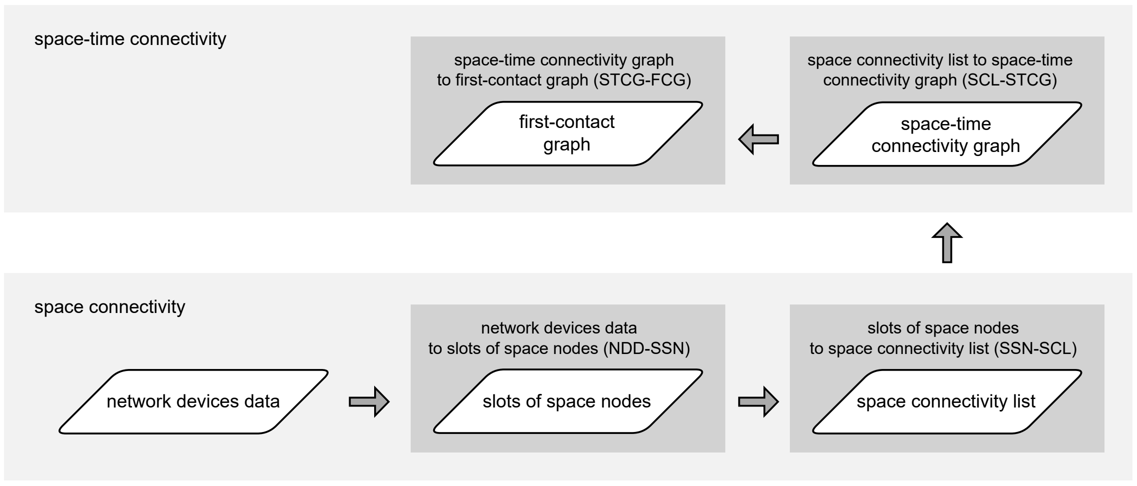

space-time connectivity
space-time connectivity graph to first-contact graph (STCG-FCG)
first-contact graph
space connectivity list to space-time connectivity graph (SCL-STCG)
space-time connectivity graph
space connectivity
network devices data
network devices data to slots of space nodes (NDD-SSN)
slots of space nodes
slots of space nodes to space connectivity list (SSN-SCL)
space connectivity list

Figure 4.6 Network modeling flow

dynamic (space-time) realistic graphs can be generated to enable graph-based analysis of network topologies and routing algorithms of interest.

The discussion of each algorithm is closed with a definition of time complexity. The usage of data structures implemented with hash tables is assumed, and therefore, average-case complexity $\mathcal{O}(1)$ applies for all basic data insertion, search, update, and deletion operations. These include, e.g., getting an element of a simple set, or accessing an element of a more advanced dictionary-like keyed structure. For this reason, the influence of this type of operations is not taken into account and centers presented time complexities on the very essence of the algorithms.

Time-changing graph representations and nomenclature

Although no methods for heterogeneous network modeling based on dynamic open data related to node locations existed so far, a number of more general, time-changing graph representations can be found in the literature. In short, they involve a time-ordered series of connected static graphs, a single graph with all node occurrences (instances) present as distinct vertices linked over time, or a single graph with node and edge attributes modeled as time series. A brief overview of the general concepts, terms and their evolution is presented in the next paragraphs. They will be used as a starting point and naming reference, to keep the description of introduced network modeling algorithms consistent with the graph theory domain.

Harary and Gupta (1997) suggested that a *dynamic graph* can be modeled as a sequence of static graphs [157]. Ferreira (2002) uses the name *evolving graph* for an analogical idea of a time-indexed ordered sequence of sub-graphs as the formal abstraction for *dynamic networks*. Here, each sub-graph corresponds to network connectivity at a given time interval. The time domain is used in this model to restrict paths in the graph from moving over edges which were possible only in past sub-graphs [158]. Li et al. (2017) call the *dynamic network* a *temporal network*, i.e., an ordered sequence of separate networks which consist of the same set of nodes [159].

Merugu et al. (2004) construct a layered *space-time graph*, where each layer refers to a discrete time interval in the observation period of a network. Each layer has one copy of every node in the network with the consecutive copies being linked by directed temporal edges. Traversing a temporal edge corresponds to carrying a message by a node. Separate nodes are connected with spatial edges and traversing those denotes forwarding a message from one node to another [160]. Huang et al. (2011) used this concept to model and investigate *time-evolving networks* in the DTN context. Each graph of this sequence of static graphs is called a *snapshot*, and the space between consecutive layers of a resulting *space-time graph* is named a *time slot* [161].

George and Shekhar (2008), in a study on *spatiotemporal networks*, call a layered graph a *time-expanded graph*, as opposed to a *time-aggregated graph* [162], which is a directed equivalent of a *temporal network* postulated by Kempe et al. (2000). Here, a *temporal network* is an undirected graph with edges annotated with time labels that indicate in which time interval the communication between nodes took place [163]. For Holme and Saramäki (2012), an *interval graph* is the one whose edges are active over a set of intervals [164]. Correspondingly, a *temporal graph*, as defined by Wu et al. (2014), consists of vertices connected by edges labeled with starting and edge traversal times [165]. Flocchini et al. (2009) investigated the usage of *time-varying graphs* whose links are defined by periodic movements of mobile agents in the context of *dynamic networks*, whose topologies change frequently in time [166]. Not only the aforementioned original papers, but also the introductions, reviews, and discussions of different aspects of various temporal graph and network modeling methods are available [167–170].

Based on presented evolution, the terms *slot*, *space graph*, and *space-time graph*, as well as, e.g., *space edge* and *time edge*, will be used as the basis for naming modeled networks and their elements. Some graphs will be additionally called *time-expanded* or *time-aggregated* to indicate if their form is layered or compacted.

4.3.1 Network devices data to slots of space nodes

The first stage of modeling, as presented in Algorithm 1, *network devices data to slots of space nodes* (NDD-SSN), is aimed at data quantization, i.e., the construction of the list of subsequent time *slots*. The term comes from the research of Huang et al. where it was used to denote the space between consecutive layers of a *space-time graph* [161]. In presented novel network modeling approach, these *slots* are network topology snapshots that capture the deployment of modeled physical wireless network *devices* in consecutive intervals of $slotLength$. Each time *slot* groups the *instances* (occurrences) of all *devices* in network *area* of interest considered to belong to given $timeFrame$. Such a time-frame is defined for every network device $class$.

The time-distribution of data related to the devices is discrete. The two-dimensional *area* is defined by two space-related closed intervals. As a result, each actual *device* gets to be represented as a *node* in the *slots* the *device* is considered to have occurred. The *slots* are sets of nodes only, i.e., the nodes are not connected with any edges yet, as presented in Figure 4.7. Stationary simple nodes are depicted with green circles, stationary advanced are depicted with blue triangles, and mobile advanced are blue triangles additionally marked with black border.

Time topology begins at given *topologyStart* time and its span, i.e., *topologyLength* is defined by the number of subsequent time *slots* of equal non-zero *slotLength* within network modeling period. Each *device* in the *area* of interest is identified by a unique *id*, performs desired network

Algorithm 1: Network devices data to slots of space nodes

```
Input:
$area \leftarrow ([X_{min}, X_{max}], [Y_{min}, Y_{max}])$    // network area of interest
$classes \leftarrow (class_i)_{i \leftarrow 1}^{j}$ :    // list of $j$ distinct device classes
  $class_i \leftarrow \{device_k\}_{k \leftarrow 1}^{l}$ :    // set of $l$ devices of class $i$
    $device_k, \leftarrow ($    // device $k$
      $instances_k, \leftarrow (instance_m)_{m \leftarrow 1}^{n}$ :    // list of $n$ time instances of device $k$
        $instance_m \leftarrow ($    // instance $m$
          $time_m,$    // instance occurrence time
          $location_m$    // instance location
        $),$
      $id_k,$    // device id
      $role_k$    // device role
    $)$
$slotLength \in \mathbb{R}_{>0}$    // length of time slots
$topologyLength \in \mathbb{N}^+$    // number of subsequent time slots of the time topology
$topologyStart$    // start time of the topology
$windows \leftarrow (window_i)_{i \leftarrow 1}^{j}$    // list of $j$ time window lengths corresponding to the respective $j$ device classes
Output:
$slots \leftarrow (slot_p)_{p \leftarrow 1}^{topologyLength}$ :    // list of subsequent time slots
  $slot_p \leftarrow \{node_r\}_{r \leftarrow 1}^{s}$ :    // set of all $s$ nodes of slot $p$
    $node_r \leftarrow ($    // node $r$ of slot $p$
      $class_r,$    // node class
      $id_r,$    // node id
      $location_r,$    // node geographic location
      $role_r,$    // node role
      $slotNumber_r$    // node slot number
    $)$
1 $slots \leftarrow assignDevicesInstancesToSlots()$    // assign devices instances to slots of nodes
2 output $slots$
```

role and belongs to one of distinct *classes*. A data-lookup *window* of given length, i.e. duration, is defined for each such *class*. Every time occurrence of a *device* is distinguished and considered a time *instance* of this *device*. Every *instance* is marked with occurrence *time* and *location*.

Procedure *assignDevicesInstancesToSlots* starts with initializing the list of empty time *slots*, setting *topologyStart* as initial *slotEnd* time and getting the number of device classes j. Then it iteratively defines the sets of nodes that belong to each *slot*. Every iteration begins with moving current value of *slotEnd* time by *slotLength*. Then, for each *class*, it is being checked which time *instance* of every *device* in network *area* is the newest time-occurrence of that *device* in the *timeFrame* of interest. This *timeFrame* is defined as half-open interval preceding *slotEnd* by time length of the *window* of the *class* of the *device*. If *newestInstance* gets selected, i.e., an *instance* which satisfies time conditions related to current *slot* and *class*, a new *node* is added to current *slot*. That *node* is marked with *class*, *id*, *location*, *slotNumber* and *role* of the *instance* of given *device*. As a result, the list of *slots* of nodes is obtained. It has to be pointed out that a list is an ordered sequence of elements while a set is an unordered collection.

Due to nested iterative nature of NDD-SSN algorithm, its upper bound of time complexity is related to the number of *slots* (topology length), device *classes*, the number of the devices in the

```
Procedure assignDevicesInstancesToSlots
  Input: defined in Algorithm 1
  Output:
  $slots$                                           // list of subsequent time slots of nodes
1 $slots \leftarrow (slot_p)_{p \leftarrow 1}^{topologyLength} \leftarrow \varnothing$      // initialize the list of empty time-slots
2 $slotEnd \leftarrow topologyStart$                // set initial slot end to topology start time
3 $j \leftarrow |classes|$                          // get the number (cardinality) of device classes
  // select and assign network devices to each slot as nodes
4 foreach $slot \in slots$ do
5     $slotEnd \ + \leftarrow \ slotLength$         // set end of current time-slot
      // process each device class
6     for $i \leftarrow 1$ to $j$ do
          // process each device of the class
7         foreach $device \in classes[i]$ do
8             $newestInstance \leftarrow \varnothing$     // set device newest instance to an empty set
9             $newestOccurrenceTime \leftarrow -\infty$   // set device newest ocurrence time to minus infinity
              // find the newest time-occurrence of the device in the area of interest
10            foreach $instance \in device.instances$ do
11                if $instance.location \in area$ then
12                    $timeFrame \leftarrow [slotEnd - windows[i], slotEnd)$   // set time-frame of interest
13                    if $instance.time \in timeFrame$ then
14                        if $instance.time > newestOccurrenceTime$ then
15                            $newestInstance \leftarrow instance$
16                            $newestOccurrenceTime \leftarrow instance.time$
              // if the newest instance of a distinct device was selected, add it as a node to the list
                 of slot nodes which occurred in the area of interest, as well as, within class window of
                 current time slot
17            if $newestInstance \neq \varnothing$ then
18                $slot \ \cup \leftarrow \ newestInstance$   // add the node to the current time-slot
19 output $slots$
```

largest *class*, as well as, the maximum number of device *instances*. Therefore it can be defined as $\mathcal{O}(|slots| \cdot |classes| \cdot |class|_{MAX} \cdot |instances|_{MAX})$.

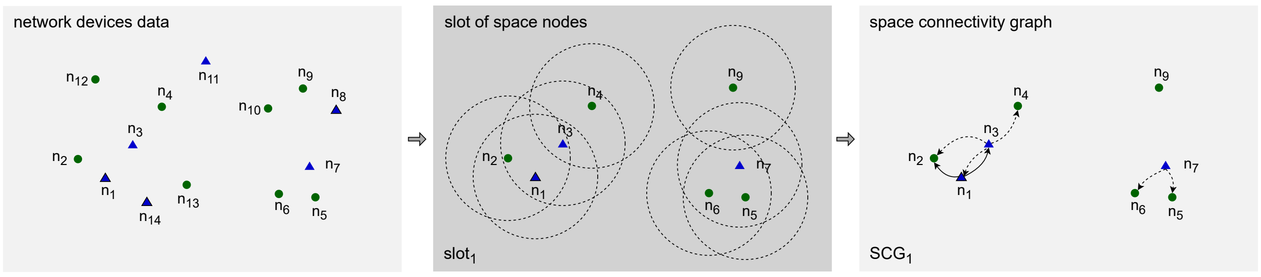


Figure 4.7 Space connectivity graph modeling example

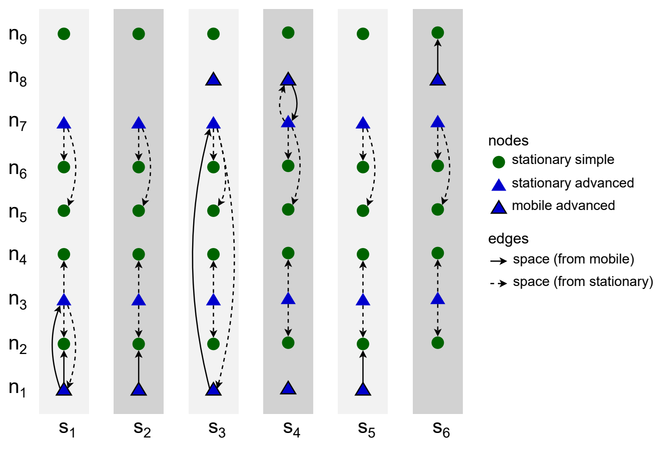


Figure 4.8 Exemplary space connectivity list

4.3.2 Slots of space nodes to space connectivity list

When the data related to network devices got turned into subsequent time slots of nodes, the next step of modeling network topology can take place. Therefore, Algorithm 2, *slots of space nodes to space connectivity list* (SSN-SCL), constructs *space connectivity list* (SCL), i.e., the list of subsequent time-ordered directed *space connectivity graphs* (SCG)—based on the list of *slots* and assumed *radioCoverage* of the devices. Such a SCL was called an *evolving graph* by Ferreira [158]. The graphs are considered to reflect possible temporary connectivity of the network in the related *slot*, as shown in Figure 4.7. Dashed links are the ones originating in stationary nodes, while solid are the links starting in mobile nodes. Here, *radioCoverage* can be a simple non-zero omnidirectional constant-range function, as well as, a an advanced model-based function which depends on, e.g., device class, state, radio transmission and reception capabilities, as well as, propagation conditions. An exemplary SCL of 6 *slots* and 9 *nodes* is presented in Figure 4.8. It is assumed that nodes of simple type can only be the ends of the edges (receivers), while advanced nodes can be also the start (transmitters). The number of nodes varies among graphs and reflects the changes of the number of space nodes int the area of interest over time.

The *boundingRegion* is used to reduce the number of inter-nodes distance computations. It is based on the observation that for complex *radioCoverage* functions it is beneficial to perform the first rough neighbor *candidate* filtering in a less compute-intensive way. Afterwards, further precise calculations are performed only for the pairs of nodes that are close enough, and therefore, likely to be able to establish a connection—depending on the shape, size and center (location) of *radioCoverage* of those nodes. The simplest approach to determine it, when working with a spheroid-based coordinate system, is to find a quasi-rectangular projected circumscribed area of *radioCoverage* of the *node*. In the most simplistic case of flat network area and uniform omnidirectional *radioCoverage*, *boundingRegion* would be a circumscribed rectangle of a circle centerd at the location of the node with radio range being the radius of this circle.

The algorithm begins by establishing the number j of time slots of nodes, i.e., getting the cardinality of the list of *slots* and initializes the list of j empty SCG graphs. Then, iteratively, each SCG_i graph is filled with edges SCE_i and vertices SCV_i. This list of l vertices is simply the list of all nodes of $slot_i$. The edges SCE_i are determined in more complex, and yet computational complexity optimized way.

For each SCG_i an empty set of edges SCE_i gets initialized. Then, the set of vertices (nodes) SCV_i is traversed. In every iteration, each not yet traversed vertex of SCV_i is iteratively considered neighbor *candidate*. By this means, the number of computations can be limited. In other words, the candidates are the $l - k$ nodes which succeed current *node* (the k^{th} one) in the list of l nodes of SCV_i. This optimization can be applied because of symmetric nature of the process and operations aimed both at *node* and *candidate* can follow.

If *candidate* is located within *boundingRegion* of *node*, then it is being checked if *node* is placed within *radioCoverage* of *candidate*, as well as, *candidate* wihin *radioCoverage* of *node*. When both conditions are met, the *distance* between *node* and *candidate* is calculated. The method *edgeDistance* can be a simple computation of geographical distance between two nodes by the means of determining great-circle distance on a sphere [171]. It can be also a more complex metric function, e.g., related to the minimum power needed complete single transmission to the neighboring node. Next, if *node* is a relay, then E_i is extended with an edge from *node* to neighbor *candidate*. Similarly, an edge from neighbor *candidate* to *node* is added to SCE_i if *candidate* is a relay. When all iterations are completed, SCE_i and SCV_i of SCG_i are stored in SCL.

The radio connectivity related functions *radioCoverage* and *boundingRegion* are advised to be either precomputed for each *node* of SCV_i at the beginning of iteration i or computed at the first usage and stored (cached) for future use, depending on the implementation. In general urban use-case, it can be assumed that *radioCoverage* and network area dimensions would be of different orders. Therefore, not using *boundingRegion*, especially with large number of highly dispersed nodes would lead to significant increase of computational complexity. In cases where *radioCoverage* and dimensions of network area tend to be of the same or close order, especially in sparse networks, it might be beneficial to omit the computation and usage of *boundingRegion*. Similarly, the use of *boundingRegion* may be counter-effective if it would be of similar or higher computational complexity than checking if a node belongs to a region defined by *radioCoverage*.

The upper bound of the complexity of SSN-SCL algorithm is $\mathcal{O}(|slots| \cdot |SCV|_{MAX}{}^2)$, i.e., is related to the number of *slots* (topology length) and the number of nodes in a graph with the largest number of nodes of all graphs of the topology.

4.3.3 Space connectivity list to space-time connectivity graph

The construction of *space-time connectivity graph* (STCG) is defined as more complex Algorithm 3, *space connectivity list to space-time connectivity graph* (SCL-STCG), composed of several procedures. Based on the structure of each directed space graph SCG_i of SCL, time node instances, as well as, space-time edges are added to $STCG$. These edges are of two types, i.e., *space*, and *time*, which indicates their role in the structure. Space edges connect different neighboring space nodes (devices) while time edges connect consecutive time instances of the same space node. Moreover, the notion of *intra-slot* and *inter-slot* edges is introduced to distinguish their graph roles. Intra-slot edges, that can be of *space* and *time* types, are used to construct the graph structure related to sptiotemporal relations within a given slot of nodes (based on SCG_i). Inter-slot edges are of *time* type and connect

Algorithm 2: Slots of space nodes to space connectivity list

```
Input:
$slots \leftarrow (slot_i)_{i\leftarrow1}^{j}$ :                // list of $j$ subsequent time slots
    $slot_i \leftarrow \{node_k\}_{k\leftarrow1}^{l}$            // set of $l$ nodes of slot $i$
$radioCoverage$                                // radio coverage function of the devices (nodes)
Output:
$SCL \leftarrow (SCG_i)_{i\leftarrow1}^{j}$ :        // list of $j$ subsequent directed space connectivity graphs
    $SCG_i \leftarrow ($                               // space connectivity graph $i$
        $SCE_i \leftarrow \{edge_m\}_{m\leftarrow1}^{n}$ :         // set of $n$ edges (arcs) of graph $i$
            $edge_m \leftarrow ($              // directed edge $m$ from $start$ to $end$ node
                $start_m$,                              // start node
                $end_m$,                                // end node
                $distance_m$,                           // geographical distance
                $slotNumber_m$                          // slot number
            ),
        $SCV_i \leftarrow \{node_k\}_{k\leftarrow1}^{l}$          // set of $l$ vertices (nodes) of graph $i$
    )

1  $j \leftarrow |slots|$                   // get the number (cardinality) of time slots
2  $SCL \leftarrow (SCG_i)_{i\leftarrow1}^{j} \leftarrow \varnothing$        // initialize $SCL$ as the list of $j$ empty graphs
   // convert each slot to space connectivity graph
3  for $i \leftarrow 1$ to $j$ do
4      $SCE_i \leftarrow \varnothing$                       // initialize an empty set of edges
5      $SCV_i \leftarrow slots[i]$                 // initialize the set of vertices (nodes)
6      $l \leftarrow |SCV_i|$                       // get the number of nodes of $SCV_i$
       // look for neighbors of each node of $SCV_i$
7      for $k \leftarrow 1$ to $l$ do
8          $node \leftarrow SCV_i[k]$                        // get node $k$ of $SCV_i$
           // look for neighbors in remaining $l-k$ nodes of $SCV_i$
9          for $p \leftarrow k+1$ to $l$ do
10             $candidate \leftarrow SCV_i[p]$     // get node $p$ of $SCV_i$ as neighbor candidate
               // check if candidate is within radio coverage bounding region of the node
11             if $candidate \in boundingRegion(node, radioCoverage)$ then
                   // check if radio connection could be established
12                 if $node.location \in radioCoverage(candidate)$ and $candidate.location \in$
                    $radioCoverage(node)$ then
                       // calculate distance between node and neighbor candidate
13                     $distance \leftarrow edgeDistance(node, candidate)$
                       // connect transmitting node with receiving node
14                     if $node.role =$ "relay" then
15                         $SCE_i \cup\leftarrow (node, candidate, distance, i)$   // extend $SCE_i$ with an edge from
                            node to neighbor candidate
16                     if $candidate.role =$ "relay" then
17                         $SCE_i \cup\leftarrow (candidate, node, distance, i)$   // extend $SCE_i$ with an edge from
                            neighbor candidate to node
18     $SCL[i] \leftarrow (SCE_i, SCV_i)$          // store edges and vertices of graph $SCG_i$ of $SCL$
19 output $SCL$
```

slot-related structures (stages) to produce a space-time connectivity graph. *STCG* is therefore a one-way multipath multistage layered structure that follows the direction of time and is a directed acyclic graph, as depicted in Figure 4.9.

To enable the usage of various algorithms, such as the well-known ones related to finding shortest paths or trees, to process constructed *STCG* as input structure, the edges share the same set of attribute types, e.g., *spaceDistance*, and *timeDistance*. These weights are computed or set based on defined unit costs. They are a means of modifying or tweaking the resulting cost structure to meet the needs of further modeling and analysis:

- *intraSlotTimeEdgeSpaceUnitCost*—space unit cost of a time edge within a slot:
  - default value: 0;
  - meaning: nonzero value stands for unit space cost related to the node (device) operating within a time slot. It can be used, for example, to model the cost of receiving location beacons;
- *interSlotTimeEdgeSpaceUnitCost*—space unit cost of a time edge between slots:
  - default value: 0;
  - meaning: nonzero value stands for unit space cost related to the node (device) transitioning between time slots. It can be used, for example, to model the cost of transmitting location beacons;
- *intraSlotSpaceEdgeTimeUnitCost*—time unit cost of a space edge within a slot:
  - default value: 0;
  - meaning: nonzero value indicates unit time cost related to transmitting a message between two devices, e.g., due to technology-dependent buffering or delays. It can be used, for instance, together with *intraSlotSpaceEdgeTimeUnitCost* to favor intra-slot time edges over intra-slot space edges by path finding algorithms. It will lead to maximizing buffering time in a single relay, minimizing the number of inter-node transmissions, and hence, the nodes involved. Although, it can happen at the expense of overall increase of space cost of constructed *STCG*;
- *intraSlotTimeEdgeTimeUnitCost*—time unit cost of a time edge within a slot:
  - default value: 0;
  - meaning: should be considered in relation to *intraSlotSpaceEdgeTimeUnitCost* for given modeling scenario. It can also be used with *interSlotTimeEdgeTimeUnitCost* to shape time-path cost properties of *STCG*, e.g., as a tie-breaker;
- *interSlotTimeEdgeTimeUnitCost*—time unit cost of a time edge between slots:
  - default value: 1;
  - meaning: indicates unit cost related to transitioning (buffering) a message over time by a node. It is of key significance for path searching scenarios that aim to optimize message delivery time, e.g., to minimize the total time cost of a path. Set to 0 may lead to unexpected or erroneous results in optimization algorithms which are based on ordering the weight of the edges. It can be of use, though, when consciously used with properly selected values of *intraSlotSpaceEdgeTimeUnitCost* and *intraSlotTimeEdgeTimeUnitCost*.

Iterating over *SCL*, Procedure *addTimeNodeInstancesAndEdges* is invoked twice for SCG_1. The first, i.e., additional call, extends the sets of *space-time connectivity edges* (STCE) and *space-time connectivity vertices* (STCV) with time node instances and edges which represent non-existent graph zero, as done by Huang et al. [161]. Such an abstract graph SCG_0 of no space edges is required to provide correct starting points for path- and tree-finding algorithms and enable traversals based on space and time metrics of the graph. For the remaining SCG_i graphs, both space and time edges are constructed.

To add time node instances and edges in Procedure *addTimeNodeInstancesAndEdges*, each node of given *graph* is used to make two new *STCV* nodes which represent the instances of the node at the "*start*" and "*end*" of time slot i. The creation of these nodes is defined in Procedure *makeTimeNodeInstance*. To initialize new *timeNodeInstance*, first the attributes of *spaceNodeInstance* are copied. Then, *id* of the space node is stored as *globalId* to keep the reference of time node instance to its parent space node. Next, new *id* is composed, i.e., the *id* of the space node gets prefixed with slot number i and time instance type indicator, either "*_s_*" for slot start, or "*_e_*" for slot end instance. In this way, for example, node "*1357*" of slot (space connectivity graph) number 3 will be converted to slot start instance "*3_s_1357*".

Afterwards, following Procedure *addTimeEdges*, *intraSlotTimeEdge* and *interSlotTimeEdge* are added to *STCE*. The directions are defined by *start* and *end* node attributes and additional labels are set, i.e., *spaceDistance*, *timeDistance*, *slotNumber*, and *type* set to "*time*". Here, *timeDistance* can mean, for example, the delay or buffering time related to traversing the edge by a message. Current slot node instances are connected with a *time* edge of *intraSlotTimeEdgeSpaceUnitCost* and *intraSlotTimeEdgeTimeUnitCost*. Current slot start instance is then linked with the newest slot end instance that exists in the set of *globalIds* which is an attribute of *STCV*. This does not always mean it is getting connected with the end instance of the previous slot. Node instance might have not been present in the directly preceding slot or space node has not yet been present in the space-time graph. Then, the slot end instance is added to the list of *endInstances* of *globalId* in *globalIDs* set related to space nodes of *STCV*. Next, in Procedure *addSpaceEdges*, each space edge of SCG_i is converted to *intraSlotSpaceEdge* and added to *STCE*. Finally, *STCE* and *STCV* are used to compose *STCG*.

The upper bound of time complexity of SCL-STCG algorithm is $\mathcal{O}(|SCL| \cdot |SCV|_{MAX} \cdot |SCE|_{MAX})$, and hence, is related to the number of space connectivity graphs, the number of nodes in the graph with the largest number of nodes, as well as, the number of edges in the graph with the largest number of edges of all graphs of the topology.

Space-time connectivity graph (STCG) is the extension of existing layered *space-time graph* (STG) concept [160, 161]. It unambiguously reflects space and time dimensions of changing network topology, and hence, enables multi-criteria spatiotemporal design and analysis. The essential innovations are the presented duplication of space nodes for each time slot as *start* and *end* node instances, as well as, the introduction of intra-slot and inter-slot edges and metrics (e.g., the sets of s_{1s} and s_{1e} nodes and edges in Figure 4.9). They enable the development of new optimization algorithms and the proper usage of existing effective path-finding ones designed for graphs of more traditional time flow ignoring contexts, i.e., static as compared to dynamic (evolving) graphs. It is worth to note the term *time-expanded graph* which was used in related context [162]. In spite of that structure being an even more simplistic model, the term itself can be additionally of use in relation to space-time graphs because it captures and highlights the time-related graph structure span.

In [162] also the model of *time-aggregated graph* is presented. It uses single instances of each node and edges between them. The edges are labeled with occurrence times of each connection.

This alternative representation of spatiotemporal graph defined as space connectivity list visible in Figure 4.8 is presented in Figure 4.12a. There, directed edge label $s_{1,2,5}$ means that the link originated by a mobile node existed in time slots 1, 2, and 5. Similarly, s_{1-6} denotes that the connection between stationary nodes were present throughout the whole time span of modeled network.

Time-aggregated representation is not used to model STCG because it does not enable direct use of well-known graph optimization and analysis algorithms which provide optimal solutions. There are being developed, though, methods aimed at solving these problems in *time-aggregated graphs*. The problem of determining minimum temporal paths is addressed by the algorithms for finding earliest-arrival, latest-departure, fastest, and shortest paths [165]. Methods for constructing *directed Steiner tree* (DST) in a structure that resembles a *space-time graph* transformed from *time-aggregated graph* called *temporal graph* are also presented [172]. Similarly, DST is aimed to be constructed directly in *space-time graph* used in topology control efforts presented in [161].

Importantly, time-aggregated graph is a representation well-fitted to capture the outcomes of algorithms that solve problems in STCGs. Therefore, it is used in the present research as practical representation of modeled first-contact and multicast graphs. Please see Figure 4.12b and Figure 4.12c for examples.

Algorithm 3: Space connectivity list to space-time connectivity graph

```
Input:
$SCL \leftarrow (SCG_i)_{i \leftarrow 1}^{j}$ :                 // list of $j$ subsequent directed space connectivity graphs
    $SCG_i \leftarrow ($                                                                  // directed graph $i$
        $SCE_i,$                                                                       // set of edges (arcs)
        $SCV_i$                                                                     // set of vertices (nodes)
    )
$intraSlotTimeEdgeSpaceUnitCost \in \mathbb{R}_{\geq 0}$       // space unit cost of a time edge within a slot
$interSlotTimeEdgeSpaceUnitCost \in \mathbb{R}_{\geq 0}$       // space unit cost of a time edge between slots
$intraSlotSpaceEdgeTimeUnitCost \in \mathbb{R}_{\geq 0}$       // time unit cost of a space edge within a slot
$intraSlotTimeEdgeTimeUnitCost \in \mathbb{R}_{\geq 0}$         // time unit cost of a time edge within a slot
$interSlotTimeEdgeTimeUnitCost \in \mathbb{R}_{\geq 0}$         // time unit cost of a time edge between slots
Output:
$STCG \leftarrow ($                                               // directed space-time connectivity graph
    $STCE,$                                                       // set of space-time connectivity edges
    $STCV$                                                      // set of space-time connectivity vertices
)

1  $STCE \leftarrow \varnothing$                              // initialize an empty set of space-time connectivity edges
2  $STCV \leftarrow \varnothing$                            // initialize an empty set of space-time connectivity vertices
3  $j \leftarrow |SCL|$                                     // get the number (cardinality) of graphs in $SCL$
   // process each graph
4  for $i \leftarrow 1$ to $j$ do
5      $SCG_i \leftarrow SCL[i]$                                                       // get a copy of graph $SCG_i$
       // add nodes and edges from first space connectivity graph also as graph zero
6      if $i = 1$ then
7          $addTimeNodeInstancesAndEdges(0, SCG_i, STCE, STCV)$
8      $addTimeNodeInstancesAndEdges(i, SCG_i, STCE, STCV)$
9      $addSpaceEdges(i, SCG_i, STCE)$
10 $STCG \leftarrow (STCE, STCV)$                                      // compose space-time connectivity graph
11 output $STCG$
```

Procedure addTimeNodeInstancesAndEdges(i, SCG, STCE, STCV)

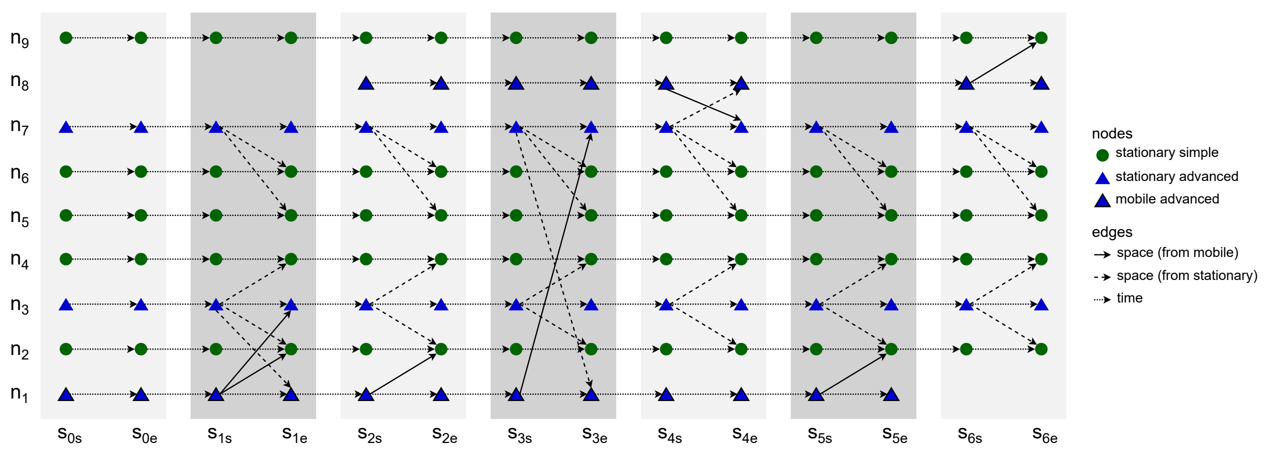
```
Input:
$i$                                         // space connectivity graph number
$SCG$                                       // space connectivity graph
$STCE$                                      // set of space-time connectivity edges (arcs)
$STCV$                                      // set of space-time connectivity vertices (nodes)

1  foreach $spaceNodeInstance \in SCG.SCV$ do
2      $timeNodeInstances \leftarrow$ (
3          $start \leftarrow makeTimeNodeInstance(i, spaceNodeInstance, “start”)$
4          $end \leftarrow makeTimeNodeInstance(i, spaceNodeInstance, “end”)$
5      )
6      $STCV \cup\leftarrow timeNodeInstances.start \cup timeNodeInstances.end$
7      $addTimeEdges(STCE, STCV, timeNodeInstances)$
       // store end time node instance in the list of end instances of current space node identified by
          globalId
8      $globalId \leftarrow spaceNodeInstance.id$
9      $STCV.globalIds[globalId].endInstances \cup\leftarrow timeNodeInstances.end$
```

Procedure makeTimeNodeInstance(i, spaceNodeInstance, timeInstanceType)

```
Input:
$i$                                         // space connectivity graph number
$spaceNodeInstance$                         // instance of a space node of graph $i$
$timeInstanceType$                          // type of time node instance to be created
Output:
$timeNodeInstance$                          // time instance of requested type

   // initialize time node instance and copy the attributes of space node instance
1  $timeNodeInstance \leftarrow spaceNodeInstance$
2  $timeNodeInstance.slotNumber \leftarrow i$                  // store slot (graph) number
   // store global (time-invariant) id of space node
3  $timeNodeInstance.globalId \leftarrow spaceNodeInstance.id$    // e.g. 1357
   // make time-prefixed id of space-time node instance based on its type
4  $timeNodeInstance.id \leftarrow i$                           // e.g. 3
5  switch $timeInstanceType$ do
6      case “start” do
7          $timeNodeInstance.id \cup\leftarrow$ “_s_”               // e.g. 3_s_
       // destination type set to node classes
8      case “end” do
9          $timeNodeInstance.id \cup\leftarrow$ “_e_”
10 $timeNodeInstance.id \cup\leftarrow spaceNodeInstance.id$       // e.g. 3_s_1357
11 output $timeNodeInstance$
```

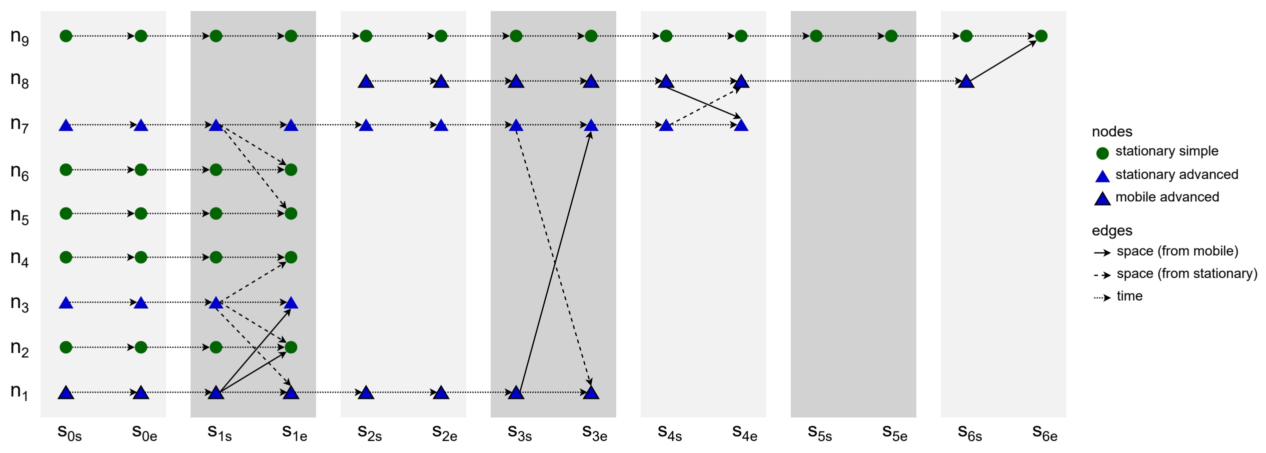
```
Procedure addTimeEdges(STCE, STCV, timeNodeInstances)
  Input:
  $STCV$                    // set of space-time vertices connectivity (nodes)
  $STCE$                    // set of space-time edges connectivity (arcs)
  $timeNodeInstances$       // time instances of a space node

  // define space-time edge from current time node slot start to end instance
1 $intraSlotTimeEdge \leftarrow ($
2     $start \leftarrow timeNodeInstances.start,$
3     $end \leftarrow timeNodeInstances.end,$
4     $spaceDistance \leftarrow intraSlotTimeEdgeSpaceUnitCost,$
5     $timeDistance \leftarrow intraSlotTimeEdgeTimeUnitCost,$
6     $slotNumber \leftarrow timeNodeInstances.end.slotNumber,$
7     $type \leftarrow$ "time"
8 $)$
9 $interSlotTimeEdge \leftarrow \varnothing$        // initialize an empty inter-slot time edge
10 $globalId \leftarrow timeNodeInstances.start.globalId$
  // check if an instance of current node already exists in $STCV$
11 if $globalId \in STCV.globalIds$ then
      // get previous time instance of current node
12    $previousEndNodeInstance \leftarrow newest(STCV.globalIds[globalId].endInstances)$
      // define space-time edge from previous slot end instance to current start instance of the node
13    $interSlotTimeEdge \leftarrow ($
14        $start \leftarrow previousEndNodeInstance,$
15        $end \leftarrow timeNodeInstances.start,$
16        $spaceDistance \leftarrow interSlotTimeEdgeSpaceUnitCost,$
17        $timeDistance \leftarrow interSlotTimeEdgeTimeUnitCost \cdot$
           $(timeNodeInstances.start.slotNumber - previousEndNodeInstance.slotNumber),$
18        $slotNumber \leftarrow timeNodeInstances.start.slotNumber,$
19        $type \leftarrow$ "time"
20    $)$
21 $STCE \cup\leftarrow intraSlotTimeEdge \cup interSlotTimeEdge$        // add time edges to $STCE$
```

```
Procedure addSpaceEdges(i, SCG, STCE)
  Input:
  $i$             // space connectivity graph number
  $SCG$           // space connectivity graph
  $STCE$          // set of space-time connectivity edges

1 $j \leftarrow |SCG.SCE|$        // get the number (cardinality) of edges in $SCE$
  // process each space connectivity edge of $SCG$
2 for $i \leftarrow 1$ to $j$ do
3     $spaceEdge \leftarrow SCG.SCE_i$        // get a copy of edge $i$ of $SCG.SCE$
      // define space-time edge based on space edge
4     $intraSlotSpaceEdge \leftarrow ($
5         $start \leftarrow i \cup$ "_s_" $\cup spaceEdge.start,$
6         $end \leftarrow i \cup$ "_e_" $\cup spaceEdge.end,$
7         $spaceDistance \leftarrow spaceEdge.distance,$
8         $timeDistance \leftarrow intraSlotSpaceEdgeTimeUnitCost,$
9         $slotNumber \leftarrow i,$
10        $type \leftarrow$ "space"
11    $)$
12    $STCE \cup\leftarrow intraSlotSpaceEdge$        // add space edge to $STCE$
```



n_9
n_8
n_7
n_6
n_5
n_4
n_3
n_2
n_1
s_{0s} s_{0e} s_{1s} s_{1e} s_{2s} s_{2e} s_{3s} s_{3e} s_{4s} s_{4e} s_{5s} s_{5e} s_{6s} s_{6e}
nodes
● stationary simple
▲ stationary advanced
▲ mobile advanced
edges
→ space (from mobile)
⇢ space (from stationary)
⋯> time

Figure 4.9 Exemplary time-expanded space-time connectivity graph



n_9
n_8
n_7
n_6
n_5
n_4
n_3
n_2
n_1
s_{0s} s_{0e} s_{1s} s_{1e} s_{2s} s_{2e} s_{3s} s_{3e} s_{4s} s_{4e} s_{5s} s_{5e} s_{6s} s_{6e}
nodes
● stationary simple
▲ stationary advanced
▲ mobile advanced
edges
→ space (from mobile)
⇢ space (from stationary)
⋯> time

Figure 4.10 Exemplary time-expanded first-contact graph

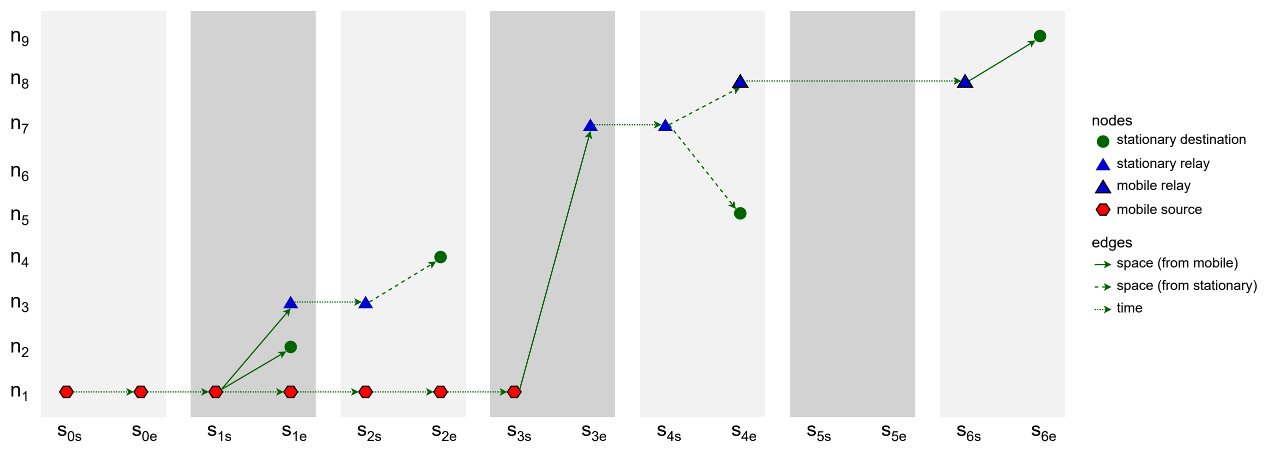
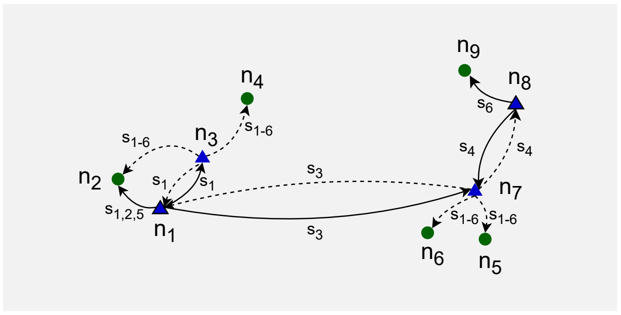
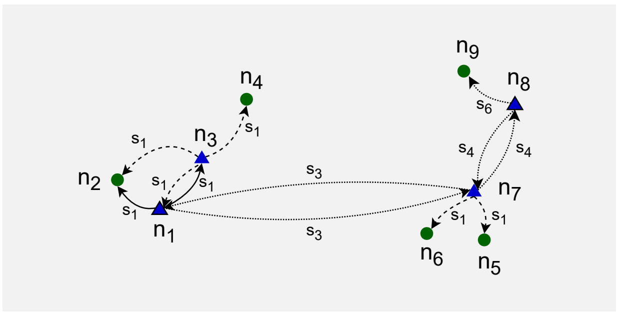


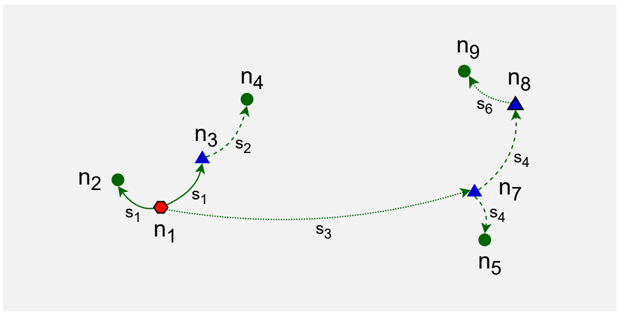
Figure 4.11 Exemplary time-expanded space-time multicast graph



(a) Space-time connectivity graph



(b) First-contact graph



(c) Multicast graph

Figure 4.12 Exemplary time-aggregated graphs

4.3.4 Space-time connectivity graph to first-contact graph

To build *first-contact graph* (FCG), Algorithm 4, *space-time connectivity graph to first-contact graph* (STCG-FCG), is used. Such a graph is a *time-aggregated graph* with single instances of all *spaceNodes* located at the coordinates of their first instances, i.e., the ones in $firstSlot$ (first SCG) in which the node was present. Each node is connected to each neighbor with a directed first-contact space or time edge. Space edges connect the nodes that are neighbors in the same $firstSlot$. Time edges connect them otherwise.

An illustrative time-expanded FCG presented in Figure 4.10 was constructed in $STCG$ introduced in Figure 4.9. The related time-aggregated form is depicted in Figure 4.12b. Presented edge labels, for instance, s_3 and s_4, indicate in which time slot of given *slotNumber* the $firstContactEdge$ existed between two nodes. Similarly, an exemplary space-time multicast graph was constructed and presented in Figures 4.11 and 4.12c. It is a time-respecting tree that connects, over time, mobile source node n_1, via intermediate relay nodes, with four stationary destination nodes n_2, n_4, n_5, and n_9.

The STCG-FCG algorithm starts with finding first instances of space nodes and their first time neighbors in $STCG$. Procedure *findFirstInstancesAndTimeNeighbors* iterates over each $nodeInstance$ of $STCG$. At the beginning it adds new space node to the set of $spaceNodes$ if it does not contain a node indexed with $globalId$ of current $nodeInstance$. Key attributes of new space node are inherited from current $nodeInstance$, i.e., $globalId$ becomes id, id is set as $firstInstance$ and $slotNumber$ becomes $firstSlot$. Moreover, an empty set of edges to successors is initialized. If a node indexed with $globalId$ was present in $spaceNodes$, then $firstInstance$ and $firstSlot$ get updated if $slotNumber$

of current $nodeInstance$ is lower than $firstSlot$ currently stored in $spaceNodes$ for current $globalId$ of interest. This means, that current $nodeInstance$ precedes the instance which has been so far considered the $firstInstance$ (occurrence) of given $globalId$ (space node). The procedure closes with adding successors of current $nodeInstance$ which are instances of different space nodes. Its complexity is related to the number of nodes of $STCG$, and hence, upper bounded by $\mathcal{O}(|STCV|^2)$.

When the set of $spaceNodes$ is ready, Procedure *buildFirstContactGraph* can be used to construct FCG out of $STCG$. At the beginning, all first node instances need to be added to FCG. Therefore, for each $spaceNode$ a $firstContactNode$ is obtained. Such a node is the space node of $STCG$ which was determined to be the $firstInstance$ of given $spaceNode$. Its *id* is set to the *id* of $spaceNode$, and then, $firstContactNode$ can be added to the set of vertices FCV of FCG. In the next step, edges going out from each $firstContactNode$ are added to FCG. For each *node* of FCG, each *edge* going out from current *node* is being evaluated. The end node of the edge is set to be current *neighbor* of the *node*. Then, *timeDistance* between them is calculated and *contactEdge* defined as any connecting *node* to *neighbor*, based on their *id* and $globalId$, respectively. If such an edge does not exist in FCG, then *edgeType* is set. It is "*time*" if *timeDistance* is positive and "*space*" otherwise. Then, $firstContactEdge$ is defined and added to FCG. If *contactEdge* exists in FCG and its *timeDistance* is larger than *timeDistance* to current *neighbor* instance, then *timeDistance* of the edge of FCG is updated to current shorter *timeDistance*.

As a result, FCG contains first instances of nodes labeled with space *id*. Those nodes are connected to their neighbors with first contact edges. The upper bound of this procedure's time complexity is $\mathcal{O}(|spaceNodes|^2)$.

Algorithm 4: Space-time connectivity graph to first-contact graph

```
Input:
$STCG \leftarrow ($                                  // directed space-time connectivity graph
    $STCE,$                                          // set of space-time connectivity edges
    $STCV$                                           // set of space-time connectivity vertices
$)$
Output:
$FCG \leftarrow ($                                   // directed first-contact graph
    $FCE,$                                           // set of first-contact edges
    $FCV$                                            // set of first-contact vertices
$)$
1  $spaceNodes \leftarrow findFirstInstancesAndTimeNeighbors(STCG)$
2  $FCG \leftarrow buildFirstContactGraph(STCG.STCV, spaceNodes)$
3  output $FCG$
```

Procedure findFirstInstancesAndTimeNeighbors(STCG)

```
Input:
$STCG$                                               // directed space-time connectivity graph
Output:
$spaceNodes$                                         // set of space nodes
1  $spaceNodes \leftarrow \varnothing$               // initialize an empty set of space nodes
2  foreach $nodeInstance \in STCG.STCV$ do
3      $globalId \leftarrow nodeInstance.globalId$
       // add first instance of space node if not already in the set
4      if $globalId \notin spaceNodes$ then
5          $spaceNodes[globalId] \leftarrow ($
6              $id \leftarrow globalId,$
7              $firstInstance \leftarrow nodeInstance.id,$
8              $firstSlot \leftarrow nodeInstance.slotNumber,$
9              $edges \leftarrow \varnothing$        // initialize an empty set of edges to successors of this space node
10         $)$
       // set this node instance as first if it precedes current first instance
11     else if $nodeInstance.slotNumber < spaceNodes[globalId].firstSlot$ then
12         $spaceNodes[globalId].firstInstance \leftarrow nodeInstance.id$
13         $spaceNodes[globalId].firstSlot \leftarrow nodeInstance.slotNumber$
       // add successors which are instances of different space nodes
14     foreach $successor \in getSuccessors(nodeInstance, STCG)$ do
15         if $successor.globalId \neq nodeInstance.globalId$ then
16             $spaceNodes[globalId].edges \cup\leftarrow (nodeInstance.id, successor.id)$
17 output $spaceNodes$
```

Procedure buildFirstContactGraph(STCV, spaceNodes)

```
Input:
$STCV$,                                                // space-time connectivity graph vertices
$spaceNodes$                                           // set of space nodes
Output:
$FCG \leftarrow ($                                     // directed first-contact graph
    $FCE$,                                             // set of first-contact edges
    $FCV$                                              // set of first-contact vertices
)
1  $FCE \leftarrow \varnothing$                        // initialize an empty first-contact edges
2  $FCV \leftarrow \varnothing$                        // initialize an empty first-contact vertices
   // add all first node instances to the FCV
3  foreach $spaceNode \in spaceNodes$ do
       // get first instance of current space node
4      $firstInstance \leftarrow spaceNode.firstInstance$
       // copy first instance and its attributes as first contact node
5      $firstContactNode \leftarrow STCV[firstInstance]$
       // set id of first contact node to id of space node
6      $firstContactNode.id \leftarrow spaceNode.id$
       // add first contact node to first contact graph
7      $FCV \cup\leftarrow firstContactNode$
   // add edges going out from each first contact node
8  foreach $node \in FCV$ do
9      foreach $edge \in spaceNodes[node].edges$ do
10         $neighbor \leftarrow STCV[edge.end]$
11         $timeDistance \leftarrow neighbor.slotNumber - node.firstSlot$
12         $contactEdge \leftarrow (node.id, neighbor.globalId)$
           // check if contact edge is not already in FCE
13         if $contactEdge \notin FCE$ then
14             if $timeDistance > 0$ then
15                 $edgeType \leftarrow$ "time"
16             else
17                 $edgeType \leftarrow$ "space"
               // define first-contact edge
18             $firstContactEdge \leftarrow ($
19                 $start \leftarrow node.id$,
20                 $end \leftarrow neighbor.globalId$,
21                 $spaceDistance \leftarrow edge.spaceDistance$,
22                 $timeDistance \leftarrow edge.timeDistance$,
23                 $slotNumber \leftarrow edge.slotNumber$,
24                 $type \leftarrow edgeType$
25             )
26             $FCE \cup\leftarrow firstContactEdge$          // add first-contact edge to FCE
           // update time distance if current contact edge is of shorter time distance
27         else if $FCE[contactEdge].timeDistance > timeDistance$ then
28             $FCE[contactEdge].timeDistance \leftarrow timeDistance$
29 $FCG \leftarrow (FCE, FCV)$                        // compose first-contact graph
30 output $FCG$
```

4.4 Network modeling proof of concept

From numerous sources of open data on the geographic location of different types of urban nodes overviewed and analyzed in Section 4.1, those related to Poznań, Poland metropolitan area were selected to be the input for the illustrative network modeling presented in this section. They include the ones from Airly (air quality meters [117]), Smart City Poznań (public transport stops, parking meters, and ticket machines [121]), and Municipal Transport Authority in Poznań (real-time vehicle-related information [138]).

The modeled graphs and usage examples presented in the next subsections neither exhaust the scope of the research field nor provide universal and final solutions to the current routing-related research problems defined in Section 2.3. Their aim is to prove that the modeling architecture introduced in Section 4.2 and network modeling algorithms presented in Section 4.3 can be successfully used to study real-life networks to improve and develop new data transmission methods. In particular, it pertains to urban delay tolerant multicast algorithms introduced in Chapter 5.

4.4.1 Spatial graph modeling

A typical routing-related research study is based on monitoring and analysis of actual network traffic or a simulation in synthetic or semi-synthetic networks [173, 174], as well as on the analysis of static or infrequently changing graphs [175]. The goal of the first part of the modeling proof is to show that a static spatial graph (a nontemporal graph) depicting a wireless network of sensing capabilities can be modeled and analyzed in the center of Poznań, a city of 547 thousand inhabitants [176], during afternoon rush hours on Wednesday, 27 November 2019, at 3:15 p.m.

Space connectivity graph

A single static undirected radio connectivity graph, presented in Figure 4.13, was constructed by selecting gathered timestamped node location data that belonged to a single time interval (i.e., a time slot) and by determining assumed wireless links between those nodes (devices). This structure can be considered as the most basic graph representation of a momentary network topology. It consists of the green circle stationary (fixed) nodes (i.e., air quality meters, parking meters, public transportation stops, and ticket machines) and the blue triangle mobile nodes (i.e., buses and trams). The node coordinates (i.e., latitude and longitude) are defined following the WGS 84 notation. A link exists between two nodes if the geographical distance between them does not exceed the radio range. This distance is calculated using the haversine formula, which determines the great-circle distance between two points on a sphere [171]. The following modeling assumptions were made:

- Slot length: 6 s;
- Area dimensions: 3 km by 1.7 km;
- Area boundaries:
  - Latitude: 52.400 – 52.415;
  - Longitude: 16.898 – 16.942;

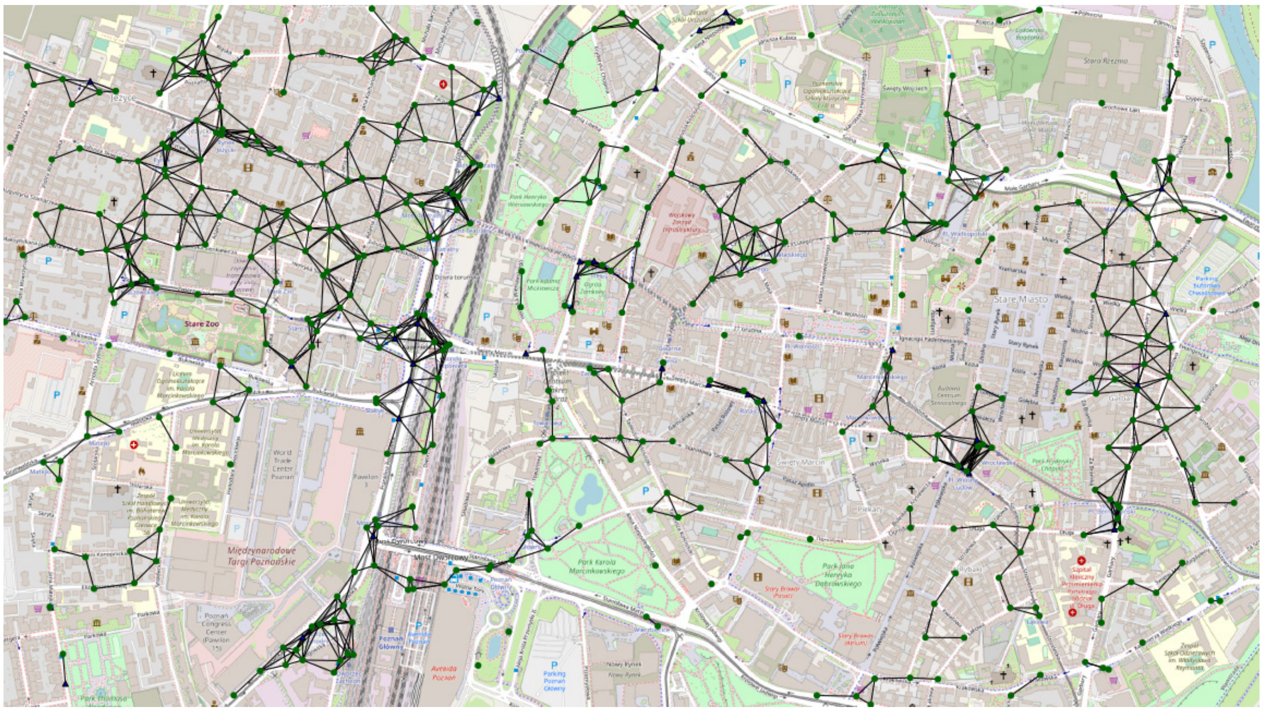


Figure 4.13 Modeled wireless connectivity graph

- Radio coverage: omnidirectional, radio range: 100 m;

A network modeled in this way, i.e., a network *snapshot*, captures the state of the assumed physical wireless network structure in a given period, i.e., the interval of duration meaningful for the analysis, and can be the basis for various studies related to routing research, in particular, in solving the open problems summarized in Section 2.3. The degree of a node is determined by the number of other nodes within the radio range of that node. The network is denser, i.e., the connected components (sub-graphs) consist of more nodes and edges, in areas with a higher density of the infrastructure elements (ticket machines, parking meters, vehicles, etc.) and more busy street routes. It is clearly visible that the shape and structure are largely related to the layout of the streets and the distribution of the supporting stationary infrastructure. The resulting space connectivity graph is characterized by the following metrics:

- Nodes: 501 — stationary nodes: 465, mobile nodes: 36;
- Average node degree: 4.02, edges: 1008, space cost: 67135 m, connected components: 59.

Space minimum spanning forest

In the next step, the minimum spanning forest (i.e., the set of minimum spanning trees determined for each connected component) of the momentary undirected graph in Figure 4.13 was determined using Kruskal's algorithm [177], with the edge weight being the geographical distance between the nodes expressed in meters. This is presented in Figure 4.14, this time without the background city map to place more emphasis on the graph itself and its components. This forest can be described with the following basic parameters:

- Nodes: 501 — stationary nodes: 465, mobile nodes: 36;
- Average node degree: 1.76, edges: 442, space cost: 23645 m, connected components: 59.

The proposed spatial modeling approach may be of use in numerous research fields. For instance, it might be related to the determination of the optimal number and placement of stationary data aggregation gateways (sinks) in a data dissemination and collection network, as discussed in Section 2.3.2. In this way, a tree connecting all disconnected components might be constructed to model and mimic the hierarchical network topology, with such special-purpose nodes located in the centers of each network cluster (connected component) or between them.

4.4.2 Spatiotemporal graph modeling

To research realistic highly dynamic networks, especially in mobile urban environments, with a number of nodes and connections varying in time, time-changing spatial graphs can be studied. Although modeling of such spatiotemporal graphs poses numerous challenges, it can be achieved on the basis of accurate geographical location and telemetry data that can be available more and more often. To prove it, the graphs discussed in the next sections were constructed.

They can be used, for instance, in the studies on opportunistic mobile DTN data routing between network clusters or dispersed nodes to enable the delivery with minimum cost or delay. Moreover, in particular, when more mobile nodes data related to smartphones, cars, bikes, etc., are available, the research may involve the analysis of street traffic trends and determination of most busy (hot-spot) areas in the city at given time of day, as well as general-purpose smart city infrastructure planning.

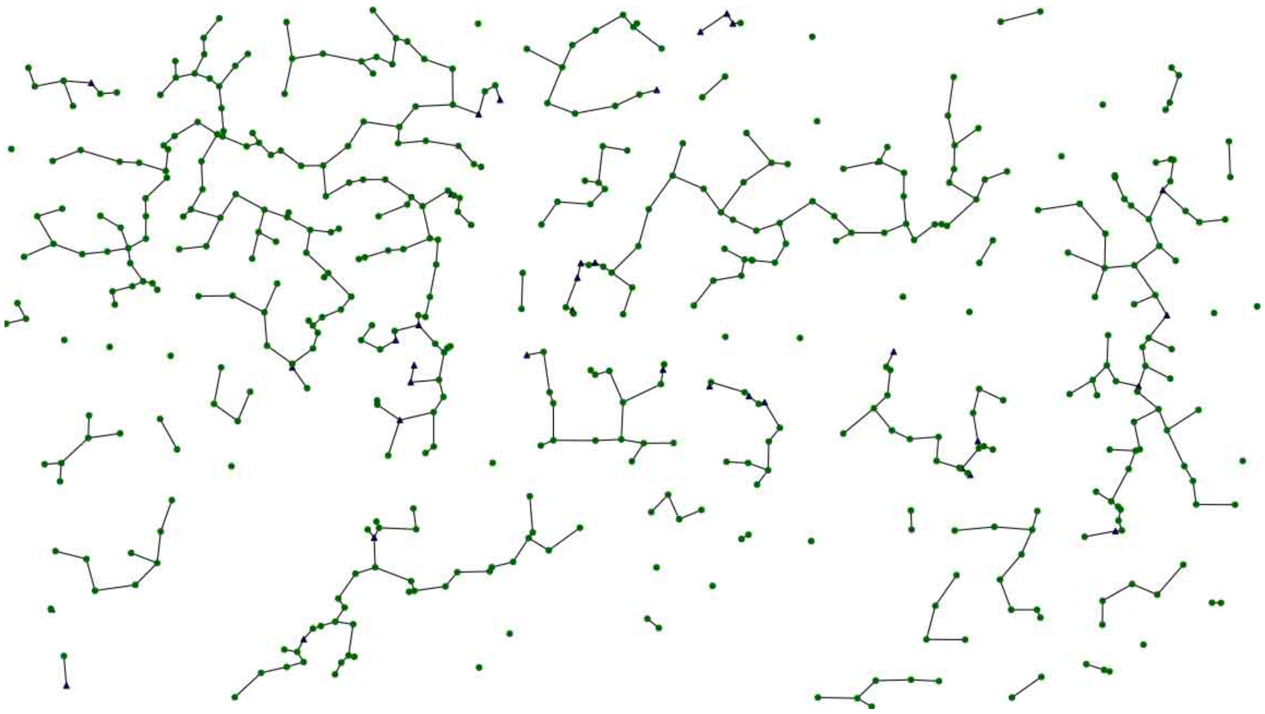


Figure 4.14 Modeled minimum spanning forest

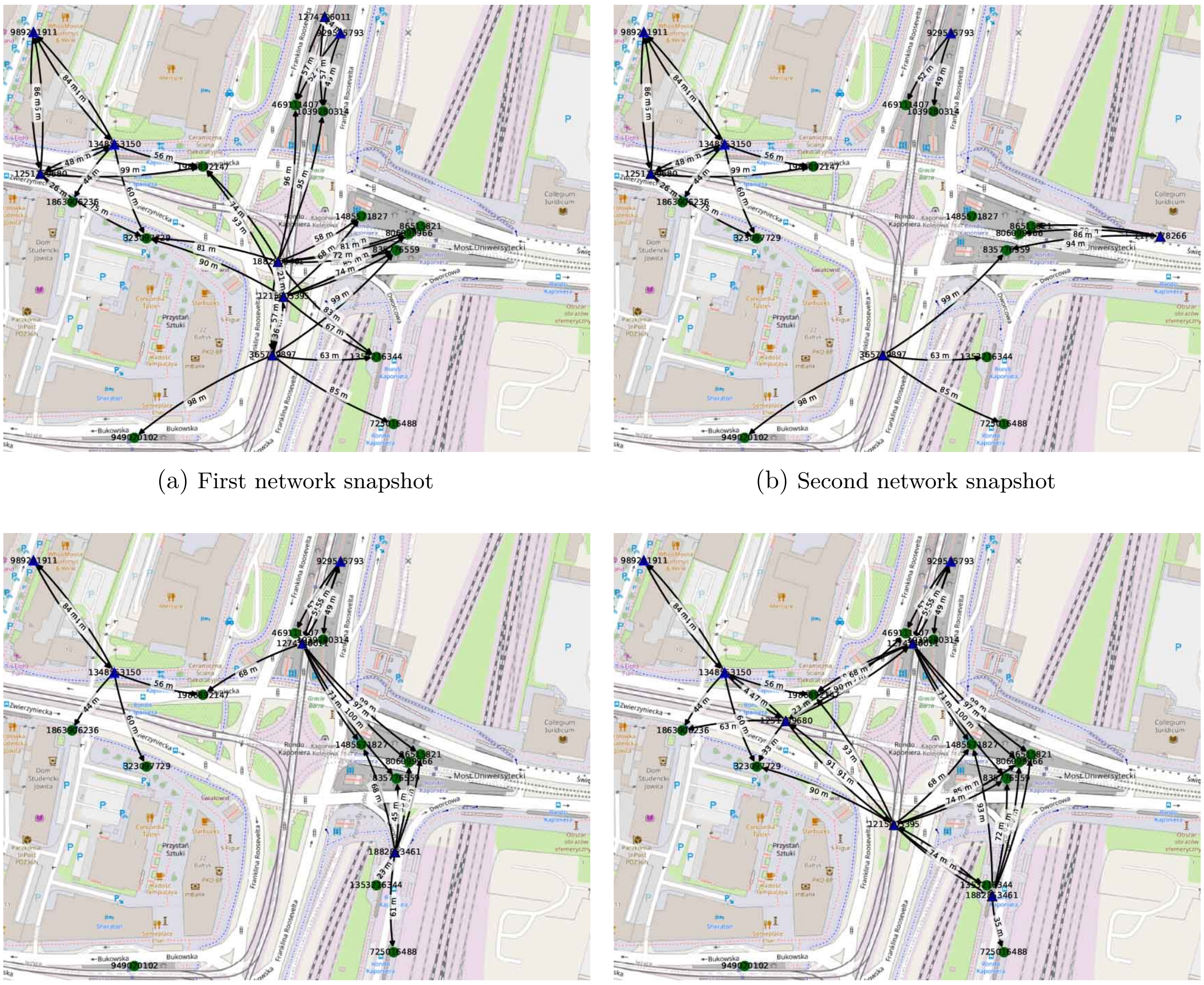
Evolving connectivity graph

The first four *snapshots* of a *dynamic network* modeled as an *evolving graph*, i.e., a sequence of space connectivity graphs, is presented in Figure 4.15. Those graphs were generated using Algorithms 1 (NDD-SSN) and 2 (SSN-SCL). Each space connectivity graph depicts potential radio connections between the networked devices in the area of the Kaponiera Roundabout in the city center of Poznań. The whole sequence (list of directed space connectivity graphs) spans 120 s on 27 November 2019, starting at 3:00 p.m. Similar to the static space connectivity graph presented in Section 4.4.1, it includes stationary (i.e., air quality monitoring stations, parking meters, public transport stops, and ticket machines) and vehicular nodes (i.e., buses and trams). Each node was a single device with its precise satellite-positioning based location known. Not only geographic coordinates were provided by data sources, but also various additional parameters related to the nodes, such as the speed of each bus and tram, the names of the stops, payment methods supported by parking meters, or air quality readings. They were not needed, though, in this network dynamics and routing modeling proof of concept but could be used, for example, in prediction-based studies or environmental and social trends analysis. The modeling parameters were the following:

- Number of slots: 20;
- Slot length: 6 s;
- Area dimensions: 357 m by 272 m;
- Area boundaries:
  - Latitude: 52.406511 – 52.408955;
  - Longitude: 16.909878 – 16.915140;
- Radio coverage: omnidirectional, radio range: 100 m;
- Relay nodes:
  - Mobile nodes: buses and trams;
  - Stationary nodes: air quality meters and parking meters;
- Destination nodes: public transport stops and ticket machines.

The green circles in Figure 4.15 are destination nodes and the blue triangles are the relays. The mobile nodes are marked with the black symbol border. The node labels (e.g., *323097729*) are unique identifiers of the devices. The edge labels, such as 68 m, indicate the geographical distance between devices.

To model the network, first, the gathered data samples related to the instances of the nodes present in the area of interest in the analyzed period were discretized (i.e., grouped), according to their timestamps, into consecutive time intervals (slots) of chosen duration. This duration is the resolution of the modeling process and should be set to be at least as short (as high) as the update interval of the most frequently updated data source. To mimic the envisaged DTN application, the space nodes (devices) were categorized into two sets—relay and destination nodes—based on their class, i.e., the device type and assumed network capabilities. Relays may originate connections with other devices, i.e., be the starting nodes for directed edges, while destination nodes can only be



(a) First network snapshot

(b) Second network snapshot

(c) Third network snapshot

(d) Fourth network snapshot

Figure 4.15 Modeled evolving connectivity graph

at the receiving end of a directed link. Then, similarly to the spatial graph modeling approach presented in Section 4.4.1, the radio connectivity graph was constructed for each of the time slots, based on the location, assumed radio range, and role of each node. The edges of such graphs shall be called space edges, being the representation of network connections and directed single message transmissions, modeled to be feasible in a given smallest considered time interval (slot). The network modeled in this way can be analyzed and used to solve different topology and routing problems defined in Section 2.3, as presented in the next subsections.

Space-time connectivity graph

In this section, the *evolving graph* from Section 4.4.2 is transformed, using Algorithm 3 (SCL-STCG), into a *space-time graph* (which is a *time-expanded graph*). The produced layered *space-time connectivity graph* depicted in Figure 4.16 consists of multiple temporal instances of space nodes, each being a distinct graph node linked with a directed edge to the next temporal instance of it and

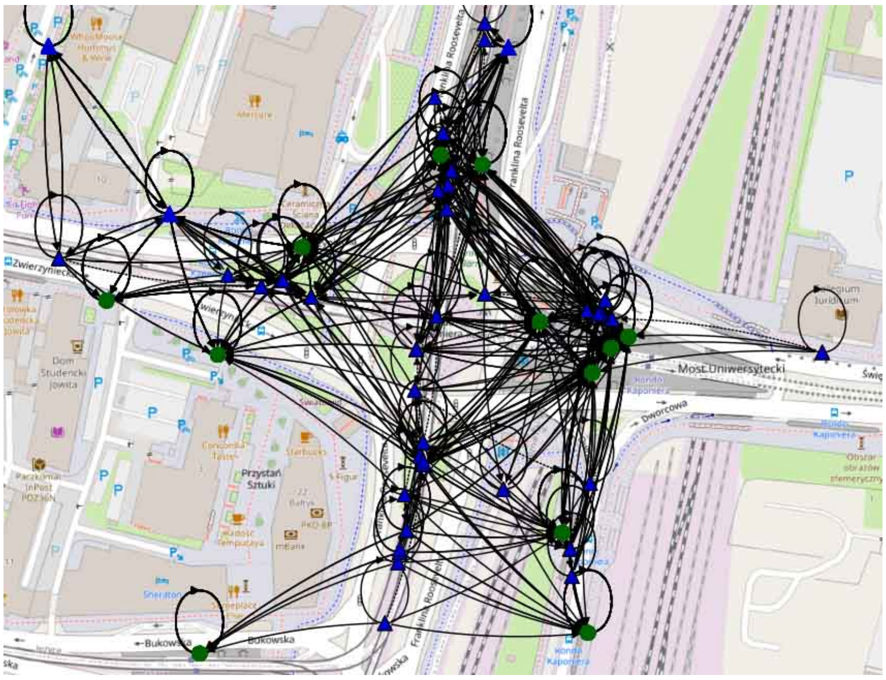


Figure 4.16 Modeled space-time connectivity graph

to respective instances of other space nodes (network devices). Such a graph preserves all information about spatial and temporal relationships between nodes—as opposed to a simple composition (addition) of nodes and edges of all spatial graphs, which would likely lead to misrepresentation caused by the existence of edges which would not be possible in the network if the direction (flow) of time was obeyed. Each node was visualized in the location of its first appearance in the graph. Visible self-loops mean that there was more than one node (instance) present in a given location. The key characteristics of the presented space-time connectivity graph are the following:

- Space nodes: 26 — stationary nodes: 15, mobile nodes: 11;
- Node instances: 714;
- Average node degree: 3.78, edges: 1348, space cost: 41580 m, time span: 20 slots.

First-contact graph

The *space-time connectivity graph* shown in Figure 4.16 can be transformed with Algorithm 4 (STCG-FCG) into a *first-contact graph*, presented in Figure 4.17. Such a graph can be viewed as a form of a *time-aggregated graph*. The edge labels presented in Figure 4.17 depict the slot number in which the space edge existed between two nodes and geographical distance between them. The connections that take place in a given time slot, i.e., the spatial edges, are depicted as solid lines. Time-delayed spatial edges are indicated by dotted lines (i.e., a number of slots had to pass before the contact between two space nodes was possible). For instance, the dotted edge labeled *5 (89 m)* means that a time-delayed connection existed in slot number 5 and the distance between devices was 89 m at this time. These are the parameters of the graph:

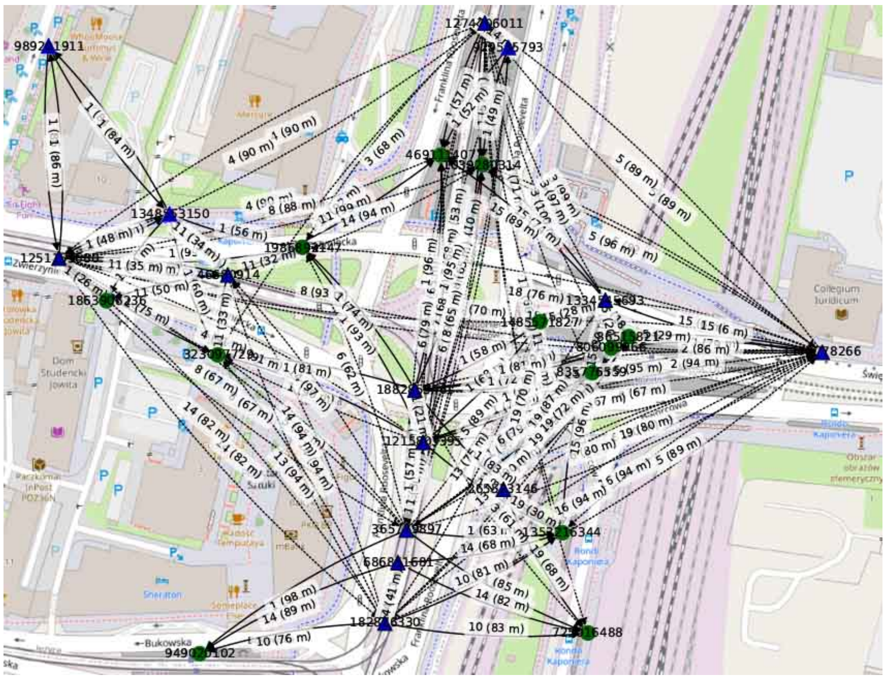


Figure 4.17 Modeled first-contact graph

- Space nodes: 26 — stationary nodes: 15, mobile nodes: 11;
- Average node degree: 10.92, edges: 142, space cost: 9949 m, time span: 20 slots.

Because some of the time-order related information might be lost in the process, a *first-contact graph* will be of no use for searching the shortest spatiotemporal paths and solving related problems. It can be used, though, in the overall analysis of the contacts of selected nodes. With this approach, different graph and node parameters can be computed and investigated, such as the first-contact node degrees computed to select the nodes with the largest number of first contacts in the network. Hence, a graph of this type can be used in the design of data offloading mechanisms described in Section 2.3.3.

Opportunistic localized class-based multicast tree

The literature presents a number of algorithms for building multicast trees and methods for assessing their quality mainly for static networks [178, 179]. The method for building multicast trees in opportunistic routing environments characterized in Section 2.3.1 can serve as another usage example. The objective of this process is to select spatial and temporal edges which allow a message (or a data bundle) to be transmitted (routed) within a given time period of interest, from the source node to all other nodes which belong to a particular class—here, a selected space node type. Each message will be propagated along its own tree. Every node is aware only of its local neighborhood, i.e., of other nodes within its radio range. In this way, e.g., control and emergency messages can be distributed to a defined class of devices.

To model this type of *time-aggregated graph*, a time series of consecutive spatial graphs, i.e., *evolving graph* constructed in Section 4.4.2, is used as the input for a distributed opportunistic

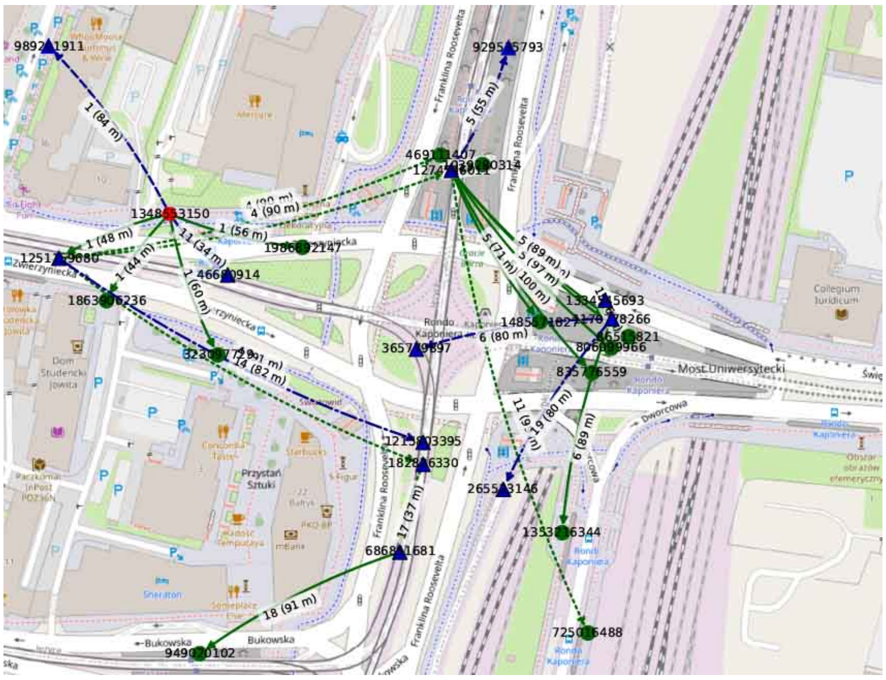


Figure 4.18 Modeled opportunistic localized class-based multicast graph

multicasting algorithm to construct a new graph—the opportunistic time-aggregated multicast tree. First, one of the relay-class nodes originates the message and becomes the root of the tree, as well as the first actual relay that stores this bundle and forwards it opportunistically. Then, in each following time slot, every relay which stores the message, i.e., every node that became a member of the tree, attempts to forward it to next relays or destination nodes, and hence new relay nodes are connected to the tree. As in Algorithm 6 (LKOM), to make routing decisions, each relay node uses its own local knowledge about past transmissions and its neighborhood, being aware of all other nodes in its radio range. Thus, every directed link between two nodes, either spatial or temporal edge, corresponds to a message transmission. It is assumed that such a transmission can successfully be completed within a single time slot and relay nodes operate with message buffers of unlimited capacity (infinite bundle queues). If a node already belongs to the tree, i.e., received the message, it will not be considered to be a recipient of future transmissions of this message, and therefore will be connected to the tree only once to store and forward or process the bundle, being a relay or a destination, respectively.

As shown in Figure 4.18, this algorithm successfully constructed a multicast tree, and the message could reach every destination node in this opportunistic routing scenario related to Section 2.3.1. Due to the opportunistic nature of the algorithm, there is a number of stub relay nodes that are connected to the tree but do not belong to the core part of it, i.e., do not lead to any destination nodes. In the visualization, the orange hexagon node labeled *1348553150* is the source, the green circles are multicast receivers (terminals), and the blue triangles are the relays. The mobile nodes are marked with a black border. The green edges are the edges that compose the actual (core) multicast tree, i.e., are the building blocks of the paths leading to multicast receivers. The blue edges are connections to the relays that do not lead to the terminals. The graph metrics are as follows:

- Space nodes: 25 — stationary nodes: 15, mobile nodes: 10;
- Multicast tree nodes: 25 — core relays: 5, stub relays: 7, destinations: 12;
- Average node degree: 1.92, edges: 24, space cost: 1685 m, time span: 19 slots.

To avoid bundle buffer overflows or transmission medium oversaturation in this simplistic example, one can consider the implementation of the mechanisms such as *time-to-live* (TTL) of a message (expiration time), *last-in, first-out* (LIFO) limited capacity bundle queues, the selection of a smaller number of relays in denser networks, etc., or the use of more advanced protocols.

Spatiotemporal shortest paths

Yet another usage area may be related to searching for globally optimal opportunistic shortest paths, based on full spatial and temporal knowledge on the structure of the graph, and, hence, network topology, as in Algorithm 7 (GKOM). Such knowledge can either be available *a posteriori*, i.e., after the observation period, or beforehand, to some extent, when prediction models that are sufficiently precise can be used. This is quite the opposite to the SBCM case, where only space- and time-limited localized topology-related knowledge is available to the routing algorithm.

What is crucial is that the presented method allows a *space-time graph* to be analyzed using well-known methods and tools designed for directed static graphs, sometimes only with minor modifications, such as one of the well-known shortest-paths-finding algorithms [180]. One can, for example, in an opportunistic routing case related to Section 2.3.1, look for the geographically shortest paths and the time shortest (fastest) paths [165], as well as the energy shortest paths between two nodes, provided such spatiotemporal paths exist [181]. Another aim to be achieved can be to

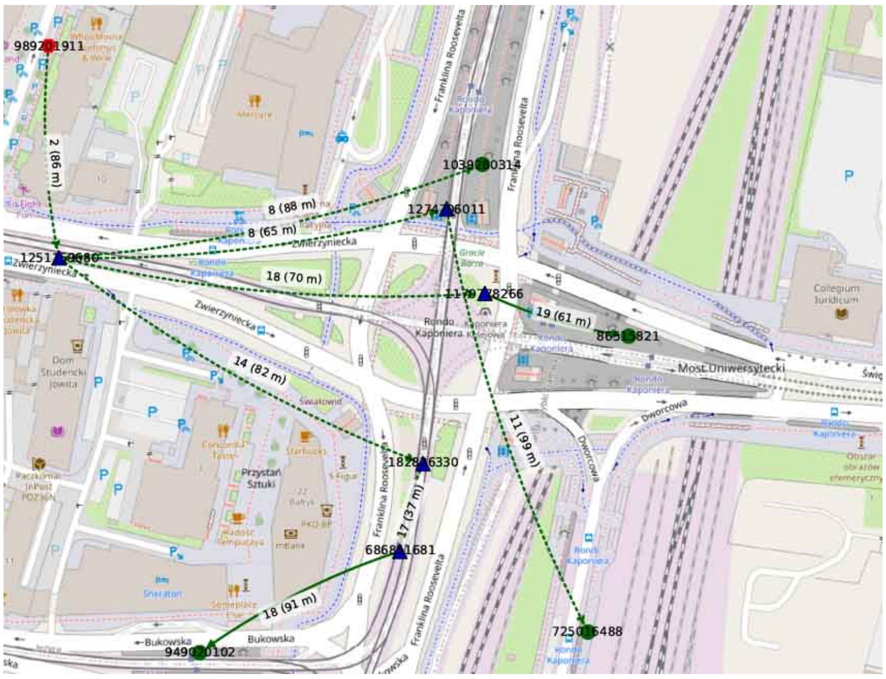


Figure 4.19 Modeled opportunistic spatiotemporal shortest paths to selected nodes

construct a multicast tree that consists of the shortest spatial distance directed paths between the root (source) and the leaves (designated destinations), i.e., a shortest paths tree [182].

An example *time-aggregated tree* constructed in the *evolving graph* modeled in Section 4.4.2 using Dijkstra's algorithm [183] is presented in Figure 4.19. The source is depicted as the orange hexagon node labeled *989201911*, the five relay nodes are the dark blue triangles, and the four destination nodes are the green circles. The parameters of the tree are as follows:

- Space nodes: 10 — stationary nodes: 5; mobile nodes: 5;
- Multicast tree nodes: 10 — core relays: 5; destinations: 4;
- Average node degree: 1.8; edges: 9; space cost: 679 m; time span: 20 slots.

4.5 Observations

The deployment of connected sensors in urban environments and the widespread availability of data they provide opens up new areas of research and creates novel study opportunities. As demonstrated, owing to publicly available online data sources, it has become possible to model real-life dynamic urban sensor networks. The introduced network modeling approach based on exact, time-varying location and number of heterogeneous sensor nodes has not previously been considered and presented in the literature. Captured topologies can be represented as accurate spatiotemporal graphs. In this way, the key research problems in the area may be studied, i.e., opportunistic routing, data aggregation, and data offloading, as proved by the presentation and discussion of numerous graphs modeled in the feasibility study.

The results of presented preliminary routing-related research indicate the complex nature of network dynamics in urban sensor networks, and hence the introduced network modeling architecture, and static (spatial) and dynamic (spatiotemporal) graph models enable network researchers to perform various types of routing-related studies using graph theory methods. As presented, those may include the development of concepts for opportunistic routing in delay-tolerant networks based on actual geographic location of nodes. More generally, they can be used in the emerging real-life data-based research or play the key role in the incorporation of new ideas into well-studied routing concepts in MANETs, VANETs, WSNs, etc. They will aid applications such as low-powered everyday and distributed delay-tolerant measurements data collection or dissemination of messages during emergencies, pandemics, power outages, and natural disasters.

Chapter 5

Urban delay tolerant multicast using uncontrolled mobile relay

To address the main objective of the research, *delay tolerant multicast framework* has been designed to meet the needs of heterogeneous urban sensor networks. The next sections describe design considerations and define the family of new generalized multicast message-oriented algorithms. The model of related *delay tolerant multicast router* is discussed and final observations are presented.

5.1 Design considerations

The key design objective is to create algorithms that will be practical and easy to implement, maintain, tweak and further develop. To enable full and unambiguous comprehension of presented algorithms and procedures, related introductory clarifications and comments are given in the following subsections.

5.1.1 Protocol agnosticism

Protocol and technology agnosticism go hand in hand with algorithmic approach and graph-based network structure abstraction. Therefore, presented multicasting methods do not follow, extend or mimic any particular existing routing protocols. Data structures and procedures are as simple and as generic as possible. Key elements are presented in an algorithmic way inspired by popular set theory notations and JSON structure. This enables high-level research and analysis, as well as, implementation according to the specifics of communication technologies and protocols of choice. For instance, in the context of DTN protocols *messages* are frequently referred to as bundles [184]. Similarly, *destination* node or nodes can be called an endpoint [185].

Some well known mechanisms can be noticed in introduced multicast algorithms, e.g., elements of epidemic routing [186], the usage of message *time-to-live* (TTL) [187], and use of periodic beacons in greedy geographic forwarding [188]. Other presented mechanisms have not been used before. On the one hand, they are dictated by heterogeneous nature of modeled urban sensor networks, i.e., various classes and different roles of nodes. On the other hand, presented multicast framework combines related procedures tailored to perform different forms of multicasting.

5.1.2 Uncontrolled mobile relays

At the heart of designed algorithms is the use of uncontrolled relay nodes, i.e., routers that move independent from the influence of routing algorithms they are running [189]. In other words, routing is not the main function of such nodes (devices) and is not influenced by their network functions. It is crucial to differentiate such uncontrolled mobile relays from autonomous nodes (agents) that could be designed and programmed to move strictly for the needs of their network role and functions. Due to uncontrolled mobility and opportunistic contacts, to achieve desired coverage and connectivity, multiple uncontrolled relay nodes should be used, especially in vast urban areas of sparse structure and limited numbers of stationary relay nodes.

5.1.3 Heterogeneous networks

The analysis of connected sensing devices in current cities and their expected future development leads to the conclusion that heavily networked urban structures shall be assumed to be heterogeneous. This means that, on the one hand, numerous classes of nodes (devices) will be operating in such networks. On the other hand, to be interoperable, as in any kind of networks, they need to use compatible message exchange procedures. Although, their role and capabilities can be different, depending on the network function they perform. Some might be simple stand-alone sensors, while other will combine advanced sensing and processing, as well as, routing and storage features. This entails modular and scalable design of routing procedures, to enable them to be applied in the scope fitting the needs and capabilities of particular node class.

5.1.4 Network utilization and delivery

A fundamental feature of opportunistic heterogeneous urban networks is the high dynamics of network topology. Frequent changes take place not only in the structure of connections but also in the number and type of nodes available at a given moment in the area of interest. Therefore, some nodes, i.e., stationary or the ones of low mobility, may be able to maintain longer-term connectivity. Other nodes, i.e., the ones in more rapid motion, might be able to establish and keep the connection only in a very limited time-frame. Such short periods do not always enable to implement more advanced transmission reliability methods, such as, acknowledgments and retransmissions. Moreover, in dense urban environments, the number of nodes is expected to be high and growing. Therefore, in DTN scenarios, carefully designed and implemented controlled network flooding with multiple copies is a viable method to ensure required multicast delivery rate. When designing and implementing algorithms of this family, though, loop and network saturation avoidance mechanism need to be implemented. The ones that are introduced to presented algorithms are the use of time-to-live counter, as well as, the list of the identifiers of already *received* messages. In this way, the number of message replications is limited and each network node stores and distributes only a single copy of given message.

5.1.5 Computing power, storage management and radio connectivity

In the designed algorithms, it is assumed that each node is capable of performing the set of operations dictated by its class and role. The aspects such as power usage, computing capabilities optimization, as well as, advanced storage management are not of interest in case of presented algorithms. The reasoning behind it is that many current sensing nodes are, in fact, integrated parts of more advanced devices of high processing and storage capabilities, such as, smartphones, onboard computers, and fixed power-grid or solar-panels connected measurement stations. In case of simpler and battery-powered devices, though, advanced power-efficiency mechanisms will most likely have to be utilized. Such mechanisms might involve, e.g., data aggregation, advanced buffer management, as well as, collision avoidance, message scheduling and duty-cycle management. They are out of the scope of discussed high-level multicast message dissemination design. Similarly, it is assumed that each node can have its own unique radio connectivity capabilities and coverage, also changing in time, depending on its location, propagation conditions, and configuration.

5.1.6 Node location

Geographic *location* of the node is expressed as pair of latitude and longitude coordinates. And yet, any other coordinate and location system could be used if it better meets the needs of a particular application. Therefore, location capabilities need to be available to mobile nodes. Those can be based on the use of satellite-based positioning systems, as well as, other location technologies, e.g., using radio-beacons to triangulate and estimate current location with sufficient precision. In some scenarios, where precise route planing and scheduling is involved and feasible, the location of the nodes can be predicted or even expected. In case of stationary nodes, they can even be not equipped with positioning hardware themselves but programmed with their location measured when the devices are deployed (installed).

5.1.7 Next hops and destinations

The term *next hop* means the node, i.e., device or routing process, which is planed to receive the message next. Such nodes can be the destinations of given message, the relays to carry or transmit it further, as well as, both of these. Next hops and destinations can be either defined by some attributes of the neighboring nodes, e.g. their *class*, *location* or *type*, as well as, by their identifiers.

5.1.8 Generic and external elements

To keep presented methods unobscured and focused on routing, a number of generic and external elements are used. For instance, there are a few generic node-related parameters available. Some are static (configured at the start of routing process) and accessible with dedicated procedures, such as, *getNodeClass()*, and *getNodeId()*. Moreover, the *router* is an element accessible by each algorithm and procedure. It is the reference (handle) which provides access to current state of the router and all used data structures. Other elements are dynamic (changing over time), e.g., current location of the node available every time *getNodeLocation()* is used. Similarly, *transmitBeacon()*, *transmitMessage()*, and *receiveMessages()* are the basic lower-level communication procedures assumed to be available to each routing process. One can think of them as of external functions accessible by a fundamental network connectivity API of the node. Furthermore, the processing of received messages is beyond the scope of routing, and therefore, messages destined for current node are passed to an external higher-level *processPayload()* procedure.

5.2 Delay tolerant multicast framework

In this section, three new heterogeneous opportunistic DTN multicast algorithms for dynamic urban environment are presented. While sharing main structure and features, they are designed to operate in different network knowledge conditions, destination group definition methods, node classes and roles. In particular, uncontrolled mobile relay nodes (agents) can be used to improve connectivity and extend network coverage. With three knowledge modes and three target group types they can be even seen as nine different algorithms. For simplicity and to ease the implementation, they are, though, presented in consolidated form, which enables users to study and implement single ones, as well as the whole family with limited effort. The algorithms are divided into procedures to ensure modularity and clarity of the presentation.

It is assumed that the data are exchanged in the form of *messages* which are the smallest possible portions of data meaningful to the application that receives them. The algorithms do not construct and maintain routing tables. Their objective is to opportunistically convey stored messages based on node location and available knowledge about its surroundings. Each network node (or routing process) performs all or some of the basic routing operations, i.e., receive, store, or transmit messages. Therefore, a single execution of Algorithms 5, 6 and 7 is an iteration over all $storedMessages$ aimed at transmission of those, following set conditions and steps, updating storage of messages accordingly, and receiving new messages. The destination nodes, i.e., the multicast group, are defined by the originating node in the parameters of each message as a set of given nodes, a set of node classes, or by the boundaries of target geographic region. Nodes do not join a multicast group or subscribe to message distribution service on their own. Therefore, for each originated message routed through the network, any given node is from the start either a destination node or it is not. The nodes are aware of their own network identifier, class and location. Due to broadcast nature of wireless medium, single omnidirectional transmission can reach multiple neighboring devices. To take advantage of this feature, each relayed message is directed at once at all destination nodes and relays of interest in the radio range of current node.

Message forwarding flowchart related to a single iteration is depicted in Figure 5.1. The purpose of this swimlane diagram is to be a high-level overview of the steps, common elements and relationships between introduced multicast algorithms. For reasons of clarity, some details are omitted or generalized. Please refer for those to the respective pseudocodes in the next sections.

The diagram is organized into three horizontal lanes. The middle one related to no knowledge opportunistic multicast is also the common part of local and global knowledge based algorithms. Each algorithm begins at respective “start” step and follows flowlines of related style, i.e., dashed for NKOM, solid for LKOM, and dotted for GKOM. In the common (middle) lane the flowlines ended with diamonds can be followed by each algorithm. An ellipse depicts the beginning or the ending

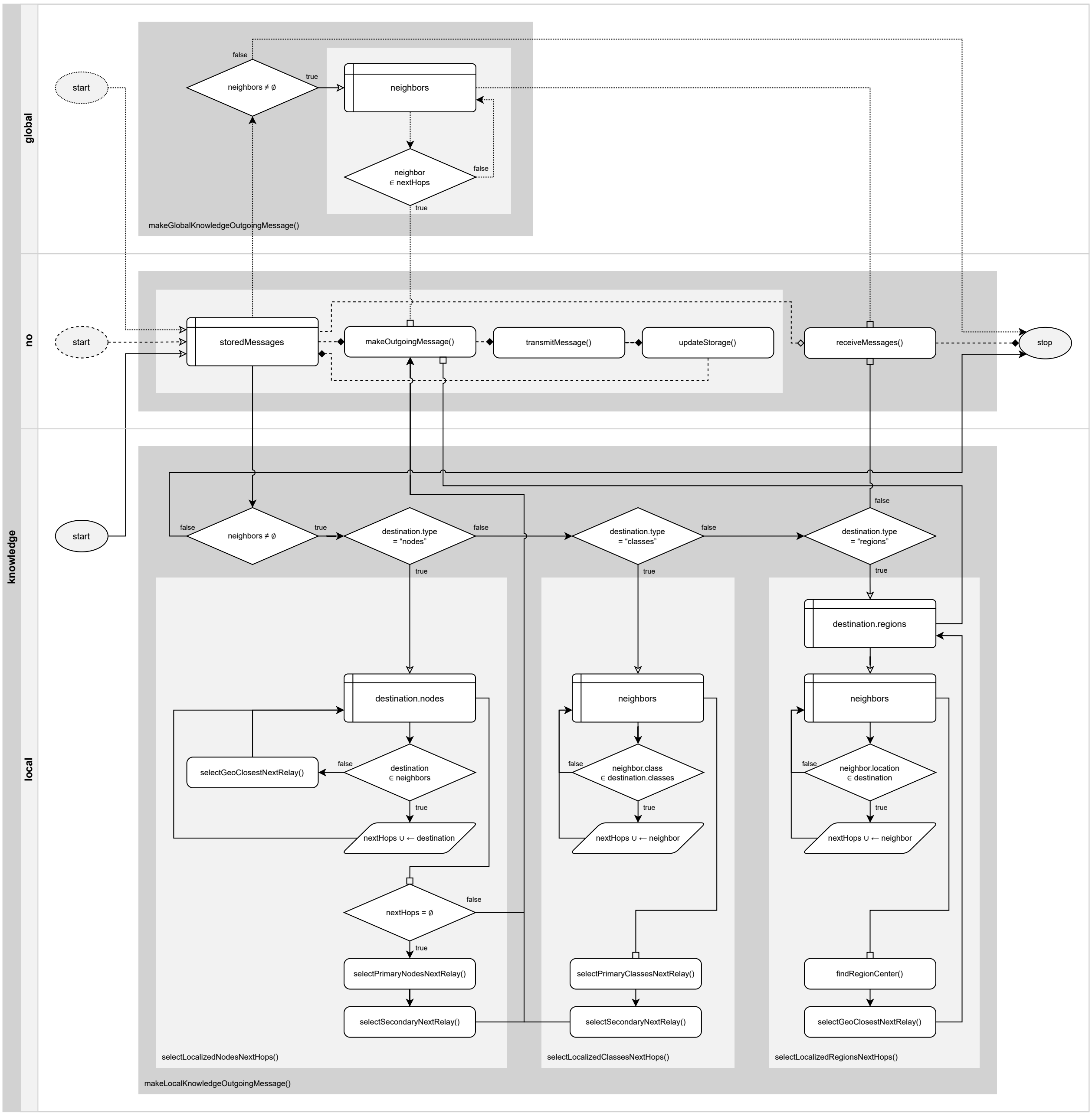

knowledge
global
no
local
start
start
start
false
true
neighbors ≠ ∅
neighbors
neighbor ∈ nextHops
false
true
makeGlobalKnowledgeOutgoingMessage()
storedMessages
makeOutgoingMessage()
transmitMessage()
updateStorage()
receiveMessages()
stop
false
neighbors ≠ ∅
true
destination.type = "nodes"
false
true
destination.type = "classes"
false
true
destination.type = "regions"
false
true
destination.nodes
selectGeoClosestNextRelay()
false
destination ∈ neighbors
true
nextHops ∪← destination
nextHops = ∅
false
true
selectPrimaryNodesNextRelay()
selectSecondaryNextRelay()
selectLocalizedNodesNextHops()
neighbors
false
neighbor.class ∈ destination.classes
true
nextHops ∪← neighbor
selectPrimaryClassesNextRelay()
selectSecondaryNextRelay()
selectLocalizedClassesNextHops()
destination.regions
neighbors
false
neighbor.location ∈ destination
true
nextHops ∪← neighbor
findRegionCenter()
selectGeoClosestNextRelay()
selectLocalizedRegionsNextHops()
makeLocalKnowledgeOutgoingMessage()

Figure 5.1 Multicast forwarding flowchart

of the algorithm. Rhombus is related to a decision step while a rhomboid represents data-related operations. Plain rectangle indicates a set of operations (a procedure). Rectangle with column and row heading lines represents a foreach loop for traversing given set of data. The entry point of such loop is indicated by an empty arrowhead and the exit point is marked with and empty square. The elements are grouped into overarching procedures with light and dark gray backgrounds.

5.2.1 No knowledge opportunistic multicast

Algorithm 5, *no knowledge opportunistic multicast* (NKOM), is the most simplistic one in the framework. Procedure *makeNoKnowledgeOutgoingMessage*, at its core makes an *outgoingMessage*

Algorithm 5: No knowledge opportunistic multicast

```
1 foreach $storedMessage \in storedMessages$ do
2     $outgoingMessage \leftarrow makeNoKnowledgeOutgoingMessage(storedMessage)$
3     if $outgoingMessage \neq \varnothing$ then
4         $transmitMessage(outgoingMessage)$
5         $updateStorage(storedMessage)$
6 $receiveMessages()$
```

Procedure makeNoKnowledgeOutgoingMessage(storedMessage)

```
Input:
$storedMessage$                                   // stored message
Output:
$outgoingMessage$                                 // outgoing message
// prepare outgoing message
1 $outgoingMessage \leftarrow makeOutgoingMessage(storedMessage)$
2 output $outgoingMessage$
```

Procedure updateStorage(storedMessage)

```
Input:
$storedMessage$                                   // message stored in current node
1 $storedMessage.ttl\ - \leftarrow 1$             // decrement Time to Live (TTL) counter of stored message
// remove message from the storage when its lifespan is exceeded
2 if $storedMessage.ttl = 0$ then
3     $router.messages.stored\ \setminus \leftarrow\ storedMessage$
```

based on *storedMessage*. Due to no knowledge about its surroundings available to the node, each stored message is transmitted in every algorithm iteration (time-slot). Then, message storage is updated and *time-to-live* (TTL) parameter of *storedMessage* is decremented. The message is removed from the storage when TTL reaches 0. In this way, the simplest possible network saturation avoidance and routing loop termination mechanism for flooding-like message dissemination methods gets implemented [38]. Finally, internal (originated by current node) and external *messages* are received, as presented in Procedure *receiveMessages*. Every *message*, aside from its *payload* contains a number of parameters. Destination *type* is always accompanied by a related set of target *classes*, *nodes*, or *regions*. Message identifier *id* is also present. To ensure its uniqueness, one of well-known *universally unique identifier* (UUID) generation methods can be used [190]. Moreover, each *message* includes the list of nodes *visited* by this message and current TTL value. A set of *nextHops* is not used in no knowledge multicasting.

This epidemic routing inspired approach leads to the lowest computational complexity. Although, due to beacon-less and no network knowledge nature, it is also potentially the most power consuming and radio medium utilizing one. Albeit, thanks to the simplicity, it can be used as base model for further extensions or implementation in the simplest types of nodes of the lowest computing capabilities. The upper bound of time complexity of a single run of NKOM is $\mathcal{O}(|storedMessages|)$, i.e., the algorithm is upper bounded by the number of messages currently stored in the node.

```
Procedure receiveMessages
  Output:
  $messages \leftarrow (message_i)_{i \leftarrow 1}^{j}$ :                      // list of received messages
    $message_i \leftarrow$ (                                                  // message $i$
      $destination_i \leftarrow$ (                                            // set of destination attributes
        $classes_i \leftarrow \{class_k\}_{k \leftarrow 1}^{l}$,              // set of $l$ destination class names
        $nodes_i \leftarrow \{node_m\}_{m \leftarrow 1}^{n}$,                 // set of $n$ destination nodes
        $regions_i \leftarrow \{region_p\}_{p \leftarrow 1}^{q}$,             // set of $q$ destination regions
        $type_i$                                                              // destination type
      ),
      $id_i$,                                                                 // identifier
      $mode_i$,                                                               // network knowledge mode
      $nextHops_i \leftarrow \{nextHop_r\}_{r \leftarrow 1}^{s}$,             // set of $s$ next hops
      $payload_i$,                                                            // message payload
      $ttl_i$,                                                                // TTL
      $visited_i$                                                             // list of nodes visited by the message
    )
  ------------------------------------------------------------
  // receive internally and externally originated messages
1 $messages \leftarrow receiveInternalMessages() \cup receiveExternalMessages()$
2 output $messages$
```

5.2.2 Local knowledge opportunistic multicast

Algorithm 6, *local knowledge opportunistic multicast* (LKOM), is a significant extension of NKOM algorithm. The changes are due to the usage of locally available real-time knowledge about current network neighborhood of the node. Therefore, Procedure *makeLocalKnowledgeOutgoingMessage* is able to take advantage of more sophisticated localized message forwarding techniques tailored to destination types. Together with network structure dynamics caused by the movement of mobile nodes, they enable multicast message to avoid getting stuck in a local optimum with no progress toward desired destinations. The $outgoingMessage$ creation process will be started only if currently there are any neighbors present and $nextHops$ were selected by one of the greedy procedures discussed in the next subsections.

The upper bound of time complexity of a single run of LKOM is related to the number of stored messages, current neighbors, and maximum number of destination elements of forwarded messages:

- nodes: $\mathcal{O}(|storedMessages| \cdot |neighbors| \cdot |destination.nodes|_{MAX})$;
- classes: $\mathcal{O}(|storedMessages| \cdot |neighbors| \cdot |destination.classes|_{MAX})$;
- regions: $\mathcal{O}(|storedMessages| \cdot |neighbors| \cdot |destination.regions|_{MAX})$.

```
Algorithm 6: Local knowledge opportunistic multicast
1 foreach $storedMessage \in storedMessages$ do
2     $outgoingMessage \leftarrow makeLocalKnowledgeOutgoingMessage(storedMessage)$
3     if $outgoingMessage \neq \varnothing$ then
4         $transmitMessage(outgoingMessage)$
5         $updateStorage(storedMessage)$
6 $receiveMessages()$
```

Procedure makeLocalKnowledgeOutgoingMessage(storedMessage)

```
Input:
$storedMessage$                                             // stored message
Output:
$outgoingMessage$                                           // outgoing message
1  $outgoingMessage \leftarrow \varnothing$                 // set outgoing message to an empty set
2  $neighbors \leftarrow router.neighbors$                  // get the set of current neighbors
3  if $neighbors \neq \varnothing$ then
4      $nextHops \leftarrow \varnothing$                    // initialize an empty set of next hops
       // proceed based on destination type of stored message
5      switch $storedMessage.destination.type$ do
           // destination type set to a group of individual nodes (identifiers)
6          case "nodes" do
7              $nextHops \leftarrow selectLocalizedNodesNextHops(storedMessage)$
           // destination type set to node classes
8          case "classes" do
9              $nextHops \leftarrow selectLocalizedClassesNextHops(storedMessage)$
           // destination type set to regions
10         case "regions" do
11             $nextHops \leftarrow selectLocalizedRegionsNextHops(storedMessage)$
12     if $nextHops \neq \varnothing$ then
           // prepare outgoing message
13         $outgoingMessage \leftarrow makeOutgoingMessage(storedMessage, nextHops)$
14 output $outgoingMessage$
```

Localized next hops to destination nodes

Procedure *selectLocalizedNodesNextHops* determines which of current *neighbors* should receive *storedMessage* and designates them as a set of *nextHops*. For each *destination* node of *storedMessage* a lookup for a *nextHop* is performed. If *destination* is one of *neighbors*, then it gets set as the *nextHop*. Otherwise, the geographically closest next relay neighbor is designated the *nextHop*, following Procedure *selectGeoClosestNextRelay*. When *nextHop* is chosen and neither belongs to the set of nodes *visited* by *storedMessage* nor already to the *nextHops*, it is added to the set of *nextHops*. If no *nextHops* were found for all of *destinations*, it means that no *destination* is a direct neighbor and the *location* od *destination* nodes was unknown. In such case, the search for primary and secondary next hops will be conducted. First, as defined in Procedure *selectPrimaryNodesNextRelay*, *primaryNextRelay* will be chosen, when possible, i.e., a neighbor relay node with the largest number of neighbors (the highest node degree). Then, if *primaryNextHop* got selected, a *secondaryNextHop* following Procedure *selectSecondaryNextRelay* will be determined as current node neighbor which is the geographically furthest one from *primaryNextHop*. In this way, the chances of delivery of *storedMessage* get increased. The message is not only directed to the next relay with the largest number of neighbors, but also maximally geographically distributed for improved coverage and redundancy. Both *primaryNextHop* and *secondaryNextHop* are added to the set of *nextHops* only if they were not yet visited by *storedMessage* to avoid causing a routing

Procedure selectLocalizedNodesNextHops(storedMessage)

```
Input:
$storedMessage$                                                  // stored message
Output:
$nextHops$                                                       // set of next hops
1  $nextHops \leftarrow \varnothing$                              // initialize next hops as an empty set
2  $neighbors \leftarrow router.neighbors$                       // get the set of current neighbors
3  $destinations \leftarrow storedMessage.destination.nodes$     // get the set of destination nodes
4  foreach $destination \in destinations$ do
5      $nextHop \leftarrow \varnothing$                           // initialize next hop as an empty set
       // set destination as next hop if it is a neighbor
6      if $destination \in neighbors$ then
7          $nextHop \leftarrow destination$
8      else
           // determine closest next hop if destination node location is known
9          if $destination.location \neq \varnothing$ then
10             $nextHop \leftarrow selectGeoClosestNextRelay(destination.location)$
11     if $nextHop \neq \varnothing$ and $nextHop \notin (storedMessage.visited \cup nextHops)$ then
12         $nextHops \cup\leftarrow nextHop$
   // select primary and secondary next hop if no next hops were found
13 if $nextHops = \varnothing$ then
       // select primary next hop (relay)
14     $primaryNextHop \leftarrow selectPrimaryNodesNextRelay()$
15     if $primaryNextHop \neq \varnothing$ and $primaryNextHop \notin storedMessage.visited$ then
16         $nextHops \cup\leftarrow primaryNextHop$
       // select secondary next hop (relay) if primary got selected
17     if $primaryNextHop \neq \varnothing$ then
18         $secondaryNextHop \leftarrow selectSecondaryNextRelay(primaryNextHop)$
19         if $secondaryNextHop \neq \varnothing$ and $secondaryNextHop \notin storedMessage.visited$ then
20             $nextHops \cup\leftarrow secondaryNextHop$
21 output $nextHops$
```

loop and unnecessary transmission medium usage and computation-related power consumption at the receiving side.

Localized next hops to destination classes

Procedure *selectLocalizedClassesNextHops* determines current *nextHops* of a *storedMessage* destined for desired *classes*. First, each current direct neighbor that belongs to one of destination *classes* is added to the set of *nextHops*. Then, additionally, a *primaryNextHop* is selected in Procedure *selectPrimaryClassesNextRelay*. This next hop is one of relay nodes with the largest number of destination classes neighbors, i.e., highest classes node degree. In such manner, *storedMessage* gets forwarded to the relay which is currently able to reach the largest number of other nodes of destination classes of interest. Next, if *primaryNextHop* got chosen, Procedure *selectSecondaryNextRelay* will be used to select a *secondaryNextHop*, i.e., a current node neighbor which is the geographically furthest one from *primaryNextHop*, when more relays are available. Similarly to Procedure *selectLocalizedNodesNextHops*, *primaryNextHop* and *secondaryNextHop* are added to

Procedure selectGeoClosestNextRelay(location)

```
Input:
location                                              // location of interest
Output:
closestNextRelay                                      // closest next relay

1  $closestNextRelay \leftarrow \varnothing$
2  $minDistance \leftarrow \infty$
3  $neighbors \leftarrow router.neighbors$                // get the set of current neighbors
4  foreach $neighbor \in neighbors$ do
5      if $neighbor.role =$ "relay" then
6          $distance \leftarrow geoDistance(location, neighbor.location)$
           // set current neighbor as secondary next relay
7          if $distance < minDistance$ then
8              $closestNextRelay \leftarrow neighbor$
9              $minDistance \leftarrow distance$
10 output $closestNextRelay$
```

Procedure selectPrimaryNodesNextRelay

```
Output:
primaryNextRelay                  // relay node with the largest number of neighbors

1  $primaryNextRelay \leftarrow \varnothing$
2  $maxNeighborsNumber \leftarrow -\infty$
3  $neighbors \leftarrow router.neighbors$                // get the set of current neighbors
4  foreach $neighbor \in neighbors$ do
5      if $neighbor.role =$ "relay" then
           // set current neighbor as primary next relay
6          if $neighbor.nodes.number > maxNeighborsNumber$ then
7              $primaryNextRelay \leftarrow neighbor$
8              $maxNeighborsNumber \leftarrow neighbor.nodes.number$
9  output $primaryNextRelay$
```

the set of *nextHops* only if they do not already belong to *nextHops* and were not yet visited by *storedMessage*. By this means, one of routing loop avoidance mechanism gets implemented and resources utilization is lowered.

Localized next hops to destination regions

Procedure *selectLocalizedRegionsNextHops* chooses *nextHops* of a *storedMessage* which lead to nodes in desired destination *regions*. This particular type of multicasting is usually called *geocasting* [181]. For each *destination* region it is checked if each *neighbor* is located within this region. If it is, then it is set current *nextHop* and added to the set of *nextHops* if it was not yet visited by *storedMessage* and not added to *nextHops*. Subsequently, an additional relay is selected, which is geographically closest to the center of *destination* region, to improve message dissemination coverage.

```
Procedure selectSecondaryNextRelay(primaryNextRelay)
    Input:
    $primaryNextRelay$                                                  // primary next relay
    Output:
    $secondaryNextRelay$                         // the furthest relay node from primary next relay
1   $secondaryNextRelay \leftarrow \varnothing$
2   $maxDistance \leftarrow -\infty$
3   $neighbors \leftarrow router.neighbors$                         // get the set of current neighbors
    // select secondary relay node which is the furthest one from from primary next relay
4   foreach $neighbor \in (neighbors \setminus primaryNextRelay)$ do
5       if $neighbor.role =$ "relay" then
6           $distance \leftarrow geoDistance(primaryNextRelay, neighbor)$
            // set current neighbor as secondary next relay
7           if $distance > maxDistance$ then
8               $secondaryNextRelay \leftarrow neighbor$
9               $maxDistance \leftarrow distance$
10  output $secondaryNextRelay$
```

```
Procedure selectLocalizedClassesNextHops(storedMessage)
    Input:
    $storedMessage$                                                      // stored message
    Output:
    $nextHops$                                                          // set of next hops
1   $nextHops \leftarrow \varnothing$                           // initialize next hops as an empty set
2   $neighbors \leftarrow router.neighbors$                         // get the set of current neighbors
3   $destinations \leftarrow storedMessage.destination.classes$    // get the set of destination classes
    // select direct next hop neighbors that belong to a destination class
4   foreach $neighbor \in neighbors$ do
5       if $neighbor \notin storedMessage.visited$ then
6           if $neighbor.class \in destinations$ then
7               $nextHops \cup\leftarrow neighbor.id$           // add neighbor to the set of next hops
    // select primary next hop relay
8   $primaryNextHop \leftarrow selectPrimaryClassesNextRelay(destinations)$
9   if $primaryNextHop \neq \varnothing$ and $primaryNextHop \notin (storedMessage.visited \cup nextHops)$ then
10      $nextHops \cup\leftarrow primaryNextHop$
    // select secondary next hop (relay) if primary got selected
11  if $primaryNextHop \neq \varnothing$ then
12      $secondaryNextHop \leftarrow selectSecondaryNextRelay(primaryNextHop)$
13      if $secondaryNextHop \neq \varnothing$ and $secondaryNextHop \notin (storedMessage.visited \cup nextHops)$
         then
14          $nextHops \cup\leftarrow secondaryNextHop$
15  output $nextHops$
```

Procedure selectPrimaryClassesNextRelay(destinations)

```
Input:
$destinations$                                            // set of destination classes
Output:
$primaryNextRelay$     // relay node with the largest number of destination classes neighbors

1  $primaryNextRelay \leftarrow \varnothing$
2  $maxNeighborsNumber \leftarrow -\infty$
3  $neighbors \leftarrow router.neighbors$                 // get the set of current neighbors
4  foreach $neighbor \in neighbors$ do
5      if $neighbor.role =$ "relay" then
6          $destinationNeighborsNumber \leftarrow 0$
           // check if neighbor has neighbors belonging to destination classes
7          foreach $class \in neighbor.nodes.classes$ do
8              if $class.name \in destinations$ then
                   // add the number of neighbor's neighbors of given destination class
9                  $destinationNeighborsNumber \mathrel{+}\leftarrow class.number$
           // set current neighbor as primary next relay when it has more destination classes neighbors
10         if $destinationNeighborsNumber > maxNeighborsNumber$ then
11             $primaryNextRelay \leftarrow neighbor$
12             $maxNeighborsNumber \leftarrow destinationNeighborsNumber$
13 output $primaryNextRelay$
```

Procedure selectLocalizedRegionsNextHops(storedMessage)

```
Input:
$storedMessage$                                           // stored message
Output:
$nextHops$                                                // set of next hops

1  $nextHops \leftarrow \varnothing$                      // initialize next hops as an empty set
2  $neighbors \leftarrow router.neighbors$                // get the set of current neighbors
3  $destinations \leftarrow storedMessage.destination.regions$   // get the set of destination regions
4  foreach $destination \in destinations$ do
5      $nextHop \leftarrow \varnothing$                   // initialize next hop as an empty set
6      foreach $neighbor \in neighbors$ do
           // check if neighbor belongs to considered destination region
7          if $neighbor.location \in destination$ then
8              $nextHop \leftarrow neighbor.id$
9              if $nextHop \neq \varnothing$ and $nextHop \notin (storedMessage.visited \cup nextHops)$ then
10                 $nextHops \cup\leftarrow nextHop$       // add next hop to the set of next hops
       // select geographically closest next relay to region center
11     $destinationCenter \leftarrow findRegionCenter(destination)$
12     $nextHop \leftarrow selectGeoClosestNextRelay(destinationCenter)$
13     if $nextHop \neq \varnothing$ and $nextHop \notin (storedMessage.visited \cup nextHops)$ then
14         $nextHops \cup\leftarrow nextHop$               // add next hop to the set of next hops
15 output $nextHops$
```

Algorithm 7: Global knowledge opportunistic multicast

```
1 foreach $storedMessage \in storedMessages$ do
2     $outgoingMessage \leftarrow makeGlobalKnowledgeOutgoingMessage(storedMessage)$
3     if $outgoingMessage \neq \varnothing$ then
4         $transmitMessage(outgoingMessage)$
5         $updateStorage(storedMessage)$
6 $receiveMessages()$
```

Procedure makeGlobalKnowledgeOutgoingMessage(storedMessage)

```
Input:
$storedMessage$                                                // stored message
Output:
$outgoingMessage$                                              // outgoing message

1 $outgoingMessage \leftarrow \varnothing$                      // set outgoing message to an empty set
2 $neighbors \leftarrow router.neighbors$                      // get the set of current neighbors
3 if $neighbors \neq \varnothing$ then
4     $nextHops \leftarrow storedMessage.nextHops$              // get the set of next hops
5     foreach $neighbor \in router.neighbors$ do
6         if $neighbor \in nextHops$ then
7             $outgoingMessage \leftarrow makeOutgoingMessage(storedMessage, nextHops)$
              // break the loop because at least one next hop neighbor exists
8             break
9 output $outgoingMessage$
```

5.2.3 Global knowledge opportunistic multicast

Algorithm 7, *global knowledge opportunistic multicast* (GKOM), is based on a principle quite different from the ones in NKOM and LKOM. The key assumption is that the node originating the message possesses not only the knowledge about current topology of the whole network but also of its future structure to the extent needed to efficiently forward messages to desired destinations, i.e., it has global knowledge of the network. Therefore, optimal multicast relays can determined, for example, with Dijkstra's algorithm and encoded in the message as the set of globally known multicast tree $nextHops$. GKOM can also be used as a benchmark when assessing, tuning and further developing the family of NKOM and LKOM algorithms.

The seemingly idealistic scenario is designed for applications characterized by high or complete predictability of nodes locations. It means the use of static node deployment, as well as, precise route planing and scheduling or high precision of location prediction of mobile nodes, especially when central controllers are involved [1]. It is also related to the abstract concept of *knowledge oracles* which provide the nodes with the information on possible contacts in modeled topology. It does not indicate how such knowledge can be distributed in the network and isolates the influence of this aspect on the routing design efforts [191].

When forwarding a $storedMessage$, the $nextHops$ predefined in this message are used. Therefore, in Procedure *makeGlobalKnowledgeOutgoingMessage* an $outgoingMessage$ is generated and then transmitted if there are any $neighbors$ and at least one of them belongs to the set of $nextHops$.

Importantly, unlike in LKOM, there is no differentiation of destination types, and hence, only *"nodes"* type is used. This approach is based on the observation, that since global network topology knowledge is always available, each destination class and region can be *a priori* mapped by the node that originates the message to a set of individual nodes of known locations. The upper bound of time complexity of a single run of GKOM is $\mathcal{O}(|storedMessages| \cdot |neighbors|)$, i.e., it is related to the number of stored messages and current neighbors of the node.

5.3 Delay tolerant multicast router

The algorithms introduced in Section 5.2, NKOM, LKOM, and GKOM, cover the principles of message forwarding throughout a delay-tolerant urban network to achieve various types of multicast dissemination in different network knowledge circumstances. There are, although, more routing and storage management aspects related to message transmission in such a structure, not necessary strictly related to message forwarding. Therefore, in this section, the generalized model of a complete routing process, i.e., a *router*, is presented and discussed. Intentional generality of the description is maintained to not obscure the concepts and keep them comprehensible to all interested parties. Implementation details will vary depending on actual software and hardware platforms.

The upper bound of time complexity of a single iteration of the router depends on enabled knowledge mode. It is also related to the numbers of stored and incoming messages, neighbors (in case of LKOM and GKOM), and maximum number of destination elements among stored messages:

- NKOM: $\mathcal{O}(|storedMessages| + |incomingMessages|)$;
- LKOM:
  - nodes: $\mathcal{O}(|storedMessages| \cdot |neighbors| \cdot |destination.nodes|_{MAX} + |incomingMessages|)$;
  - classes: $\mathcal{O}(|storedMessages| \cdot |neighbors| \cdot |destination.classes|_{MAX} + |incomingMessages|)$;
  - regions: $\mathcal{O}(|storedMessages| \cdot |neighbors| \cdot |destination.regions|_{MAX} + |incomingMessages|)$;
- GKOM: $\mathcal{O}(|storedMessages| \cdot |neighbors| + |incomingMessages|)$.

The general flowchart of proposed delay tolerant multicast router is depicted in Figure 5.2 following the convention described in Section 5.2. Six main procedures are marked with light and dark gray backgrounds. Further components are presented as respective blocks. For details please refer to Sections 5.3.1 and 5.3.2.

5.3.1 Basic router operations

The high-level steps of Algorithm 8, *delay tolerant multicast router* (DTMR), are as simple as possible. First, the *router* gets initiated and continues to operate, as long as, its *state* is set to *"routing"*. In each iteration, consecutively, the current *location* of the *router* is read and kept. Then, radio beacons are processed, if required. Next, stored messages are forwarded, and finally, incoming (received) messages are processed.

It is assumed that many routing processes can be run on a single node, depending on the requirements of particular usage scenario, as well as, capabilities and network role of particular

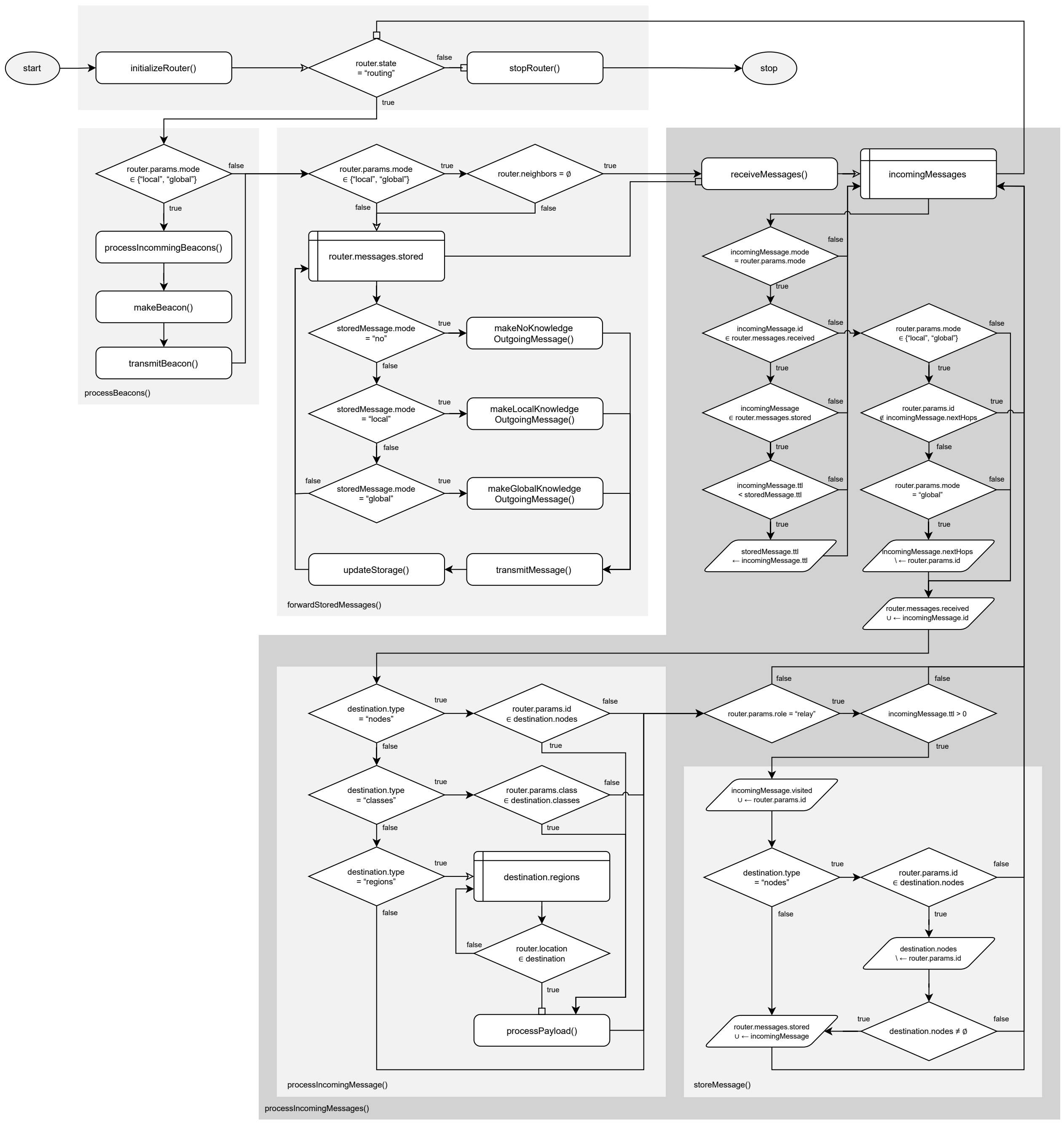

start
initializeRouter()
router.state = "routing"
false
true
stopRouter()
stop
router.params.mode ∈ {"local", "global"}
false
true
processIncommingBeacons()
makeBeacon()
transmitBeacon()
processBeacons()
router.params.mode ∈ {"local", "global"}
true
false
router.neighbors = ∅
true
false
router.messages.stored
storedMessage.mode = "no"
true
false
makeNoKnowledge OutgoingMessage()
storedMessage.mode = "local"
true
false
makeLocalKnowledge OutgoingMessage()
storedMessage.mode = "global"
true
false
makeGlobalKnowledge OutgoingMessage()
updateStorage()
transmitMessage()
forwardStoredMessages()
receiveMessages()
incomingMessages
incomingMessage.mode = router.params.mode
false
true
incomingMessage.id ∈ router.messages.received
false
true
incomingMessage ∈ router.messages.stored
false
true
incomingMessage.ttl < storedMessage.ttl
false
true
storedMessage.ttl ← incomingMessage.ttl
router.params.mode ∈ {"local", "global"}
false
true
router.params.id ∉ incomingMessage.nextHops
true
false
router.params.mode = "global"
false
true
incomingMessage.nextHops \ ← router.params.id
router.messages.received ∪ ← incomingMessage.id
destination.type = "nodes"
true
false
router.params.id ∈ destination.nodes
false
true
destination.type = "classes"
true
false
router.params.class ∈ destination.classes
false
true
destination.type = "regions"
true
false
destination.regions
router.location ∈ destination
false
true
processPayload()
processIncommingMessage()
router.params.role = "relay"
false
true
incomingMessage.ttl > 0
false
true
incomingMessage.visited ∪ ← router.params.id
destination.type = "nodes"
true
false
router.params.id ∈ destination.nodes
false
true
destination.nodes \ ← router.params.id
destination.nodes ≠ ∅
true
false
router.messages.stored ∪ ← incomingMessage
storeMessage()
processIncommingMessages()

Figure 5.2 Delay tolerant multicast router flowchart

device. When a new routing process starts, Procedure *initializeRouter* is called. There, an initial *router* instance gets created with empty geographic *location* of the node, lists of *received* and *stored* messages, set of *neighbors*, and *state* of the router. Then, the parameters, i.e., *class* name, unique *id*, knowledge *mode*, and network *role* are set. Afterwards, the *state* of the *router* is set to *"routing"*.

Next, looped routing process starts and proceeds until *state* of the router keeps to be set to *"routing"*. There, first, current *location* of the node is obtained and set as its latitude and longitude. Then, the body of Procedure *processBeacons* is executed only if network knowledge *mode* is set to *"local"* or *"global"*, since only these rely on radio beacons transmitted by each node. The procedure

Algorithm 8: Delay tolerant multicast router

```
1 $router \leftarrow initializeRouter()$                // initialize router (routing process) of the node
  // continue routing as long as router state is set to "routing"
2 while $router.state$ = "routing" do
3     $router.location \leftarrow getNodeLocation()$    // set current location of the node
4     $processBeacons()$
5     $forwardStoredMessages()$
6     $processIncomingMessages()$
7 $stopRouter(router)$
```

Procedure initializeRouter

```
  Output:
  $router$                                          // router instance
1 $router \leftarrow ($                             // initialize empty (start) state of the router
2     $location \leftarrow ($                       // location of the node
3         $lat \leftarrow \varnothing,$             // latitude
4         $lon \leftarrow \varnothing$              // longitude
5     $),$
6     $messages \leftarrow ($                       // message lists
7         $received \leftarrow \varnothing,$        // list of identifiers of received messages
8         $stored \leftarrow \varnothing$           // list of messages stored in the router
9     $),$
10    $neighbors \leftarrow \varnothing,$           // set of current neighbors of the node
11    $params \leftarrow ($                         // parameters of the router
12        $class \leftarrow getNodeClass(),$        // set class name
13        $id \leftarrow getNodeId(),$              // set identifier
14        $mode \leftarrow getKnowledgeMode(),$     // set network knowledge mode
15        $role \leftarrow getNodeRole()$           // set network role
16    $),$
17    $state \leftarrow$ "routing"                  // set router state to "routing"
18 $)$
19 output $router$
```

Procedure processBeacons

```
1 if $router.params.mode \in \{$"local", "global"$\}$ then
2     $processIncommingBeacons()$
3     $beacon \leftarrow makeBeacon()$
4     $transmitBeacon(beacon)$
```

starts with calling *processIncomingBeacons* designed to receive and process *incomingBeacons* to compile the set of current *neighbors*. As presented in *getCurrentNeighbors*, each *neighbor* is described with its *class*, *id*, geographic *location*, and network *role*. The important attributes of each *neighbor* are the overall *number* of its neighboring nodes and the set of *classes* of those neighbors. For each such *class*, the *name* and the *number* of neighbors that belong to this *class* are received in the beacons. These beacons are generated by each *router* following Procedure *makeBeacon* and transmitted.

Procedure processIncomingBeacons

```
1  $incomingBeacons \leftarrow receiveBeacons()$                    // receive beacons
   // determine current neighbors of the node
2  if $incomingBeacons \neq \varnothing$ then
3      $router.neighbors \leftarrow getCurrentNeighbors(incomingBeacons)$
```

Procedure getCurrentNeighbors(incomingBeacons)

```
   Input:
   $incomingBeacons$                                                  // incoming beacons
   Output:
   $neighbors \leftarrow \{neighbor_i\}_{i \leftarrow 1}^{j}$ :        // set of $j$ neighbors of current node
      $neighbor_i \leftarrow ($                                       // neighbor $i$
         $class_i$,                                                   // class
         $id_i$,                                                      // identifier
         $location_i \leftarrow ($                                    // location
            $lat_i$,                                                  // latitude
            $lon_i$                                                   // longitude
         ),
         $nodes_i \leftarrow ($                                       // classes and number of neighbors of neighbor $i$
            $classes_i \leftarrow \{class_k\}_{k \leftarrow 1}^{l}$ :  // set of $l$ neighbor classes
               $class_k \leftarrow ($                                 // class $k$ of neighbor classes
                  $name_k$,                                           // name of class $k$
                  $number_k$                                          // number of neighbors of class $k$
            ),
            $number_i$                                                // number of neighbors
         ),
         $role_i$                                                     // network role
      )
   )
   ----------------------------------------------------------------
   // determine current neighbors based on received beacons
1  foreach $beacon \in incomingBeacons$ do
2      $neighbor \leftarrow getNeighborDetails(incomingBeacon)$       // get received neighbor details
3      $neighbors \cup\leftarrow neighbor$                            // add neighbor to the set of neighbors
4  output $neighbors$
```

Procedure makeBeacon

```
   Output:
   $beacon$                                                           // beacon of current node
   ----------------------------------------------------------------
1  $beacon \leftarrow ($
2      $class \leftarrow router.params.class$,                        // set node class name
3      $id \leftarrow router.params.id$,                              // set node id
4      $location \leftarrow router.location$,                         // set node location
5      $nodes \leftarrow ($                     // set classes and number of neighbors of current node
           // group and count neighbors by their class name
6          $classes \leftarrow getNeighborClassNamesAndNumbers(router.neighbors)$,
7          $number \leftarrow |router.neighbors|$  // set the number (cardinality) of current neighbors
8      ),
9      $role \leftarrow router.params.role$                           // set node role
10 )
11 output $beacon$
```

```
Procedure forwardStoredMessages
1  if $router.params.mode \in \{$"local", "global"$\}$ then
2      if $router.neighbors = \varnothing$ then
3          break        // break forwarding procedure because currently there are no neighbor nodes
   // process each stored message
4  foreach $storedMessage \in router.messages.stored$ do
5      $outgoingMessage \leftarrow \varnothing$        // set outgoing message to an empty set
       // proceed based on knowledge mode of stored message
6      switch $storedMessage.mode$ do
           // beaconless routing
7          case "no" do
8              $outgoingMessage \leftarrow$ makeNoKnowledgeOutgoingMessage(storedMessage)
           // localized routing
9          case "local" do
10             $outgoingMessage \leftarrow$ makeLocalKnowledgeOutgoingMessage(storedMessage)
           // global knowledge routing
11         case "global" do
12             $outgoingMessage \leftarrow$ makeGlobalKnowledgeOutgoingMessage(storedMessage)
13     if $outgoingMessage \neq \varnothing$ then
14         $transmitMessage(outgoingMessage)$        // transmit message
15         $updateStorage(storedMessage)$        // update storage
```

5.3.2 Message forwarding, processing and storage

In the core phase of Algorithm 8, *delay tolerant multicast router* (DTMR), Procedure *forwardStoredMessages* is executed. If *router* operates in *"local"* or *"global"* network knowledge *mode* and there are currently no *neighbors*, then the forwarding procedure is not processed further in this iteration. In case of *"no"* network knowledge it will proceed, since this *mode* is not aware of and does not use the information about neighboring nodes. Then, for each $storedMessage$, an $outgoingMessage$ is attempted to be generated. If it gets created, then it is transmitted to be received by other nodes in the radio range. Afterwards, message storage gets updated, as defined in Procedure *updateStorage*.

The $outgoingMessage$ is prepared using one of three introduced procedures, i.e., *makeNoKnowledgeOutgoingMessage*, *makeLocalKnowledgeOutgoingMessage*, and *makeGlobalKnowledgeOutgoingMessage*, based on knowledge *mode* of $storedMessage$.

Next, Procedure *processIncomingMessages* is called, where at the very beginning *incomingMessages* are received. When there are any internal (originated) or external (forwarded) messages, each such $incommingMessage$ is processed if its knowledge *mode* is the same as knowledge *mode* the *router* operates in. Next, it is checked if this message has already been received by current node. It is done by comparing *id* of $incomingMessage$ with the list of identifiers of messages received so far by the routing process. If it has been received, it is checked if a message with this *id* is still present in message storage of current node. When it is, the *ttl* of $storedMessage$ is updated to *ttl* value of $incomingMessage$ if the latter value is lower. This situation means, that the message, after reaching current node, has been relayed in the network more, beyond current node, and hence, *ttl* should be updated to the lower value, to reflect this message dissemination progress.

Procedure processIncomingMessages

```
// receive internal (originated) and external (forwarded) incoming messages
1  $incomingMessages \leftarrow receiveMessages()$
2  if $incomingMessages \neq \varnothing$ then
       // process each incoming message
3      foreach $incomingMessage \in incomingMessages$ do
           // check if message knowledge mode is compatible with router knowledge mode
4          if $incomingMessage.mode = router.params.mode$ then
               // message has already been received by current node, based on id of this message
5              if $incomingMessage.id \in router.messages.received$ then
                   // if incoming message is present in message storage of current node
6                  if $router.messages.stored[incomingMessage.id] \neq \varnothing$ then
7                      $storedMessage \leftarrow router.messages.stored[incomingMessage.id]$
                       // update TTL of stored message if incoming is labeled with lower value
8                      if $incomingMessage.ttl < storedMessage.ttl$ then
9                          $storedMessage.ttl \leftarrow incomingMessage.ttl$
               // message reached current node for the first time
10             else
11                 if $router.params.mode \in \{$"local", "global"$\}$ then
12                     if $router.params.id \notin incomingMessage.nextHops$ then
13                         continue      // move to processing of next message because this one was not
                                          intended for current node
14                     else if $router.params.mode =$ "global" then
                           // remove current node from the set of next hops
15                         $incomingMessage.nextHops \setminus\leftarrow router.params.id$
                   // store the id of received message
16                 $router.messages.received \cup\leftarrow incomingMessage.id$
                   // process incoming message
17                 $processIncomingMessage(incomingMessage)$
                   // store the message if current node is a relay and time-to-live is not exceeded
18                 if $router.params.role =$ "relay" then
19                     if $incomingMessage.ttl > 0$ then
20                         $storeMessage(incomingMessage)$
```

If the message reached current node for the first time, different type of processing takes place. When knowledge *mode* of the router is *"local"* or *"global"*, a check is performed to verify if the *id* of current *router* belongs to the set of *nextHops* of this message. If it does not, processing has to move to the next received message, because current *incomingMessage* was not destined to this node. Otherwise, in case of *"global"* knowledge *mode*, the *id* gets removed from the set of *nextHops* since it reached desired global network knowledge next hop.

When the processing continues in current iteration, the *id* of *incomingMessage* is added to the set of unique identifiers of *received* messages. Subsequently, Procedure *processIncomingMessage* is executed. Based on the destination *type* of *incomingMessage*, it performs applicable checks to asses if the message reached current node as the member of requested multicast group. When the target is *"nodes"* and the *id* of the *router* is one of destination *nodes*, the payload of *incomingMessage* is directed to be processed. Similarly for *"classes"* when the *class* of the *router* is one of the destination *classes*. as well as, for *"regions"*. Related check is performed for *"regions"* but in this

Procedure processIncomingMessage(incomingMessage)

```
Input:
incomingMessage                                                  // incoming message
1  $payload \leftarrow \varnothing$
   // proceed based on destination type of incoming message
2  switch $incomingMessage.destination.type$ do
       // destination type set to a group of individual nodes
3      case "nodes" do
4          if $router.params.id \in incomingMessage.destination.nodes$ then
               // payload was intended for this (current) node
5              $processPayload(incomingMessage.payload)$
6      case "classes" do
7          if $router.params.class \in incomingMessage.destination.classes$ then
               // payload was intended for this node class
8              $processPayload(incomingMessage.payload)$
       // destination type set to regions
9      case "regions" do
10         foreach $destination \in incomingMessage.destination.regions$ do
11             if $router.location \in destination$ then
                   // payload was intended for this destination region
12                 $processPayload(incomingMessage.payload)$
                   // break the loop because the node belongs to at least one destination region
13                 break
```

Procedure storeMessage(incomingMessage)

```
Input:
incomingMessage                                                  // incoming message to be stored
1  $incomingMessage.visited \cup\leftarrow router.params.id$   // add current node to the list of visited nodes
2  if $incomingMessage.destination.type$ = "nodes" then
       // if this node is one of destination nodes
3      if $router.params.id \in incomingMessage.destination.nodes$ then
           // remove current node from the set of destination nodes
4          $incomingMessage.destination.nodes \setminus\leftarrow router.params.id$
           // do not store the massage if it just reached the last desired destination node
5          if $incomingMessage.destination.nodes = \varnothing$ then
6              $incomingMessage \leftarrow \varnothing$
7  if $incomingMessage \neq \varnothing$ then
8      $router.messages.stored \cup\leftarrow incomingMessage$           // store the message
```

case each *destination* region of *incomingMessage* is checked one by one to test if current *locatin* of the *router* is located within multicast destination region. Since there may be defined numerous regions and some could be overlapping, the payload is processed and further checks are stopped for current *incomingMessage* when the first matching *destination* region is found.

In the final stage, if current *router* is an actual *"relay"* and the *ttl* of *incomingMessage* is greater than zero, it means that the message must be stored for further forwarding. Procedure *storeMessage* first adds the *id* of the *router* to the list of nodes *visited* by *incomingMessage*. Next, if destination type is set to *"nodes"*, the *id* of current node (router) is removed from the set of destination nodes of this message. If, as a result, the list of destination *nodes* becomes empty, it means that the message has just reached the last requested destination node. Therefore, *incomingMessage* cannot be stored for further forwarding. Otherwise, the procedure concludes with adding *incomingMessage* to the list of *stored* messages.

5.4 Observations

Introduced multicast algorithms are aimed at heterogeneous urban sensor networks using uncontrolled mobile relays. They are based on the availability of no, local, or global network knowledge. These opportunistic algorithms are intended to be operating with various destination group types, node classes and roles. Three supported knowledge modes and three target group types result in nine different algorithm flows. Although, they are designed in a modular and reusable way. The algorithms are presented as pseudocodes in the form inspired by set theory notation and JSON-structure. Flowcharts related to multicast message forwarding, as well as, to delay tolerant multicast router are presented in a protocol agnostic way.

Unlike in typical greedy geographic routing strategies, not only the relay or relays that provide geographical progress towards the destinations of known location are selected as next hops. Additionally, primary and secondary relay nodes are involved, when necessary. The first one is the relay that appears to maximize the likelihood of reaching multiple destinations. The second one distributes the message in a possibly opposite geographical direction. This, together with network topology dynamics, is a method for local optima avoidance.

Since presented routing mechanisms are designed for delay-tolerant networks, a message is not only simply forwarded from node to node but it also requires to be maintained in local buffers, when carried over space and time. This includes processing and updating of message metadata to decide if it ought to be further stored, forwarded, or discarded. Moreover, local and global knowledge modes require more computational resources, and yet, they are less transmission-intensive than no knowledge multicasting. Therefore, to achieve optimal network usage and desired delivery rate, all related aspects and parameters have to be thoroughly considered and carefully planned.

Chapter 6

Simulation study of introduced models and algorithms

This chapter is dedicated the multi-dimensional simulation-based analysis of network modeling algorithms introduced in Section 4.3 and urban delay tolerant multicast algorithms presented in Section 5. First, simulation and analysis methodology is described. Then, space connectivity is analyzed and related space-time connectivity is investigated. Next, simulated multicast algorithms are analyzed and the observations are summarized.

6.1 Simulation and analysis methodology

The main objective of the simulation study is to enable multi-criteria evaluation and comparison of proposed models and algorithms. Due to unknown characteristics of underlying urban infrastructure, numerous features of constructed networks are also of interest.

Network modeling flow of the simulation follows logical order in which algorithms introduced in Section 4.3 and 5.2 are related to one another. The high-level steps of the simulation are depicted in Figure 6.1. Simulation environment was built as the extension of custom-made network modeling software described in Section 4.2. In this way, the efficacy and efficiency of the algorithms can be investigated and compared using graph theory methods. Key features and metrics of modeled networks can be thoroughly analyzed as well.

Multicast structures are constructed and analyzed as graphs. In this approach, the existence of a particular graph edge indicates that multicast message reached given node. No actual radio propagation models, communication protocols and power management mechanisms are simulated. By this means, technology and protocol agnosticism is ensured in all aspects.

Comparative study methodology

There exist no equivalent algorithms designed for modeling heterogeneous urban sensor networks. Therefore, the trends of metrics of interest are compared and analyzed in relation to different urban areas, topology durations, radio ranges, multicast sources, destination node numbers, and network knowledge modes. Space minimum and maximum spanning forests are constructed for each

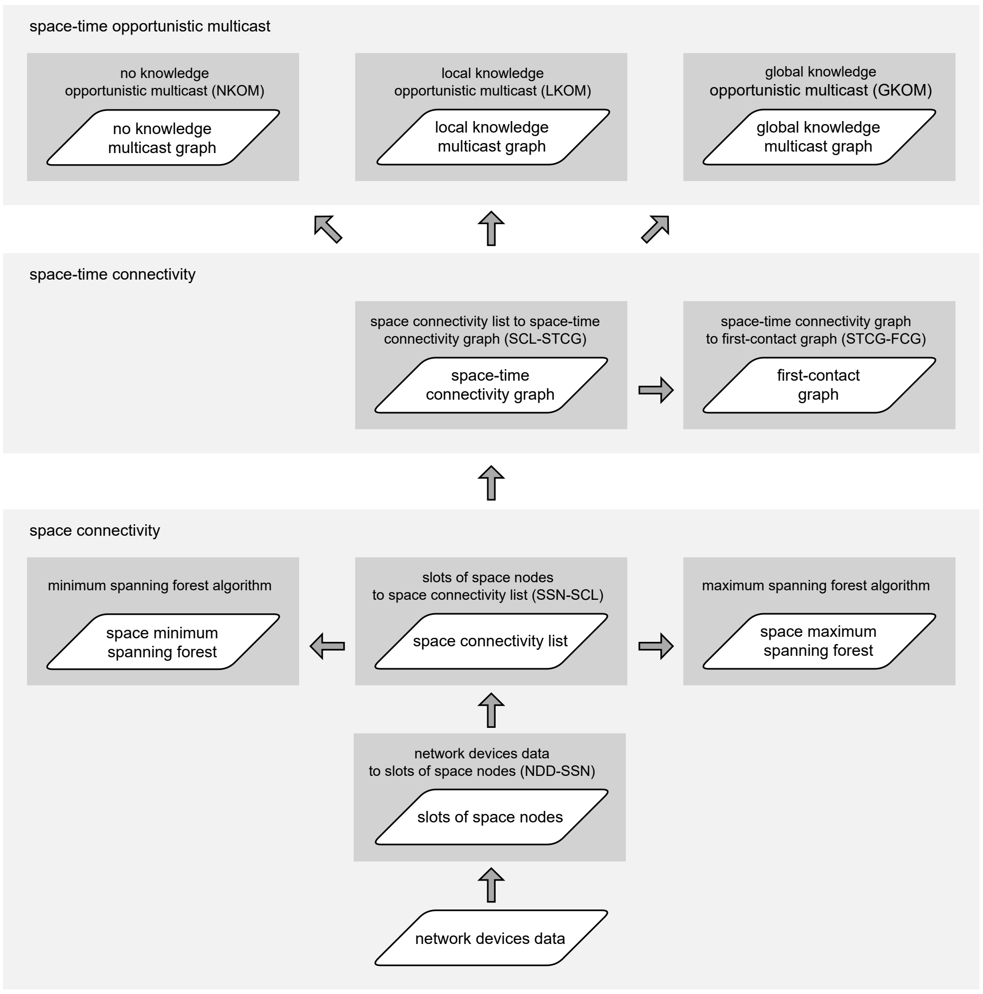

space-time opportunistic multicast
no knowledge opportunistic multicast (NKOM)
no knowledge multicast graph
local knowledge opportunistic multicast (LKOM)
local knowledge multicast graph
global knowledge opportunistic multicast (GKOM)
global knowledge multicast graph
space-time connectivity
space connectivity list to space-time connectivity graph (SCL-STCG)
space-time connectivity graph
space-time connectivity graph to first-contact graph (STCG-FCG)
first-contact graph
space connectivity
minimum spanning forest algorithm
space minimum spanning forest
slots of space nodes to space connectivity list (SSN-SCL)
space connectivity list
maximum spanning forest algorithm
space maximum spanning forest
network devices data to slots of space nodes (NDD-SSN)
slots of space nodes
network devices data

Figure 6.1 Simulation modeling flow

space connectivity graph undirected analog using Kruskal's algorithm [177]. These forests are the generalized solutions of the problem of *time-sub-interval minimum spanning tree* (TSMST) in a spatiotemporal network defined in [192]. Key metrics of the forests are compared to provide more insight into momentary space connectivity topologies. Those forests are the graphs composed of sets of trees built for every connected component of a space connectivity graph. The most informative node-related metrics of constructed space-time connectivity graphs are compared to the related first-contact graphs. Further FCG-related parameters are gathered and investigated as well.

Introduced multicast algorithms are compared with one another. For general metrics, graphs constructed with no network knowledge are the baseline of the comparisons. It is dictated by the fact that no network knowledge approach is the simplest one. Local knowledge multicast is related but more sophisticated. In both cases, multicast graphs are expected to be composed of both core and stub multicast nodes and edges since some paths do not lead to any multicast destination node. Conversely, each global knowledge solution is constructed as a *shortest paths tree* (SPT) that connects multicast source, with time shortest paths, only to desired destinations. Therefore, for core multicast structure metrics global knowledge graphs become the baseline. The spatiotemporal shortest paths are determined with Dijkstra's algorithm [183] applied to each space-time connectivity graph with source and destination nodes because their identifiers and locations are known in global knowledge mode in every destination type case. Then, the paths are merged and potential cycles are removed to build multicast tree. Space-time connectivity graphs are directed which is caused both by modeled arrow of time and by different roles of different classes of the nodes. Multicast tree could be alternatively modeled as *directed Steiner tree* (DST) which minimizes the overall cost of the tree but there exists no known exact algorithm to solve this problem [193], and hence, to be used to model optimal baseline solution.

Statistical analysis and visualization

Statistical data are generated and gathered in the processes related to each step of network modeling. Some metrics are calculated using custom developed functions while other are computed with the methods provided by NetworkX graph modeling framework. The data are then represented as pandas data structure called DataFrame [194]. In this way, advanced multi-dimensional data combining, filtering, categorization and statistical processing is performed. Although, the values of individual data points and their numerical aggregates are not the center of attention of this the study. The trends and relationships between the parameters of the networks and their metrics are the ones of key concern. Therefore, analyzed data are visualized with Seaborn data visualization library [195].

The data of interest is of discrete nature, and therefore, analyzed sets are presented in the form of scatter plots. The styles of the points represent different subsets of the data. Regression lines are overlayed on the plots to make the trends more visible in larger, denser, and overlapping data sets of a single chart, as in Figure 6.13. The sets of closely related subplots are grouped in named rows or columns of a single plot, e.g., the charts related to different graph type and radio range in Figure 6.18. Furthermore, pair plots are used to present the relationships between the sets of variables, as in Figure 6.21. A number of plots are categorical to group and shift the data horizontally around the values of interest which makes the categories more distinguishable, such as city or knowledge mode. This kind of plot also introduces small jitter, i.e., random deviations, to horizontal distributions of the categories to make them more visible when there are multiple close related values present. It does not change the values, i.e., the vertical distribution of the measurements, as visible in Figure 6.29. Due to multi-dimensional nature of the data the figures related to the subsets of metrics are also grouped and discussed in dedicated sections, e.g., in Section 6.2.1 focused on space connectivity nodes parameters.

6.1.1 Simulation data sources and node classes

Investigation and analysis of data sources discussed in Section 4.1 lead to the conclusion that only in case of four Polish cities there exist sets of open data sources which meet the requirements of the study. Other cities provide limited scope or do not provide similar open data at all. The author did not succeed in discovering equivalent open sets related to urban areas in other countries either. Therefore, Gdańsk, Poznań, Warsaw, and Wrocław sources listed in Table 6.1 are used—being the ones that provide the data of comparable scope, granularity, and update frequency.

Table 6.1 Urban open data sources used in simulation study

city	class	scope	format	updates	provider
Gdańsk	mobile advanced	buses and trams [137]	JSON	20 s	Open Gdańsk
	stationary simple	public transport stops [154]	JSON	24 h	Open Gdańsk
	stationary advanced	ticket machines [150]	JSON	24 h	Open Gdańsk
Poznań	mobile advanced	buses and trams [138]	protobuf	continuous	ZTM Poznań
	stationary simple	public transport stops [196]	JSON	infrequent	Poznan City Hall
	stationary advanced	ticket machines [197]	JSON	infrequent	Poznan City Hall
Warsaw	mobile advanced	buses and trams [152]	JSON	10 s	City of Warsaw
	stationary simple	public transport stops [122]	JSON	infrequent	City of Warsaw
Wrocław	mobile advanced	buses and trams [198]	JSON	continuous	Open Data Wrocław
	stationary simple	city bike rental stations [144]	JSON	5 min	Open Data Wrocław
	stationary simple	Vozilla parking lots [199]	JSON	continuous	Open Data Wrocław
all	stationary advanced	air quality meters [117]	JSON	continuous	Airly

Geographic coordinates of urban infrastructure elements extracted from the data are used in presented comparative research and transformed with introduced network modeling algorithms. In all four cities the real-time locations of buses and trams are available. In Gdańsk and Poznań the locations of public transport stops and ticket machines can be used. In Warsaw and Wrocław ticket machines location data are not available. Although, in case of Wrocław the coordinates of city bike rental stations (as the one in Figure 2.3c) and parking lots of Vozilla (city electric car rental service) can be used instead. Most of the data are available in JSON format while Poznań-related mobile nodes data in *Protocol Buffers* (protobuf). Update frequencies of those range from a few seconds in case of the continuously updated ones to 10 and 20 seconds for Warsaw and Gdańsk, respectively. The numbers and locations of stationary nodes do not change that frequently. This means that even when the source updates the whole data set frequently, e.g., for air quality meters and city bike rental stations, the location-related data can remain unchanged for hours or days, just like for each 24 hours of more infrequently updated ones. To enable heterogeneous structure modeling, data sources were classified into three meaningful logical classes—mobile advanced, stationary advanced, and stationary simple. The nodes of each class are assigned a network role in connectivity and multicast modeling scenarios:

- mobile advanced class ⇒ mobile relays;
- stationary advanced class ⇒ stationary relays;
- stationary simple class ⇒ stationary destinations.

It is being assumed that advanced nodes are the nodes of more significant computing, storage, communications and power resources. Therefore, they are capable of performing complex DTN forwarding operations. In contrast, simple nodes are the simple recipients of the communication.

6.1.2 Simulation areas and example modeled networks

The four cities of interest—Gdańsk, Poznań, Warsaw, and Wrocław—are among the largest and the most populated ones in Poland, as presented in the next paragraphs. Warsaw is the most populated urban area, which is more than two and a half times the population of Wrocław. Wrocław is twenty three percent more populated than Poznań, while the population of Poznań is thirteen percent larger than the one of Gdańsk. Population densities also differ in a related way—Gdańsk is the least densely populated urban area, followed by Poznań, Wrocław, and Warsaw which is almost two times more densely populated than Gdańsk. Interestingly, in terms of the expected numbers of public transport routes (lines) that operate during the day, Poznań is the city with the lowest number, followed by slightly more routes in Gdańsk and Wrocław. In Warsaw, twice as many routes are present on average.

An area of 3 by 2 kilometers was selected in each of the cities. The choice is aimed at covering partially alike and partially distinct regions that include both the busy heart, as well as, less dense surroundings of each urban area. A closer examination of city topologies visible in presented figures reveals unique terrain, building, street, and infrastructure layouts. Therefore, the modeled networks are expected to indicate both different and similar features.

Example presented graphs constructed in those areas show the state modeled on Wednesday, 27 November 2019, at 3:00 p.m. when omnidirectional radio coverage is assumed. The relays are dark blue triangles, destination nodes are pink circles and multicast source is a red hexagon. Mobile nodes are the ones with black border. Each node is presented in the location it occurred for the first time. Destination regions are marked as red dashed rectangles. Solid link depicts a space connection, i.e., the one that occurs without message buffering (in the same time interval). Dotted are multicast core space-time connections, i.e., the ones that lead to the destinations and require message buffering in the relay. Dashed space links and dash-dot space-time links do not lead to destination nodes. Label *30 (24 s, 21 m)* in Figure 6.8 indicates that the edge exists in slot number 30, the message has to be buffered for 24 slots in the relay before being forwarded and that space distance between the relay and the next hop node is 21 m.

Gdańsk

- Population:
  - Total: 486 thousand [200] in the metropolis of around one million in northern Poland;
  - Density: 1797 per km^2 [201];
- Public transport day routes: around 80 [202];
- Simulation area:
  - Latitude: 54.34398 – 54.36191;
  - Longitude: 18.62036 – 18.66666;
- Example space connectivity graph in Figure 6.2:
  - Slot length: 6 s;
  - Radio range: 100 m;
  - Nodes: 121 — mobile relays: 9, stationary relays: 20, stationary destinations: 92;
  - Average node degree: 2.45, edges: 148, space cost: 6177 m, connected components: 66;

- Example no network knowledge multicast graph to destination nodes in Figure 6.3:
  - Duration: 30 min — Number of slots: 300;
  - Slot length: 6 s;
  - Radio range: 100 m;
  - Nodes: 144 — mobile relays: 114, stationary relays: 20, stationary destinations: 10;
  - Average node degree: 1.99, edges: 143, space cost: 7716 m, time cost: 8073, time span cost: 4944, core nodes: 22.

Poznań

- Population:
  - Total: 547 thousand [176] in the metropolis of almost one million in west-central Poland;
  - Density: 2031 per km^2 [201];
- Public transport day routes: around 70 [203];
- Simulation area:
  - Latitude: 52.39853 – 52.41645;
  - Longitude: 16.88965 – 16.93389;
- Example space minimum spanning forest in Figure 6.4:
  - Time interval: 6 s;
  - Radio range: 100 m;
  - Nodes: 171 — mobile relays: 38, stationary relays: 17, stationary destinations: 116;
  - Average node degree: 1.20, edges: 103, space cost: 4904, connected components: 68;
- Example local knowledge multicast graph to destination classes in Figure 6.5:
  - Duration: 30 min — Number of slots: 300;
  - Slot length: 6 s;
  - Radio range: 100 m;
  - Nodes: 303 — mobile relays: 173, stationary relays: 15, stationary destinations: 115;
  - Average node degree: 1.99, edges: 302, space cost: 19406 m, time cost: 16319, time span cost: 7594, core nodes: 150.

Warsaw

- Population:
  - Total: 1.794 million [204] in the metropolis of 3 million in east-central Poland;
  - Density: 3469 per km^2 [201];
- Public transport day routes: around 190 [205];
- Simulation area:

  - Latitude: 52.22082 – 52.23879;
  - Longitude: 20.97058 – 21.01454;
- Example space maximum spanning forest in Figure 6.6:
  - Slot length: 6 s;
  - Radio range: 100 m;
  - Nodes: 213 — mobile relays: 49, stationary relays: 4, stationary destinations: 160;
  - Average node degree: 1.10, edges: 117, space cost: 8199 m, connected components: 96;
- Example local knowledge multicast graph to destination regions in Figure 6.7:
  - Duration: 30 min — Number of slots: 300;
  - Slot length: 6 s;
  - Radio range: 100 m;
  - Nodes: 358 — mobile relays: 310, stationary relays: 1, stationary destinations: 47;
  - Average node degree: 1.99, edges: 357, space cost: 21391 m, time cost: 17580, time span cost: 11599, core nodes: 80.

Wrocław

- Population:
  - Total: 674 thousand [206] in the metropolis of around 1.25 million in southwestern Poland;
  - Density: 2192 per km^2 [201];
- Public transport day routes: around 85 [207];
- Simulation area:
  - Latitude: 51.10015 – 51.11813;
  - Longitude: 17.01273 – 17.05570;
- Example first-contact graph in Figure 6.8:
  - Slot length: 6 s;
  - Radio range: 25 m;
  - Nodes: 217 — mobile relays: 145, stationary relays: 2, stationary destinations: 70;
  - Average node degree: 1.66, edges: 180, space cost: 3089 m, time cost: 4992, connected components: 120;
- Example global knowledge multicast graph to destination regions in Figure 6.9:
  - Duration: 30 min — Number of slots: 300;
  - Slot length: 6 s;
  - Radio range: 100 m;
  - Nodes: 43 — mobile relays: 27, stationary relays: 0, stationary destinations: 16;
  - Average node degree: 1.95, edges: 42, space cost: 2411 m, time cost: 825, time span cost: 641, core nodes: 43.

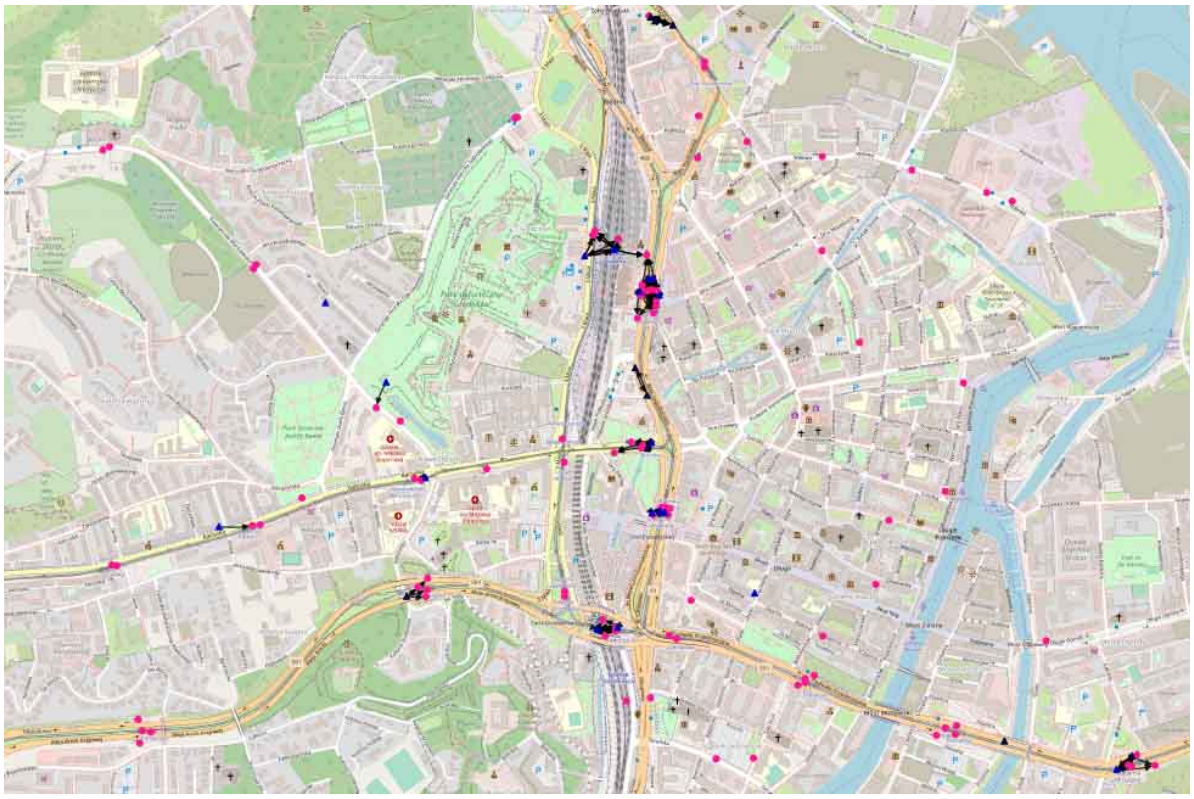


Figure 6.2 Example space connectivity graph in Gdańsk

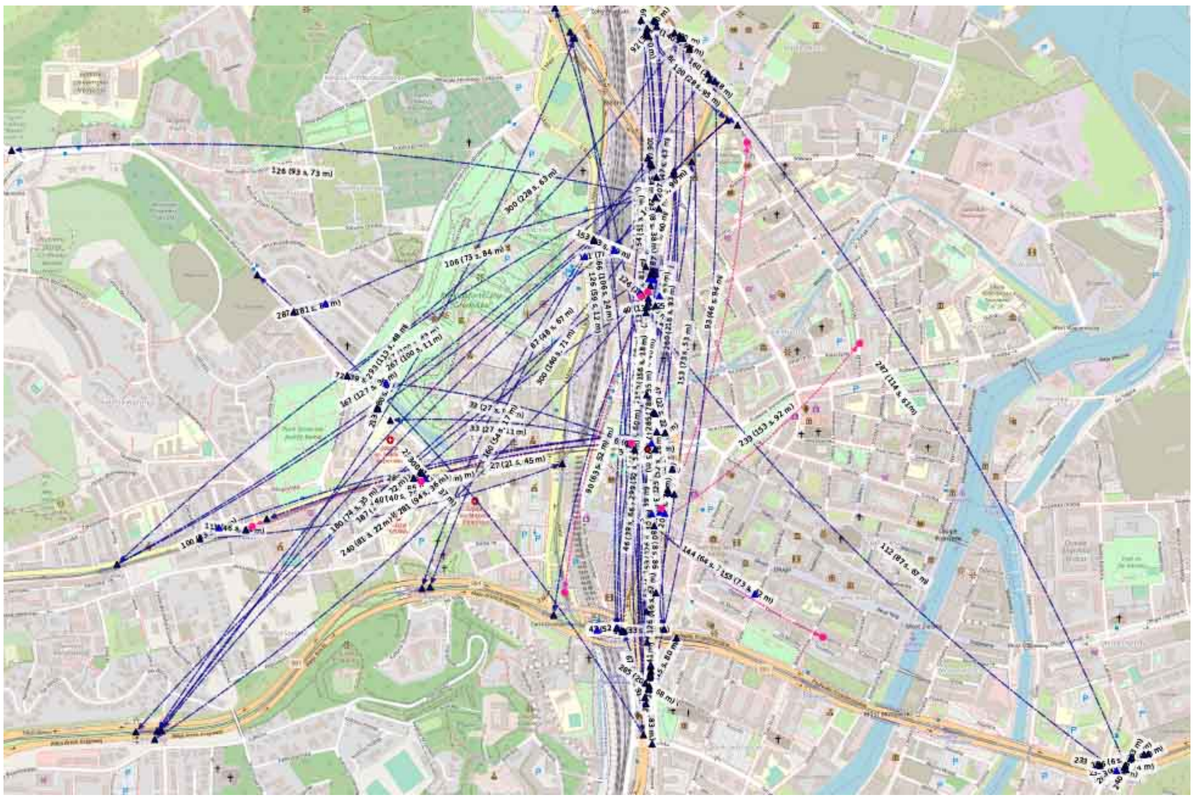


Figure 6.3 Example no network knowledge multicast graph to destination nodes in Gdańsk

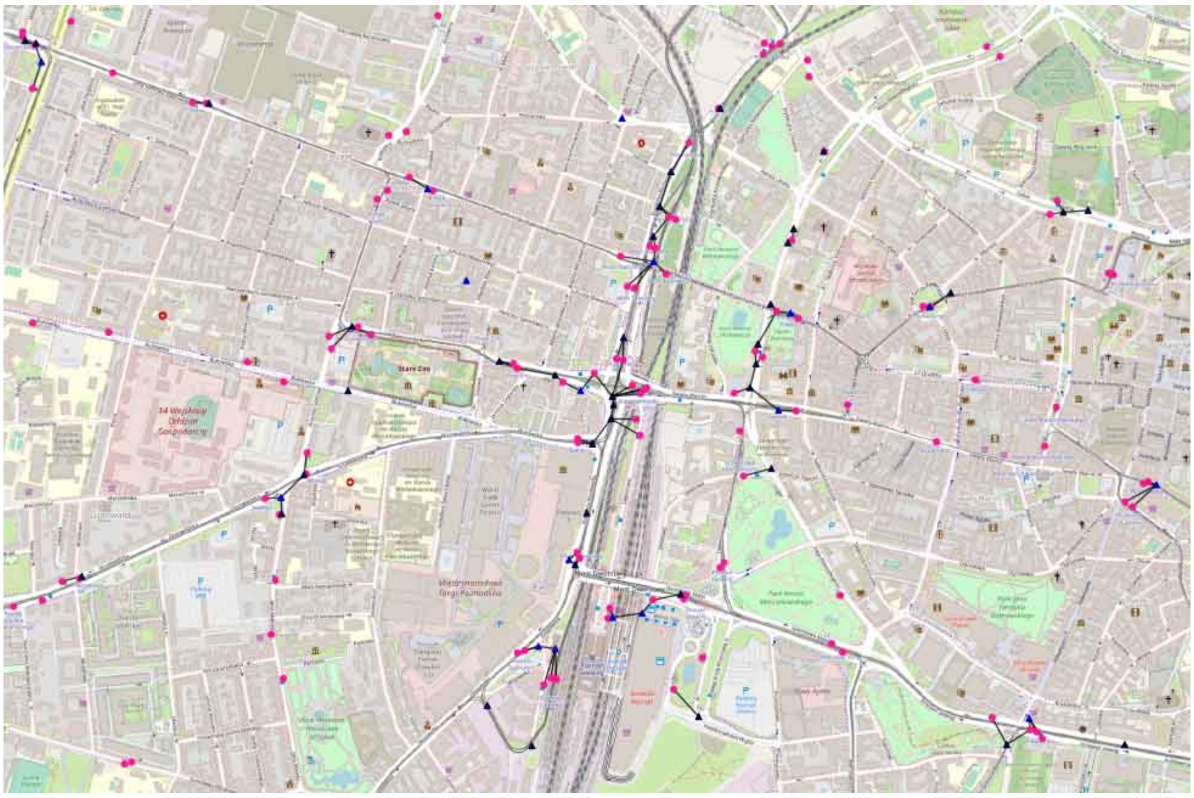


Figure 6.4 Example space minimum spanning forest in Poznań

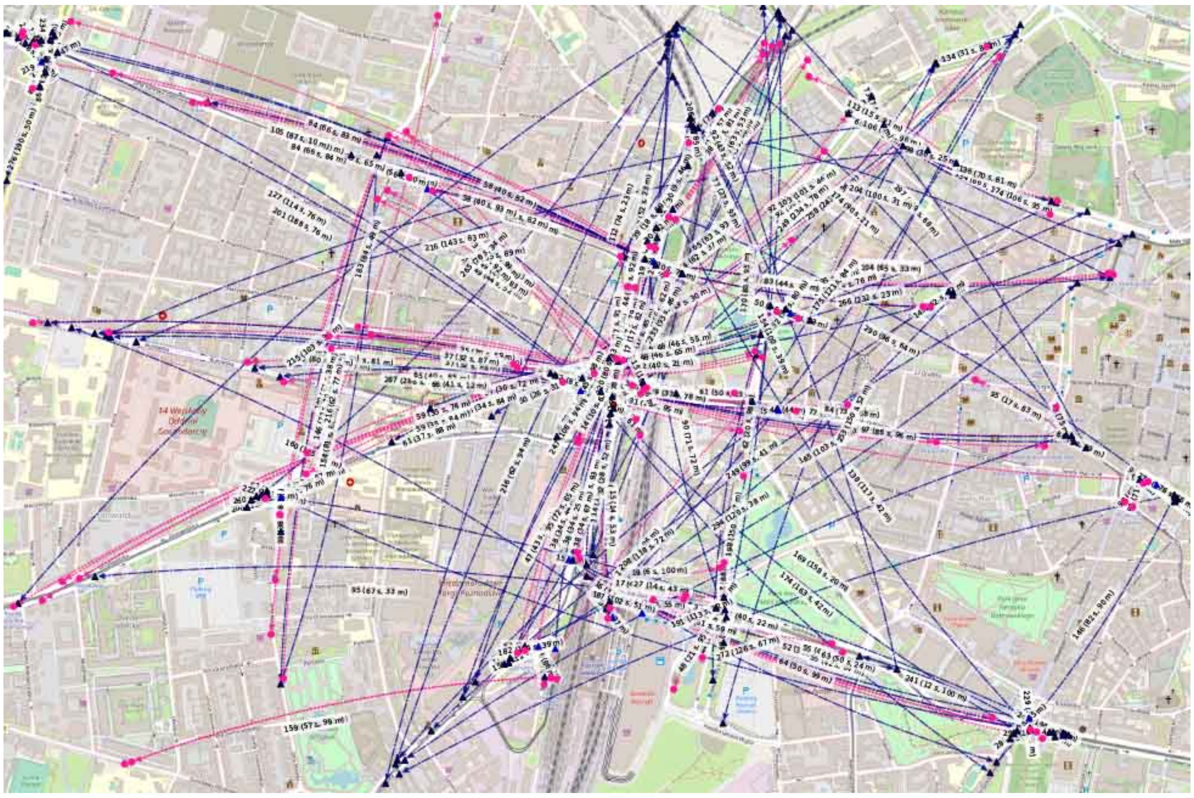


Figure 6.5 Example multicast to destination classes with local network knowledge in Poznań

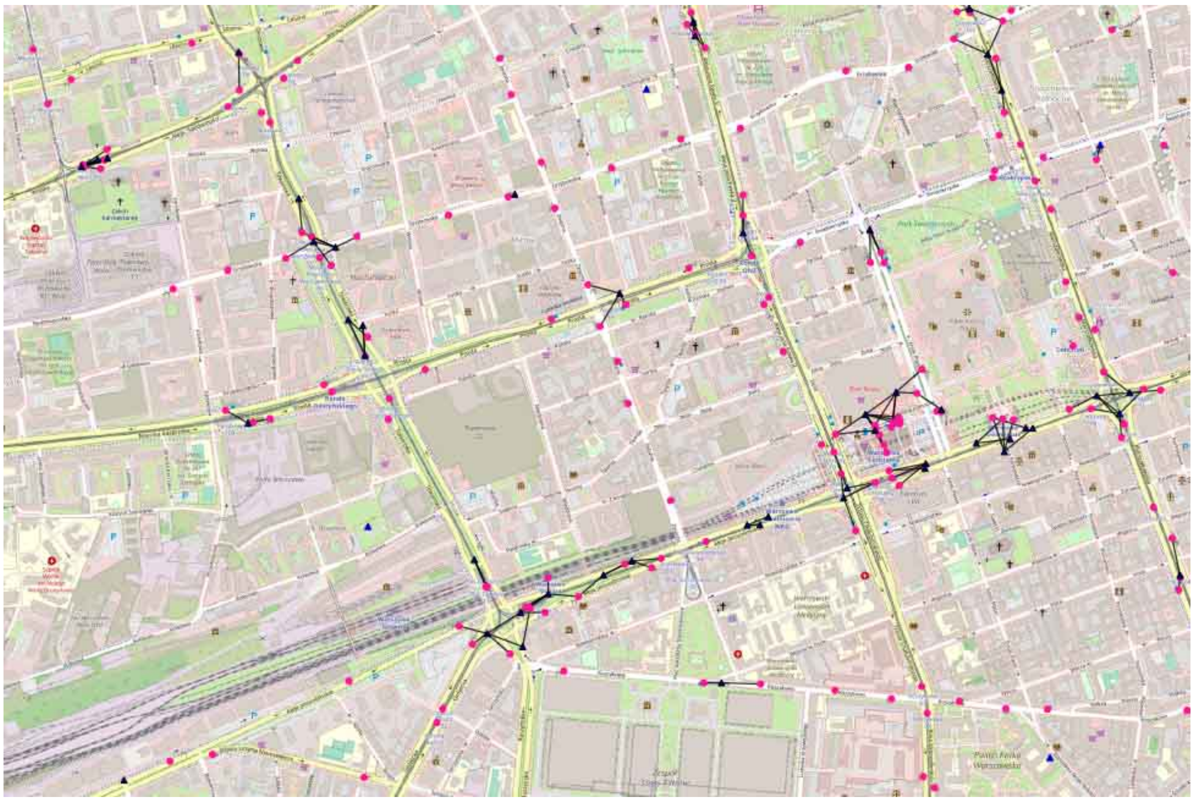


Figure 6.6 Example space maximum spanning forest in Warsaw

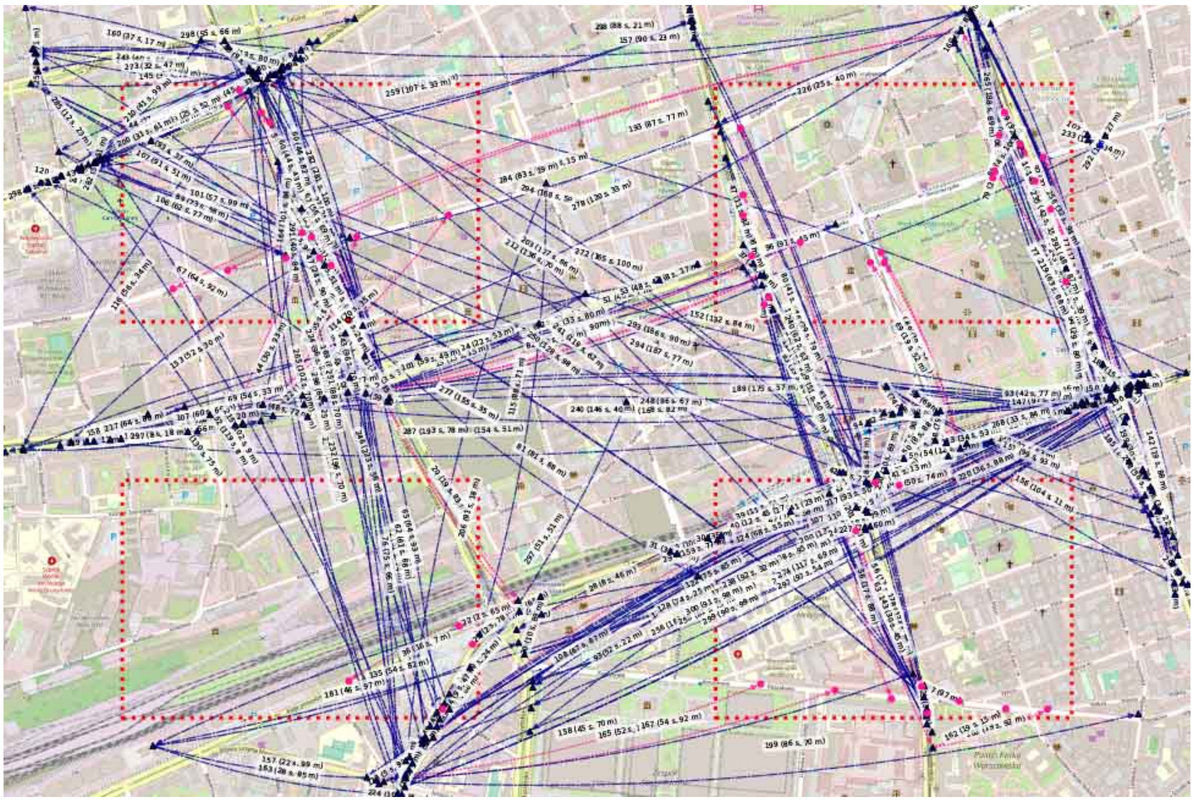


Figure 6.7 Example multicast to destination regions with local network knowledge in Warsaw

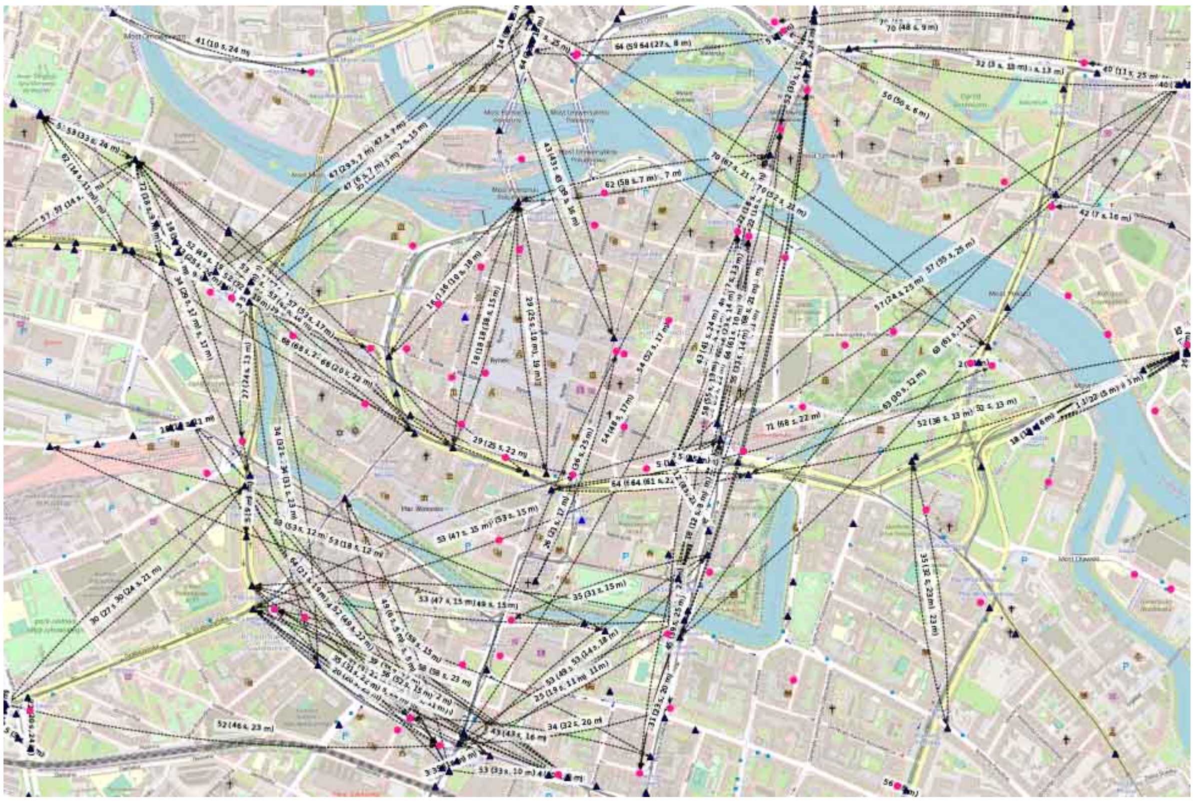


Figure 6.8 Example first-contact graph in Wrocław

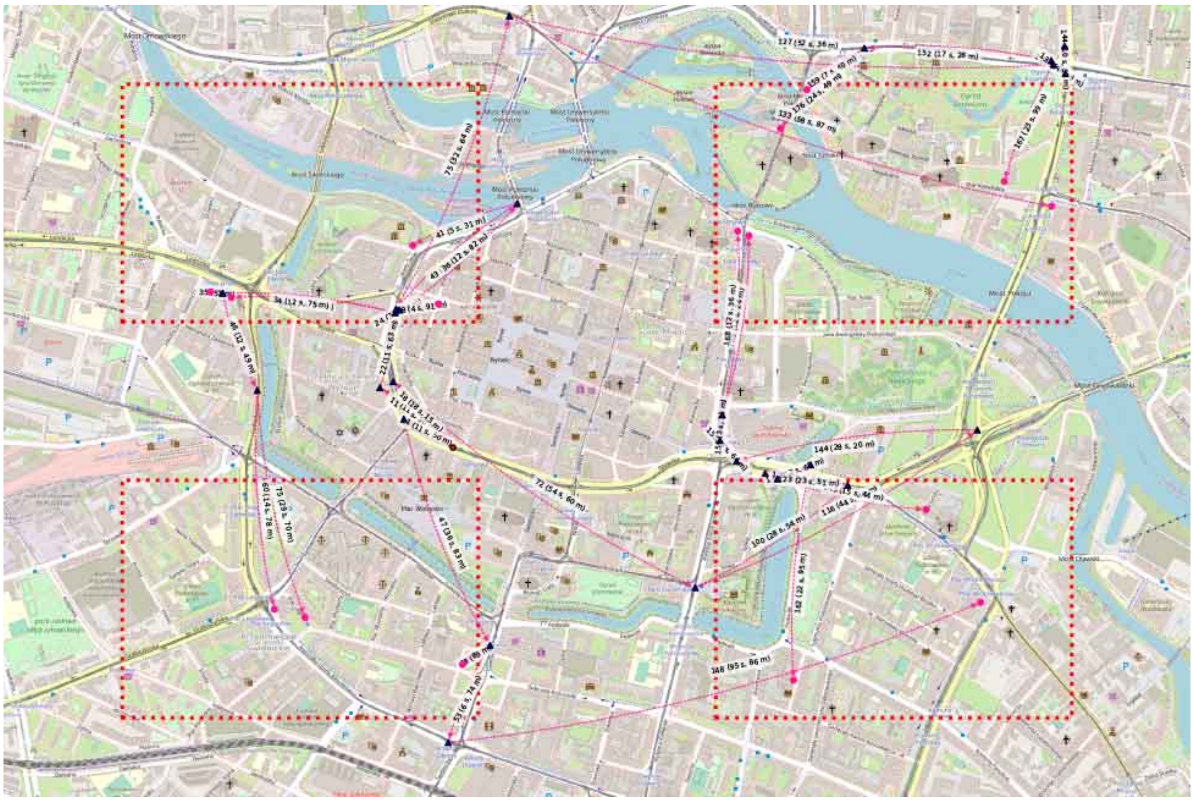


Figure 6.9 Example multicast to destination regions with global network knowledge in Wrocław

6.1.3 Simulation architecture and parameters

To enable multi-faceted modeling and analysis, simulation architecture is based on object-oriented data structures implemented as a hierarchy of nested lists presented in Figure 6.10. The simulations are executed with the parameters related to the algorithms which are the main steps in modeling flow depicted in Figure 6.1. The key simulation scope characteristics (numbers), denoted with single capital letters, resulting from the architecture and parameters are also indicated:

- Algorithm 1, *network devices data to slots of space nodes* (NDD-SSN):
  - period: 27 November 2019 from 3:00 p.m. to 5:00 p.m.;
  - areas: 4 $\Rightarrow J = 4$;
    - $area_1$: $([54.34398, 54.36191], [18.62036, 18.66666])$ — Gdańsk;
    - $area_2$: $([52.39853, 52.41645], [16.88965, 16.93389])$ — Poznań;
    - $area_2$: $([52.22082, 52.23879], [20.97058, 21.01454])$ — Warsaw;
    - $area_2$: $([51.10015, 51.11813], [17.01273, 17.05570])$ — Wrocław;
  - topology lengths: (75, 150, 300, 600, 1200);
    - durations: (7.5 min, 15 min, 30 min, 60 min, 120 min) $\Rightarrow L = 5$;
    - topologies: (16, 8, 4, 2, 1) $\Rightarrow N = 31$;
  - slot length: 6 s $\Rightarrow S = 1200$;
  - classes: (mobile advanced, stationary simple, stationary advanced);
  - windows: (10 s, 24 h, 24 h);
  - relays: (mobile advanced, stationary advanced);
- Algorithm 2, *slots of space nodes to space connectivity list* (SSN-SCL):
  - radio coverage: omnidirectional;
    - radio ranges: (25 m, 50 m, 100 m) $\Rightarrow Q = 3$;
    - space distance: great-circle distance between two nodes;
- Algorithm 3, *space connectivity list to space-time connectivity graph* (SCL-STCG):
  - unit cost:
    - intra-slot time edge space unit cost: 0;
    - inter-slot time edge space unit cost: 0;
    - intra-slot space edge time unit cost: 0;
    - intra-slot time edge time unit cost: 0;
    - inter-slot time edge time unit cost: 1;
- Algorithm 8, *delay tolerant multicast router* (DTMR):
  - multicast sources: 5 $\Rightarrow U = 5$;
    - class: mobile;
  - network knowledge mode: (no, local, global) $\Rightarrow W = 3$;
  - destination types: $\Rightarrow Y = 3$;
    - nodes: 10;

- classes: stationary simple;
- regions: 4;
    - shape: rectangle;
    - width: 30% of simulation area width;
    - height: 30% of simulation area height;
    - separation: 10% of respective simulation area dimension.

Node data sources for the simulation are grouped into three classes—mobile advanced, stationary simple, and stationary advanced, as introduced in Table 6.1. With each of the classes a data-lookup window is related. The widths of these were determined based on the analysis of update frequencies of the sources. The data on mobile nodes are the ones that get updated the most frequently, i.e., as often as each few seconds. Therefore, a window of 10 seconds ensures that location changes will be timely reflected in the modeled structures. Each location of a node is marked with occurrence time. When slot length is set to 6 seconds, the window of 10 seconds is also the means to correct brief node data or source outages—to avoid the node missing in a single space connectivity graph when actually it was still present in the network. Longer window, in case of rapidly moving nodes and frequent data outages, may lead to misrepresentation of the node in its previous known location. Hence, the connections may appear which, in reality, would not be possible to get established at that point in time since the node was, in fact, already at a different location. Also, the node could be able to established links that were not modeled when the data was not available. The observations of stationary nodes data lead to the conclusion that their location changes or is updated not more frequently than once every 24 hours. Therefore, this interval is used as the window for fixed nodes-related location data.

The simulations were conducted in four urban areas of interest based on the data gathered on Wednesday, 27 November 2019 between 3:00 p.m. and 5:00 p.m. This 2 hours period was selected because it includes afternoon rush hours in the middle of a work week and enables to cover all desired modeling and analysis scenarios. In the modeling process, the period is divided in Algorithm 1 into topologies of duration defined by *topologyLength*. The values of interest are 75, 150, 300, 600, and 1200 which are the subset of a geometric sequence with common ratio 2. These are the numbers of space connectivity graphs in the space connectivity list of given topology. They allow to model and study network structures that are related, and yet, of different properties and time span. Therefore, the trends connected to the matters of scalability and optimization can be investigated.

When *slotLength* is set to 6 seconds, series of topologies of 16, 8, 4, 2, and 1 space-time connectivity graphs are distinguished that last 7.5, 15, 30, 60, and 120 minutes, respectively. Time of *slotLength* is expected to be sufficient to transmit the message between two neighboring nodes. Unlimited message storage (buffer) is assumed in each relay. Omnidirectional radio coverage is modeled for three effective radio ranges, i.e., 25 m, 50 m, and 100 m. These range limits are based on empirical observations that current popular sensing-related short-range wireless connectivity technologies tend to provide up to around 100 m range at higher throughputs in outdoor urban non-line-of-sight scenarios, depending on transmit power [208] and data transmission parameters [209]. The space distance between two nodes is determined with the haversine formula, which computes the great-circle distance between

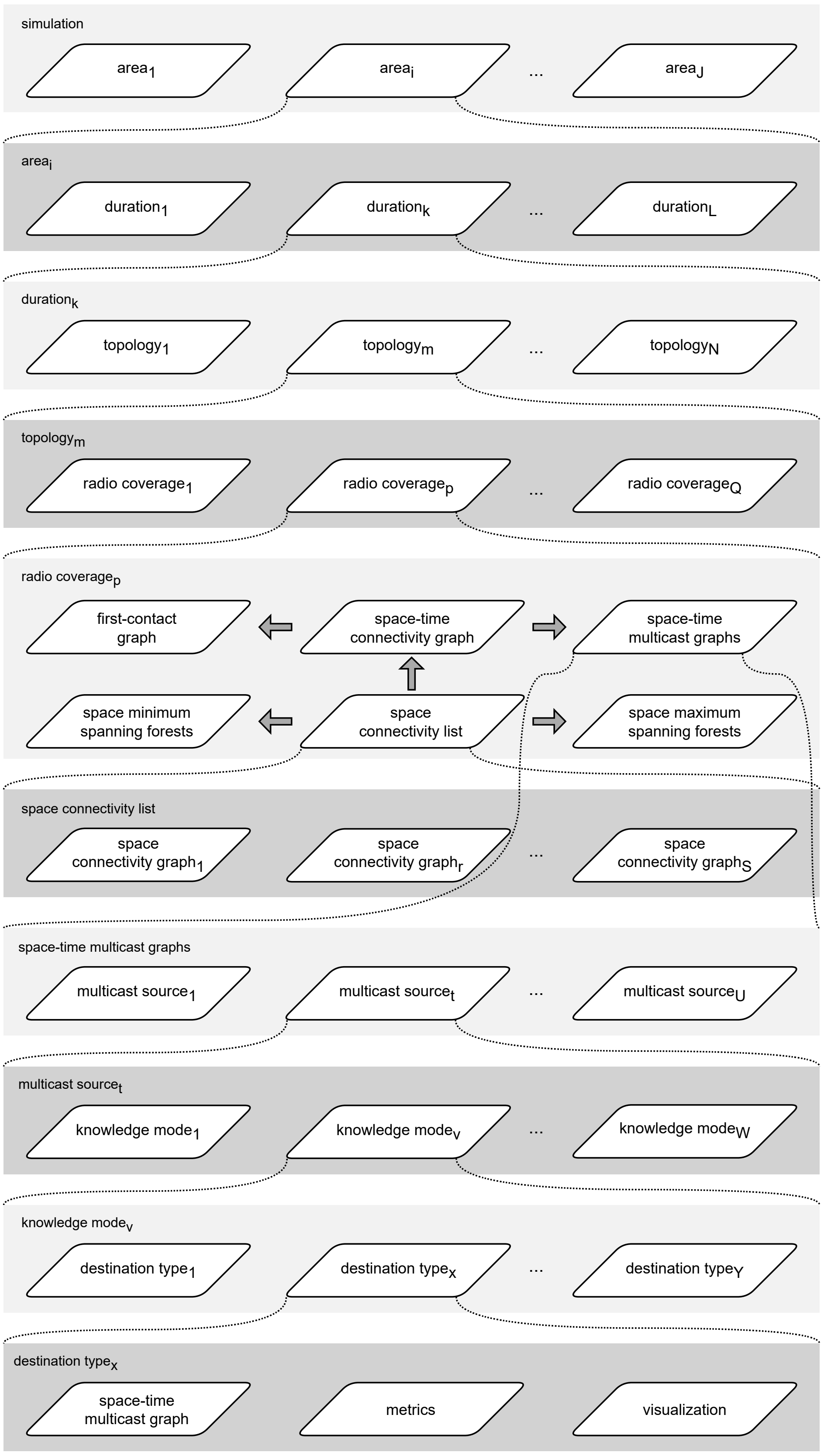


Figure 6.10 Simulation architecture

two points on a sphere [171]. For each of the resulting graphs minimum and maximum spanning forests are constructed.

Then, space-time connectivity graphs (networks) are modeled based on each of 31 space connectivity lists. Default values of unit costs of Algorithm 3, SCL-STCG, are used. By this means, correct time-shortest paths can be determined with Dijkstra's algorithm based on time distance weight of the edges. The process set up in this way aims to balance buffering time with message forwarding, and therefore, buffering resources use with transmission-related power consumption of the relays. Related first-contact graphs are constructed as well. Time distance in those graphs means how much time, i.e., time slots, has to pass before the node will get close enough to the neighboring node (device) to establish a connection.

Multicast graphs are modeled for a single message originated by a source. Their durations are limited by setting *time-to-live* (TTL) of the message to respective *topologyLength* value. Before constructing multicast structures in given space-time connectivity topology, a source node is randomly selected from among its mobile relay nodes. Stationary relays could also be used but mobile sources introduce more real-life dynamics into modeled structures. This process is based on the use of uniform random number generator and repeated to select 5 sources in the first graph of space connectivity list which has any mobile relays. If there are fewer relays present, their number limits the number of selected sources, i.e., every mobile relay becomes the source node in one of network variants. Each source and related space-time connectivity graph become the input structure for constructing multicast graphs. Additionally, up to 10 stationary destination nodes are randomly selected in the same space connectivity graph in which the source was selected. When fewer stationary simple nodes are present in the graph then as many as possible are designated destinations. Only stationary nodes are considered as destination candidates since their location does not change over time and can be known when recorded at the time of their deployment. In this way, they can be addressed in destination regions (geocast) scenario when no other network knowledge is available.

The nodes are categorized routing-wise as multicast source, relays and destinations. Mobile advanced, as well as, stationary advanced nodes are considered relays. It is assumed that both types of nodes are capable of performing this network role while stationary simple nodes are mere receivers (destinations) of multicast communication. Mobile relays bridge the gaps between disconnected network components. Stationary relays help to spread the messages to more mobile relay nodes that otherwise would not get in contact. They also extend the network reach to other nodes in their range. Multicast graphs are modeled with no, local and global network knowledge. The two latter knowledge modes use location beacons transmitted by each node.

Edge structures are aimed at reaching defined multicast destination nodes, regions and classes. Time distance of an edge in space-time multicast graph indicates how long a message will have to be buffered (carried) by the relay before it will be forwarded to the next hop node (device).

When 5 different multicast sources are selected, it results, together with 9 destination type and network knowledge mode combinations, in the maximum of 45 separate multicast graphs constructed in each of 93 topologies in every of 4 areas of interest. Some of the graphs may not be constructed when no nodes meet the criteria. Other might consist only of the source when source node does not

Table 6.2 Numbers of graphs modeled in simulation study

category	type	number of graphs
space	connectivity	14400
	minimum spanning forest	14400
	maximum spanning forest	14400
space-time	connectivity	372
	first-contact	372
	multicast	16362
in total		60306

establish any connections, e.g., due to its insufficient radio coverage or route. When constructed, the metrics and visualization of each space-time multicast graph are produced and stored.

Simulation scope and parameters resulted in total in 60306 graphs being modeled, as listed in Table 6.2.

6.1.4 Simulation study metrics

Studied metrics are depicted in the next sections first in absolute values, then some are presented as ratios (percentages) related to the reference ones. The metrics are listed in this section in relation to the first type of modeled structures they are discussed for. Other types may use the same or related metrics. The parameters are further divided into those that pertain nodes (devices) and into the ones that relate to edges (connections) of the graphs (networks). All used metrics are non-negative.

1. Space connectivity:
   (a) nodes:
      i. stationary destination nodes — the number of stationary destinations;
      ii. stationary relay nodes — the number of stationary relays;
      iii. mobile relay nodes — the number of mobile relays;
      iv. mobile relay nodes to all nodes ratio — the percentage of mobile relay nodes as compared to the number of all nodes;
      v. all nodes — total number of nodes;
      vi. connected components — the number of sets of nodes that are connected with each other by direct or indirect paths;
      vii. nodes per component — average number of nodes in a component;
   (b) edges:
      i. average node degree — average number of edges adjacent to a node;
      ii. edges — total number of edges;
      iii. cost — the sum of space distances of all edges;
2. Space-time connectivity:
   (a) nodes
      i. instances per node — the number of time nodes (instances) per unique device (space node)

   ii. instances per mobile relay node — the number of time nodes (instances) per unique mobile relay device (space node)

3. Space-time multicast

   (a) nodes:

      i. potential destinations — the nodes that could be multicast destinations in space-time connectivity graph of interest;
      ii. reached to potential destinations ratio — the percentage of multicast destination nodes that got reached;
      iii. nodes ratio to no knowledge — the percentage of number of nodes in graphs constructed with local and global network knowledge, as compared to the number of nodes when no network knowledge was used;
      iv. mobile relay nodes ratio to no knowledge — the percentage of number of mobile relays in graphs constructed with local and global network knowledge, as compared to the number of mobile relays when no network knowledge was used;
      v. core nodes ratio — the percentage of core multicast nodes as compared to the number of all nodes;

   (b) edges:

      i. space cost — the sum of space distances of all edges. It can be interpreted as an indicator of total power required to transmit messages;
      ii. time cost — the sum of time distances of all edges. It can be used as an indicator of total computing resources needed to propagate the message to the leaves of the tree;
      iii. time span — the number of time intervals (slots) covered by multicast graph. It is the maximum time reach of the message originated by source node. It can be interpreted as maximum delivery delay of given message;
      iv. time span cost — the sum of all maximum neighbor time distances for each space node (device). It indicates the total time the buffers of the relays need to be occupied by the message to reach all desired neighbors of each relay node.

6.2 Space connectivity analysis

Following the objectives and parameters defined in Section 6.1, Algorithm 1, *network devices data to slots of space nodes* (NDD-SSN) was used to select and group physical devices (space nodes) data into time slots. Then Algorithm 2, *slots of space nodes to space connectivity list* (SSN-SCL), was applied to construct *space connectivity lists* (SCL) related to each city and modeled radio ranges. With 6 second slots and 2 hours period of interest, each SCL consists of 1200 subsequent *space connectivity graphs* (SCG). Four cities and three radio ranges require twelve SCLs. This results in 14400 space connectivity graphs with 2.3 million nodes and 1.6 million edges in total. Those space connectivity lists are both the object of the analysis in this section, as well as, the key starting elements to construct space-time connectivity graphs modeled in the next sections.

6.2.1 Space connectivity nodes

Each city network is characterized by a constant number of stationary destination nodes—Warsaw: 160, Poznań: 116, Gdańsk: 92, Wrocław: 70—as compared in Figure 6.11. The numbers of stationary relay nodes presented in Figure 6.12 are also constant but the decreasing order is quite different—Gdańsk: 20, Poznań: 17, Warsaw: 4, Wrocław: 2.

Mobile relay numbers are the ones that vary significantly across cities which can be examined in Figure 6.13. Those numbers also change over time in the areas of interest. Although, they are

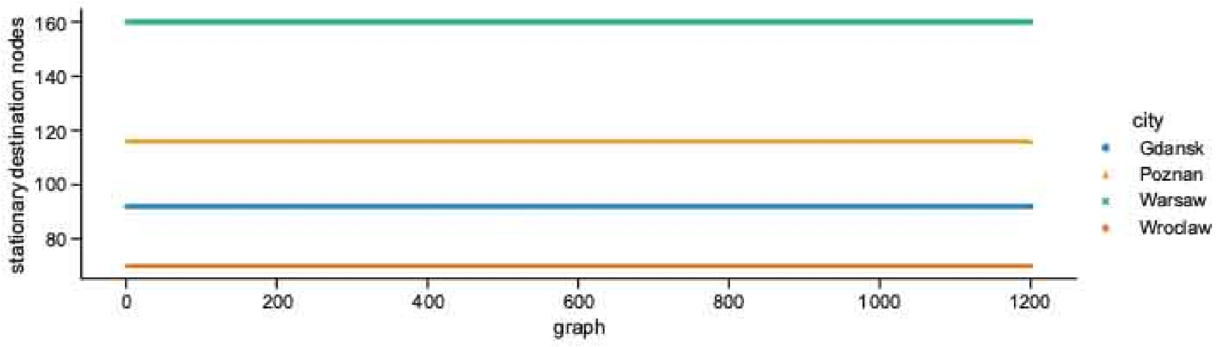


Figure 6.11 Stationary destination nodes in space connectivity graphs

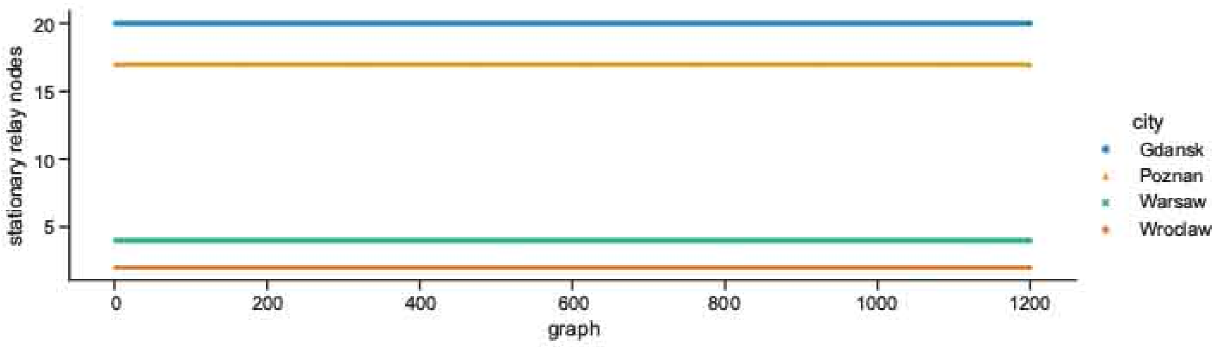


Figure 6.12 Stationary relay nodes in space connectivity graphs

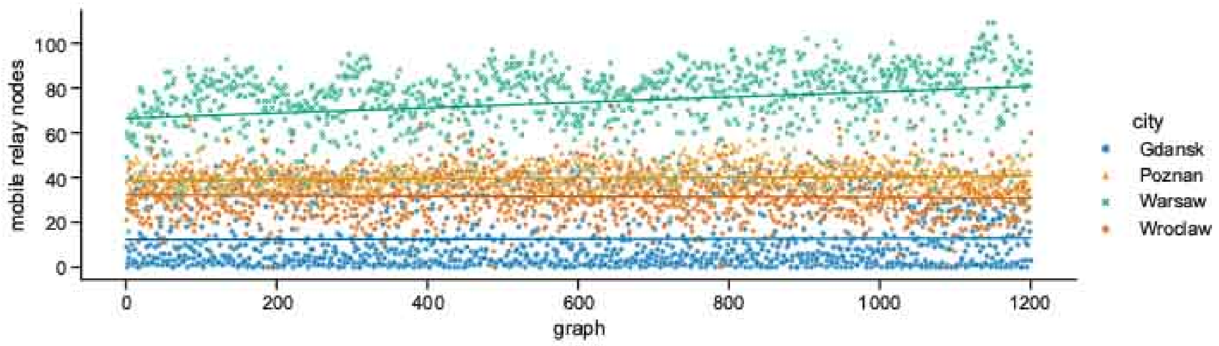


Figure 6.13 Mobile relay nodes in space connectivity graphs

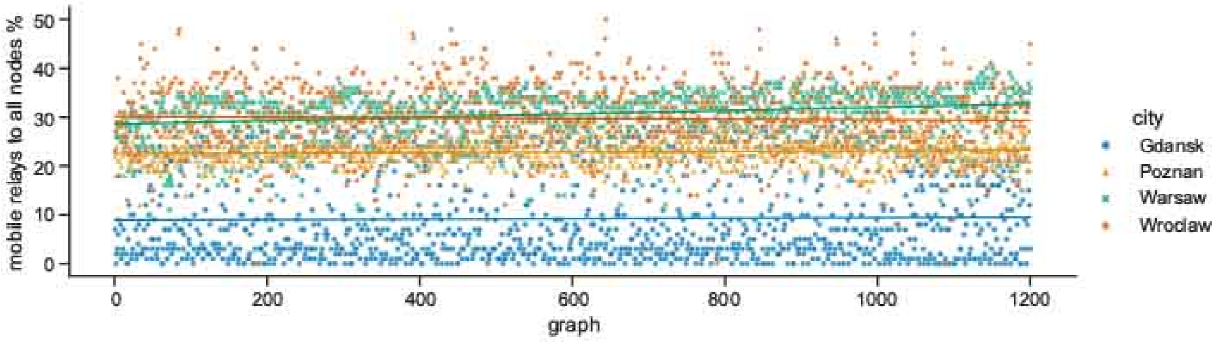


Figure 6.14 Mobile relay nodes to all nodes ratio in space connectivity graphs

oscillating around characteristic values—Warsaw: 74, Poznań: 39, Wrocław: 32, Gdańsk: 13. Only in case of Warsaw distinctive slope increase of the regression line can be observed. This shows that the number of mobile relays (vehicles) increases on average, which may be caused both by the need to address the increasing numbers of rush hours commuters, and by the traffic jams that may slow down the nodes. Warsaw also expresses the largest deviations from the regression line, especially for lower values. Conversely, in Gdańsk the numbers increase more dynamically but the maximum values, i.e., the numbers of public transport vehicles, drop by around 10 in about last 150 graphs (15 minutes). Mobile relay nodes number in Wrocław tends to be more evenly distributed while in Poznań the numbers are more tightly grouped.

The numbers of mobile nodes represent a different percentage of all nodes in given city. What is visible in Figure 6.14, the relative numbers are substantially lower in Gdańsk, while Warsaw and Wrocław tend to be close and characterized by the highest ratios, with Poznań falling slightly behind—Warsaw: 30, Wrocław: 30, Poznań: 23, Gdańsk: 9. The sums of respective stationary nodes numbers are the lower bounds of the total numbers of nodes in SCL compared in Figure 6.15. The deviations, and hence the maximum values, are related to the number of mobile relay nodes changing over time—Warsaw: 237, Poznań: 172, Gdańsk: 125, Wrocław: 104.

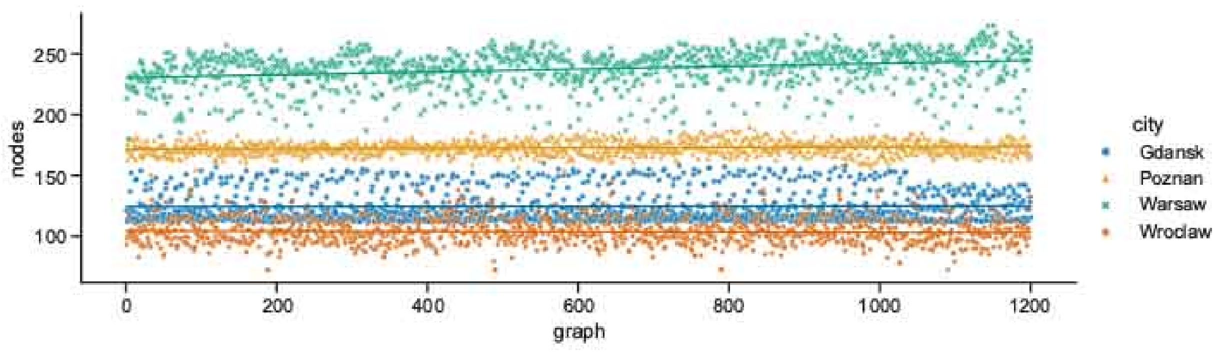


Figure 6.15 All nodes in space connectivity graphs

6.2.2 Space connectivity edges

While the numbers of nodes are shared by the networks in given areas of all investigated radio ranges, other graph metrics vary. It is clearly visible in the graphs of Figure 6.16 that as the radio range increases, the number of connected components decreases. The topologies in Wrocław are the most sparse, while in other cities average numbers of nodes per connected component depend more on the radio range, as compared in Figure 6.17. The numbers of average node degrees in Figure 6.18 and numbers of edges in Figure 6.19 increase with radio range. This follows the intuitive understanding that with the increasing radio range, the resulting network will be less fragmented. Moreover, in different cities the changes of node radio range influence the structure of the network to different extent. This is related to unique infrastructure topology and mobility patterns of each area. It is visible in particular when the changes in Warsaw and Wrocław get compared. Also, the average node degrees and edge numbers suggest that nodes in Poznań and Warsaw are denser and evenly distributed.

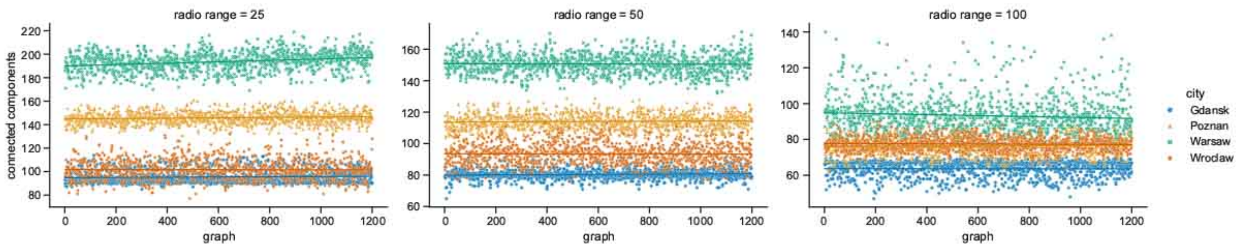


Figure 6.16 Connected components in space connectivity graphs

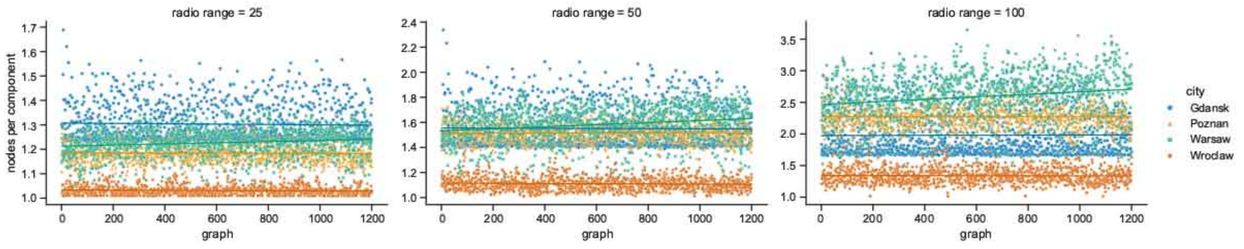


Figure 6.17 Average number of nodes per connected component in space connectivity graphs

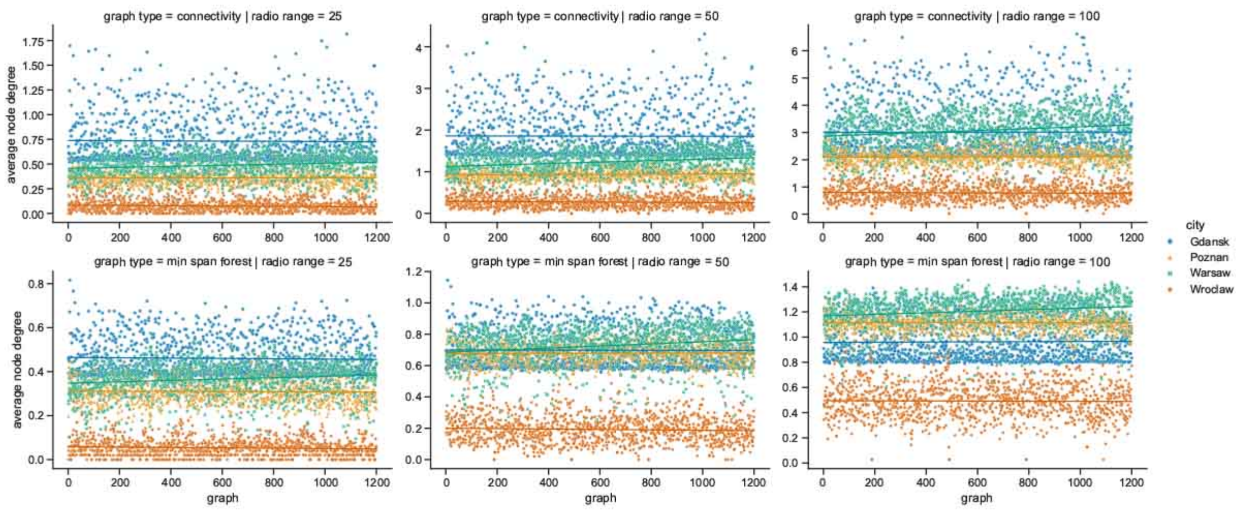


Figure 6.18 Average node degree of space connectivity graphs and minimum spanning forests

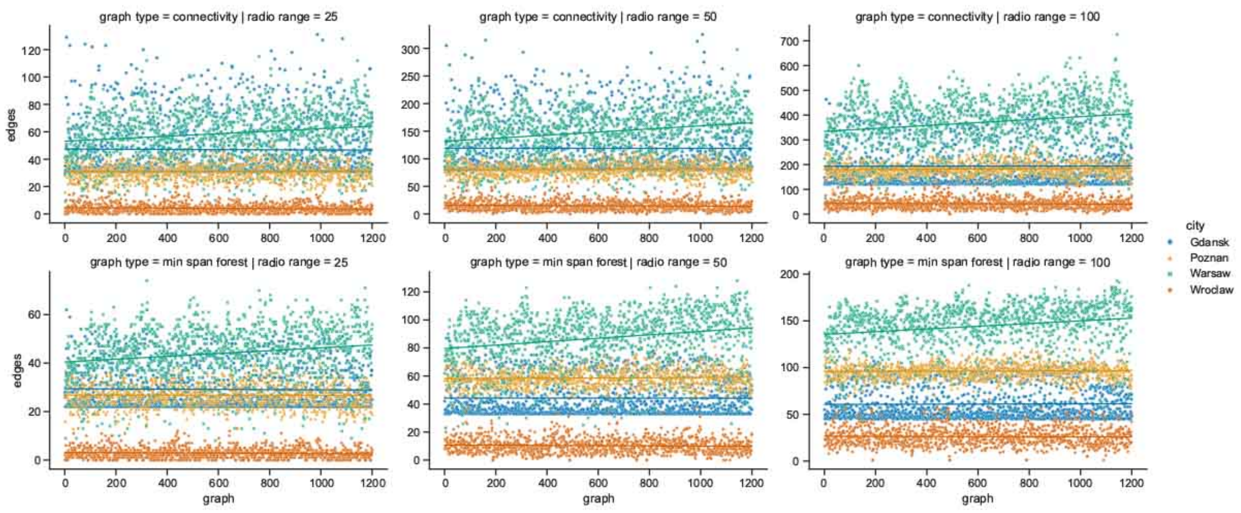


Figure 6.19 Edges of space connectivity graphs and minimum spanning forests

Moreover, the parameters of minimum spanning forests constructed in space connectivity graphs are influenced by increasing radio range in different way in different cities. When the networks are less fragmented, the number of edges in such trees, as well as, average node degrees increase. Albeit, they do not influence the distributions in the same way, as visible in the bottom row graphs of Figure 6.18 and Figure 6.19. Those figures do not present the parameters of maximum spanning forests since the numbers of edges and average node numbers were the same as for minimum spanning forests. The cost of the structures is different, though, as presented side-by-side in Figure 6.20. There, the larger the radio range, the bigger the cost and the difference between the costs of minimum and maximum spanning forests.

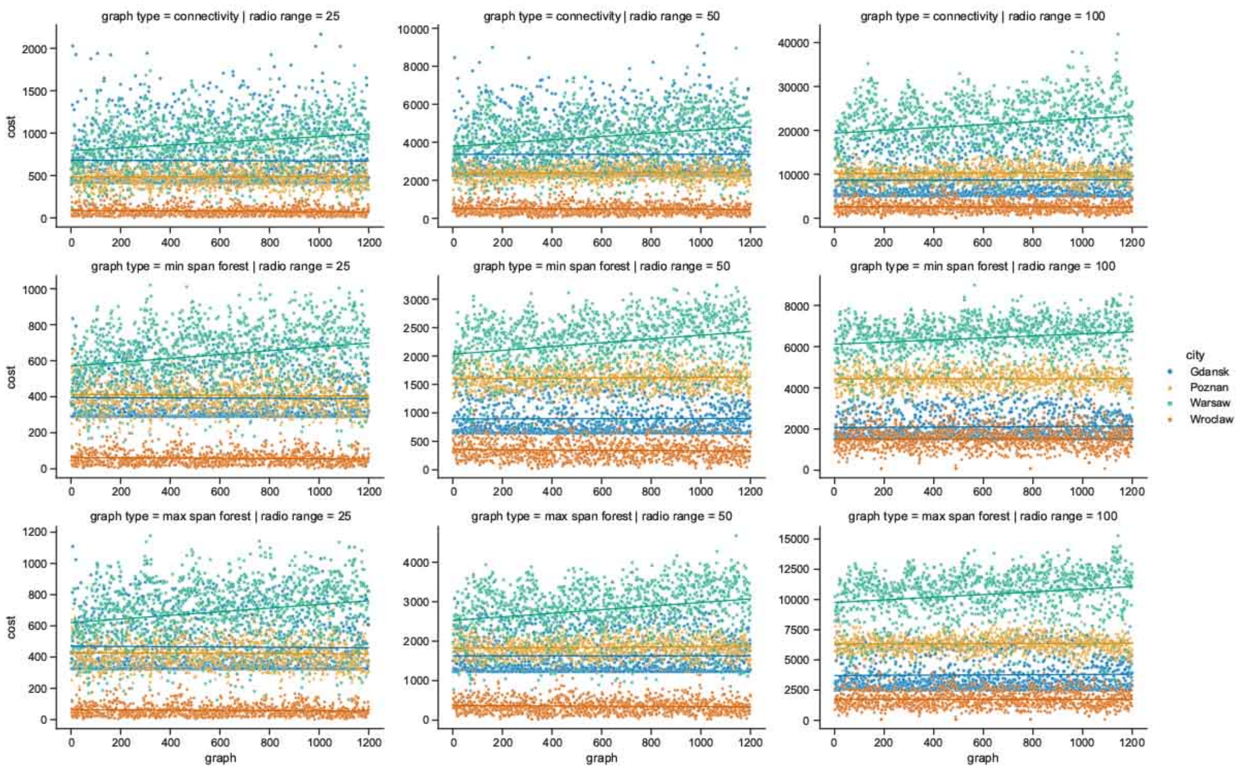


Figure 6.20 Cost of space connectivity graphs and spanning forests

6.2.3 Space connectivity relationships

Figure 6.21 presents the relationships between multiple variables of space connectivity graphs when radio range was set to 50 m. It has been already shown that each range results in unique topological features of the networks. Although, this particular range, being the middle one of the simulated values, was selected to give an overview of the general differences of characteristics of the urban areas (cities) under scrutiny. One set of parameters, i.e., the columns, changing along horizontal axis of the figure, covers slot (graph) number, the number of mobile relay nodes, as well as, the number of edges. The other set, the rows, changing along vertical axis fo the figure, covers the numbers of mobile relay nodes, the numbers of edges, graph costs, average node degrees, and the numbers of connected components.

Both presented histograms, i.e., the number of mobile relay nodes and the number of edges, suggest that each urban area and infrastructure has indeed its own features and parameters. Almost every other related graph proves it even stronger by presenting mostly disjoint groups of measurements for each city. They display, though, a degree of correlation in terms of the increasing trends they follow. Interestingly, only in Warsaw the number of connected components clearly decreases with the increase of the number of mobile relay nodes and edges. In Wrocław, the increase in the number of mobile relay nodes and in the number of edges are followed by the increase of connected components. At the first glance, one could think that it means, that despite getting more populated with mobile relays, the structure becomes more disconnected. Here the situation is quite different and when more

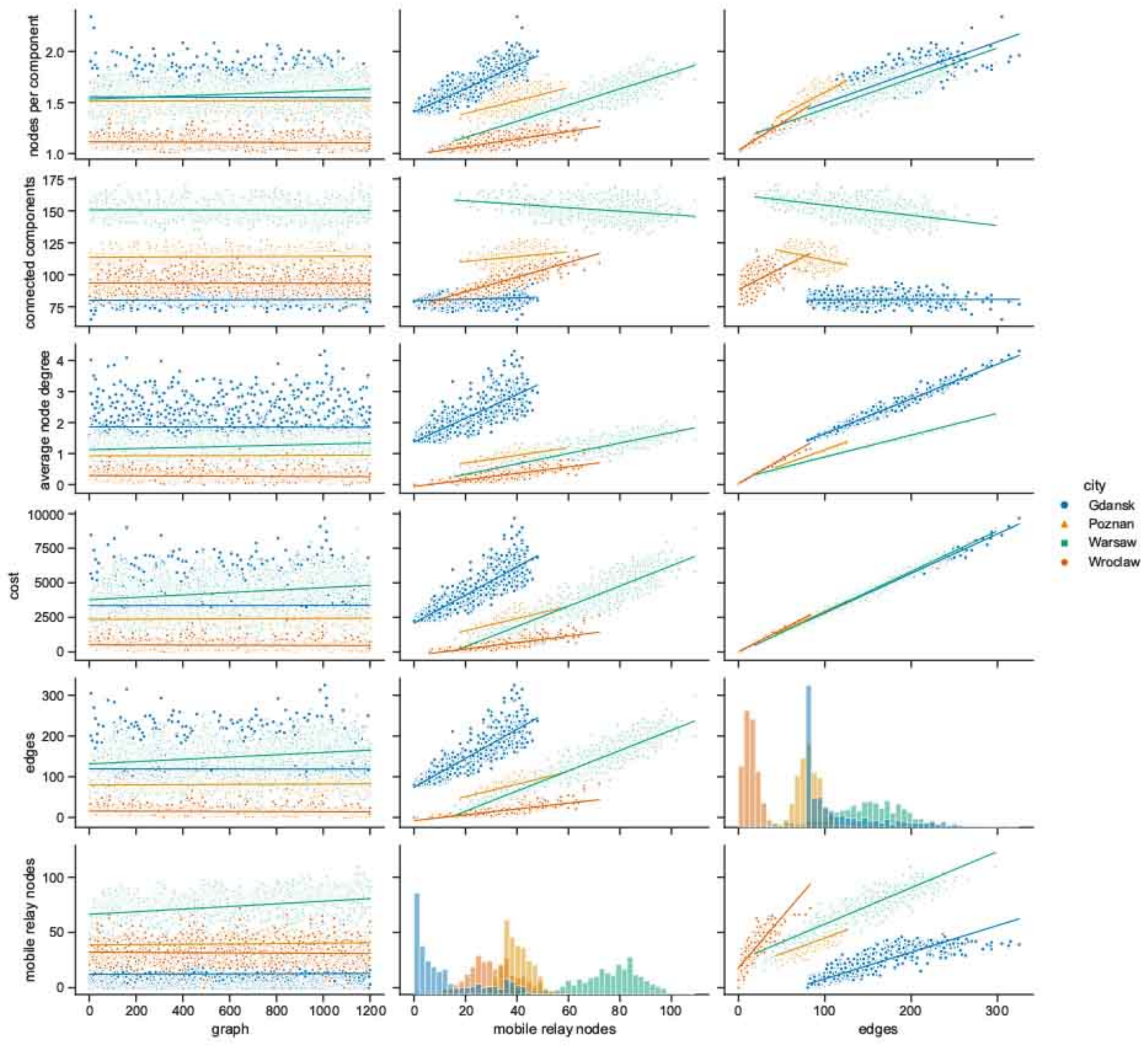


Figure 6.21 Space connectivity parameters in Polish cities at 50 m radio range

mobile relay nodes are present, more nodes can be connected, and yet, forming new disconnected components, rather than getting connected to the larger ones.

The conditions in Poznań may seem even less intuitive when the number of connected components increases with the number of mobile relays and decreases with the increase of the number of edges. It means that with more graph edges, the nodes are, on average, connected in larger groups (components). In this urban area, more mobile relays, like in Wrocław, cause more nodes to be connected and constitute more smaller groups. In Gdańsk, which is the city with the lowest numbers of mobile relay nodes, those numbers and slope features are influenced only to little extent but with noticeable deviations from the trend. The highest direct correlation and overlapping is present between the numbers of edges and the costs of the trees in all of the cities. The relationship is almost linear, with little deviations, which suggests that the average edge costs are similar. Average node degrees differ more, and yet, depend on the number of mobile relays in a way resembling the

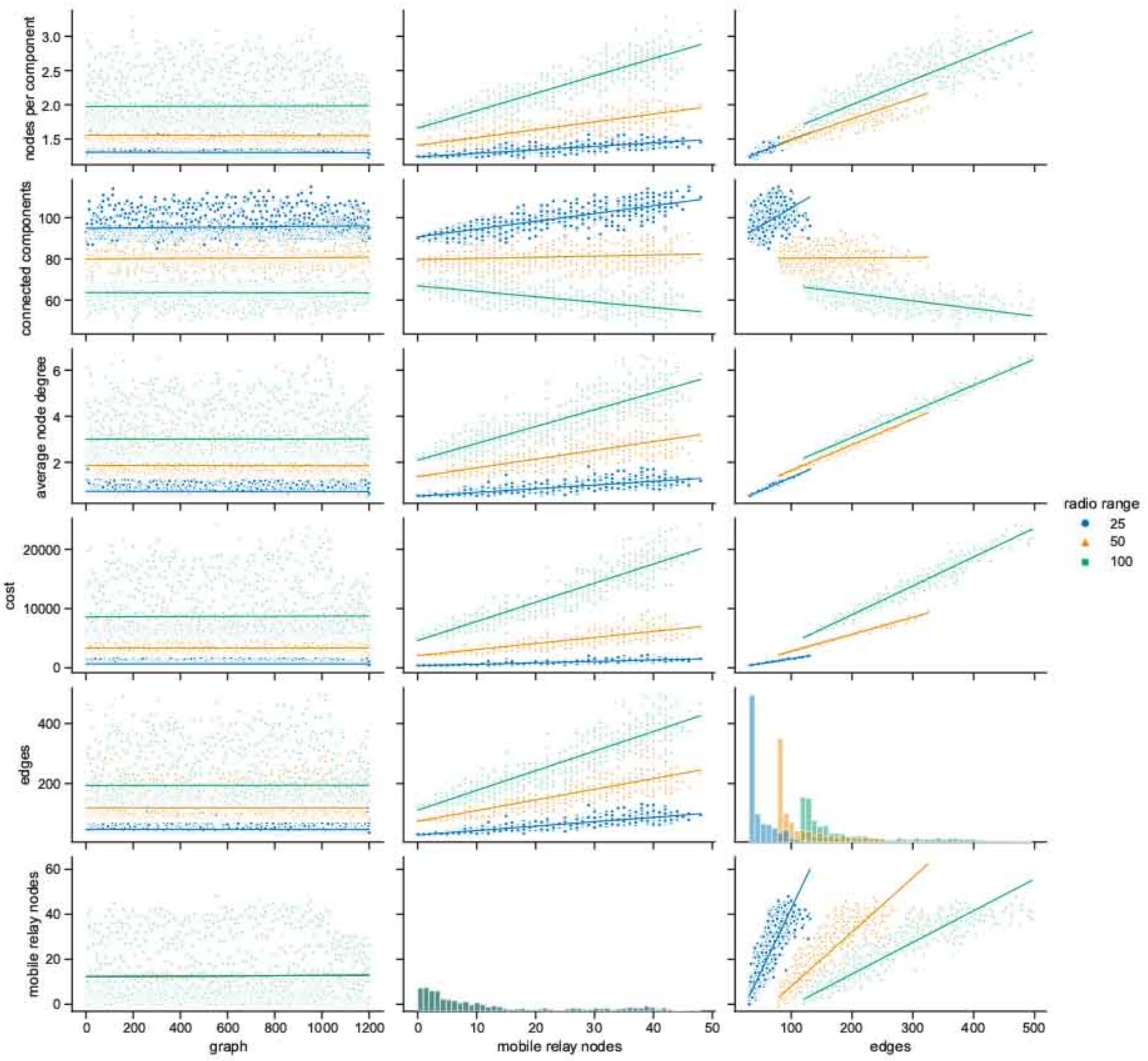


Figure 6.22 Space connectivity parameters in Gdańsk

relationship between the number of mobile relay nodes and edges. In terms of edges to average node degree relationship the city-related distributions appear as distinctive linearly condensed groups.

To investigate the features in more detail, space connectivity parameters of each city were presented and analyzed for all radio ranges separately in the next subsections.

Space connectivity in Gdańsk

Taking a closer at the parameters of space connectivity graphs in Gdańsk in Figure 6.22 reveals that the trends related to connected components are linked not only with the number of mobile relay nodes and edges but also with radio range. When the range increases from 25 to 50 meters the slope of regression line changes from increasing to almost horizontal. When radio range extends as far as 100 meters it becomes decreasing. This tendency is also influenced by the largest number of

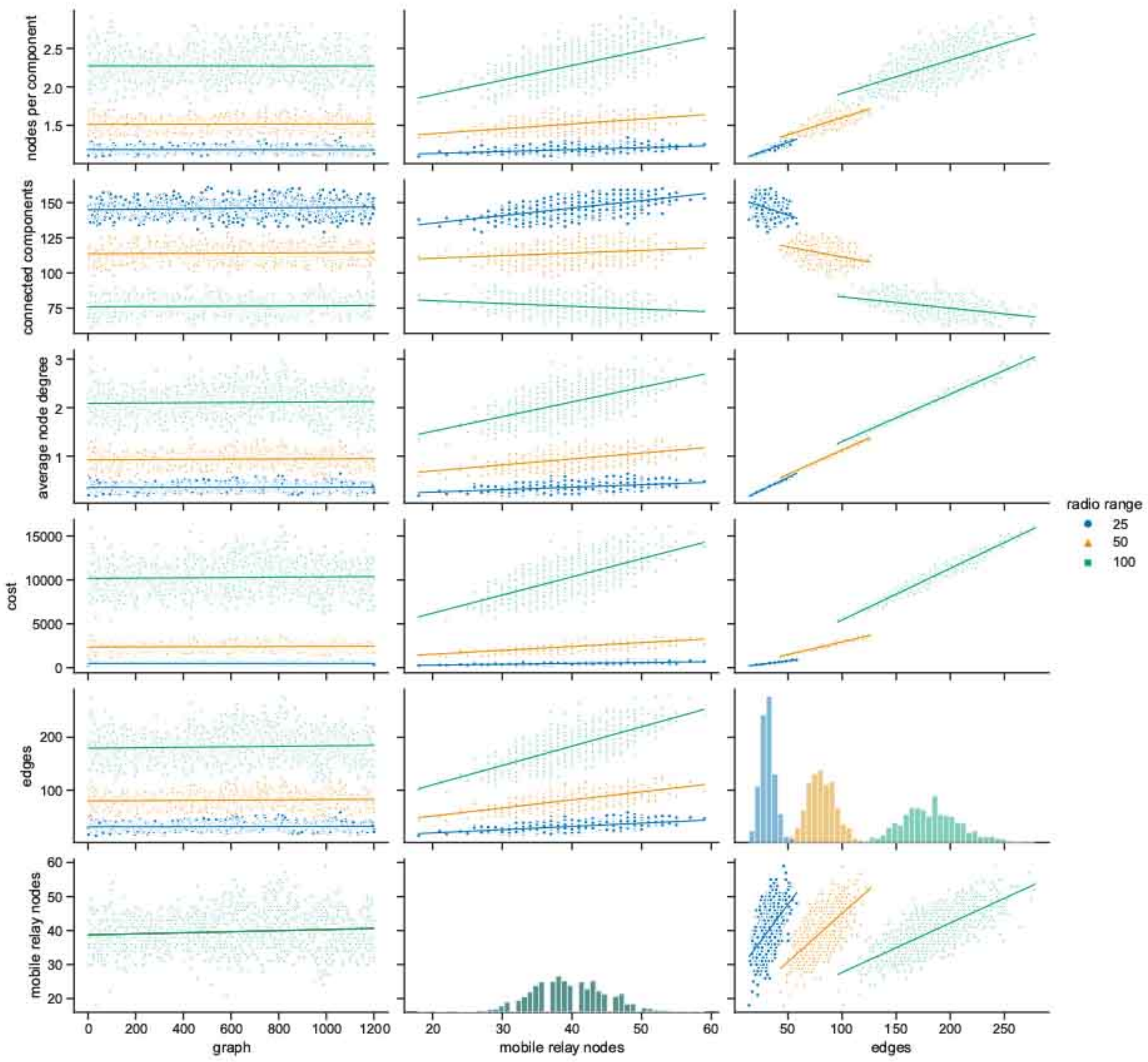


Figure 6.23 Space connectivity parameters in Poznań

stationary relays of all studied cities, as visible in Figure 6.12. This means that with more radio range more nodes could be reached and connected to make larger components of the graph. The average node degrees, and hence, complexity and costs of such structures are significantly higher. It cannot be overlooked that in Gdańsk usually only a few mobile relays were present, which is about one order of magnitude less than in other cities. The distribution of the numbers of edges is correlated with the distribution of mobile relay nodes, getting more flattened and shifted in the direction of higher values as the radio range increases. It should be noted that similar situation occurs in the relationship between the number of edges and average node degree.

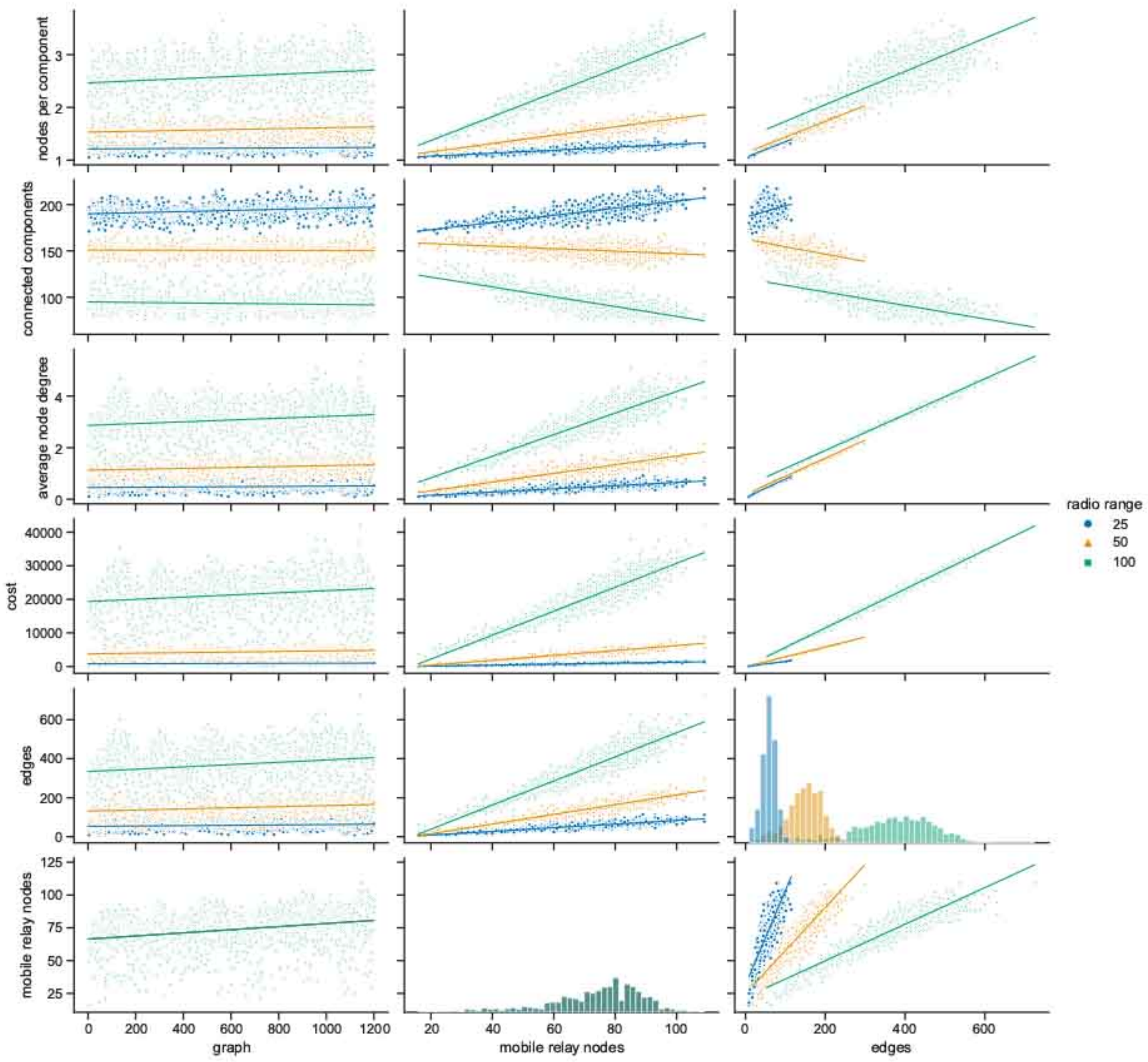


Figure 6.24 Space connectivity parameters in Warsaw

Space connectivity in Poznań

Space connectivity parameters in networks modeled in Poznań and compared in Figure 6.23 at first seem similar to the ones of Gdańsk. They share common features but a number of characteristics are quite distinctive. The first difference is the much larger number of mobile relays in Poznań. The distributions of edge numbers are more Gaussian and shifted as the radio range increases. Average node degrees and graph costs are of nature comparable to Gdańsk. The slope of the trend of connected components in relation to edges is here decreasing in all range cases. This informs that, even at the lowest radio range the numbers of mobile relays are high enough and their routes coincide with other nodes at a level which makes the network more connected and dense. The situation is also influenced by seventeen stationary relays, as introduced in Figure 6.12. It can be said, that

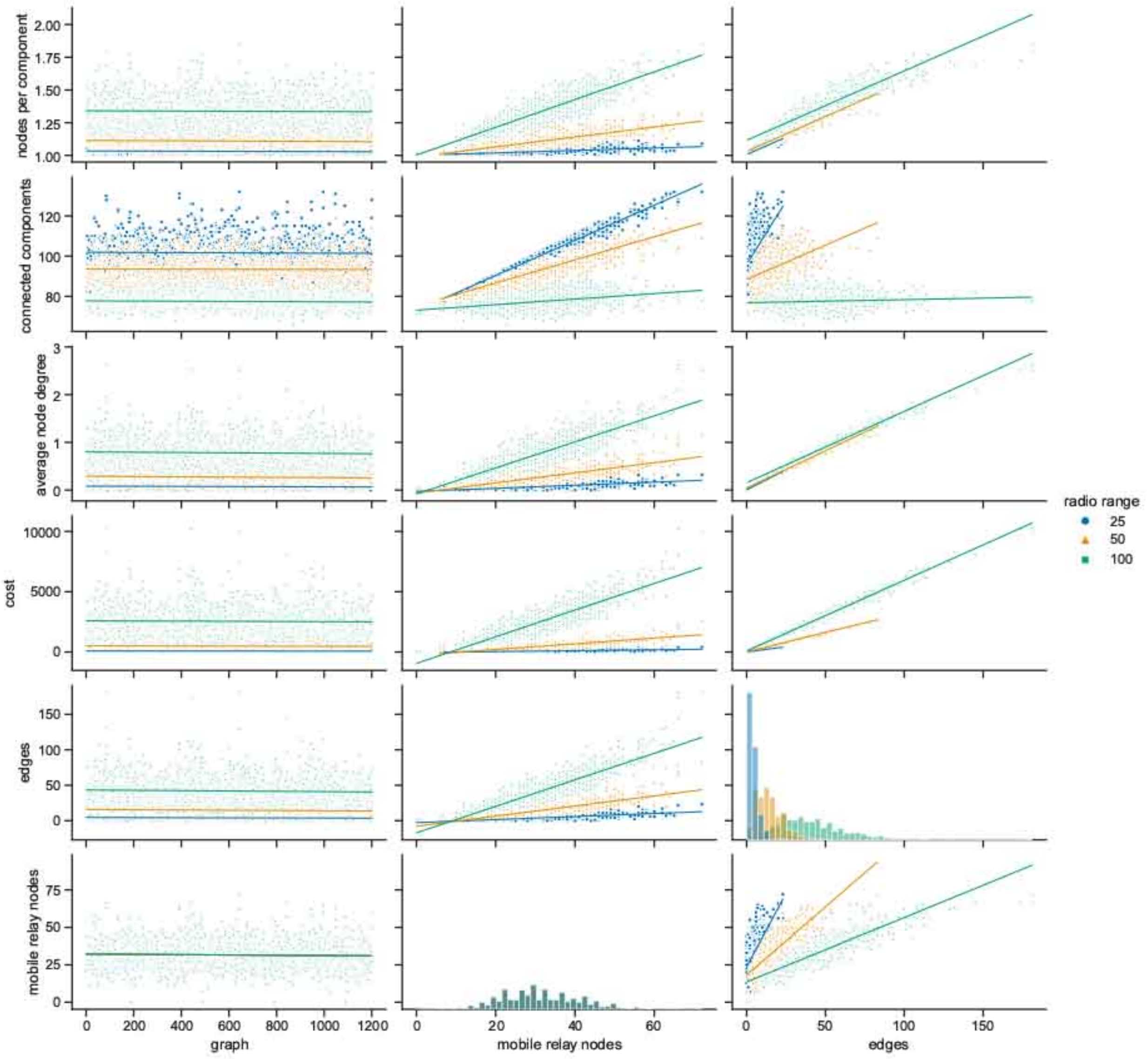


Figure 6.25 Space connectivity parameters in Wrocław

Poznań is the city of a topology which is the easiest one to achieve high level of connectivity at the lowest cost.

Space connectivity in Warsaw

The average numbers of mobile relay nodes in Warsaw are almost 6 times higher than in Gdańsk. Although, the distributions of metrics of Warsaw space connectivity graphs in Figure 6.24 share more similarities with Gdańsk than with Poznań. In Warsaw, the numbers of connected components, edges, and graph costs are twice as high as the ones in Gdańsk. Moreover, most of the metrics increase over time. Average node degree is the one that stays on similar level in both cities, being more compressed in Warsaw. Importantly, despite having the largest number of mobile relays, for each radio range, there is a number of graphs of only a few edges. Hence, stationary network nodes are more distributed and disconnected on their own than in Gdańsk and Poznań. Only four stationary relay nodes, please see Figure 6.12, do not improve the connectivity enough either. Although, the isolated stationary nodes get connected owing to the largest and increasing over time numbers of mobile relay nodes. It results in the highest numbers of edges of all the cities, as well as, the larges graph costs.

Space connectivity in Wrocław

In Figure 6.25 it is striking that Wrocław is the only city in which, in spite of tens of mobile relays, there are numerous graphs with no edges. This means that stationary nodes are heavily disconnected, especially when radio range is the lowest. Moreover, many of them are located beyond the routes and range of mobile relays. Wrocław is also the city of the lowest number of stationary relays with only two nodes of this kind, please see Figure 6.12. On the one hand, despite Wrocław being the city with the lowest overall numbers of nodes, it is the city with around 30% of mobile relays, on a par with Warsaw. This is the largest percentage among the cities under scrutiny, as presented in Figure 6.14. On the other hand, the exact numbers of mobile relay nodes are comparable to Poznań which has quite different distributions related to connected components. In the edges to connected components relation the trend is highly increasing for lower radio ranges, becoming almost vertical for the largest one. This stresses out the uniqueness of both the topology of the area and networked infrastructure deployment.

6.3 Space-time connectivity analysis

Based on space connectivity graphs analyzed in Section 6.2, *space-time connectivity graphs* (STCG) were constructed using Algorithm 3, *space connectivity list to space-time connectivity graph* (SCL-STCG). Then, each STCG was used to build related *first-contact graph* (FCG) with Algorithm 4, *space-time connectivity graph to first-contact graph* (STCG-FCG). While STCG, which is a *time-expanded graph*, is mostly an intermediate network modeling structure, FCG is a *time-aggregated graph* of more practical meaning. The analysis of FCG provides a general and easier to comprehend impression of how the space-time network changes in real environment and how the adjacencies occur for the first time. Therefore, the results for 372 graphs of both space-time types are presented and discussed side by side.

The graphs were constructed in four cities, at three radio ranges and for five durations that divide the period of interest into respective networks. Each consecutive duration is the double of the preceding one. Therefore, due to this geometric growth nature, the increasing trend visible in Figure 6.26 for connectivity graph which seems exponential is in fact of rather linear nature. Conversely, it looks linear in case of first-contact graph, and hence, it is of more logarithmic type.

Since the numbers of stationary nodes were constant in space connectivity graphs, also in the respective space-time connectivity graphs they remain at the same level in each area and duration. The numbers of mobile relay nodes presented in Figure 6.27 are varying resulting in correlated changes in the overall number of nodes in the graphs. It is caused by the fact that not only the number of mobile relays changes over time but also the mobile nodes (the devices) present in the

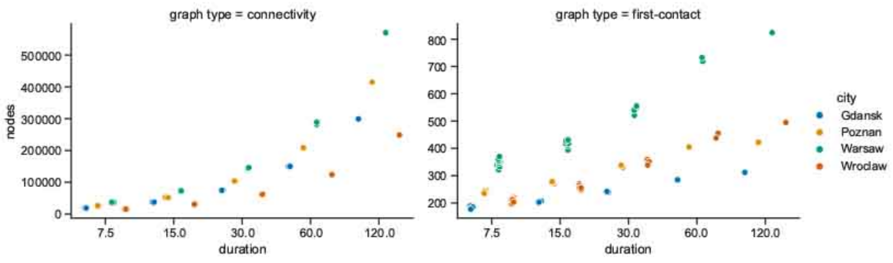


Figure 6.26 All nodes in space-time connectivity and first-contact graphs

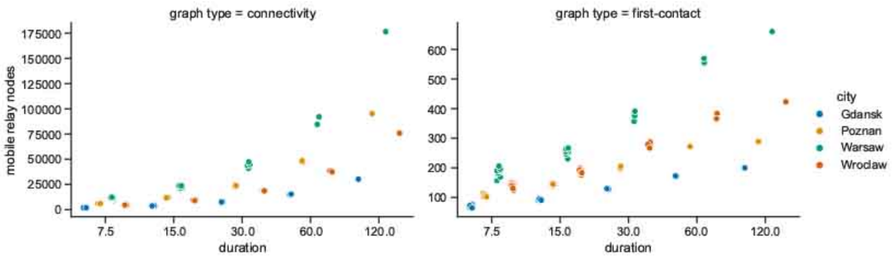


Figure 6.27 Mobile relay nodes in space-time connectivity and first-contact graphs

area change. Moreover, some leave the area while other enter at different points in time. In the spatiotemporal network there are more unique nodes (devices) than in each single space connectivity graph. This is the reason why the numbers of mobile relays in first-contact graphs are at lease a few times higher than in space connectivity graphs in Figure 6.13. They hardly exceed 200 mobile relay nodes in Warsaw for the shortest duration. For the longest one, the number reaches 660 mobile relays. In other areas the numbers are lower. Interestingly, Poznań-related STCGs consist of more mobile nodes than the ones related to Wrocław. In FCG, due to the structure of the connections, the situation is the opposite. Gdańsk remains the area with the lowest numbers of mobile relay nodes at all durations.

Importantly, the STCG construction algorithm is the reason of large numbers of nodes present in space-time connectivity graphs since each node of each SCG is represented by two instances in STCG. It means, that for each unique device (space node) in the observation area and duration, multiple instances (time nodes) are present in space-time connectivity graph, as depicted in Figure 6.28. For

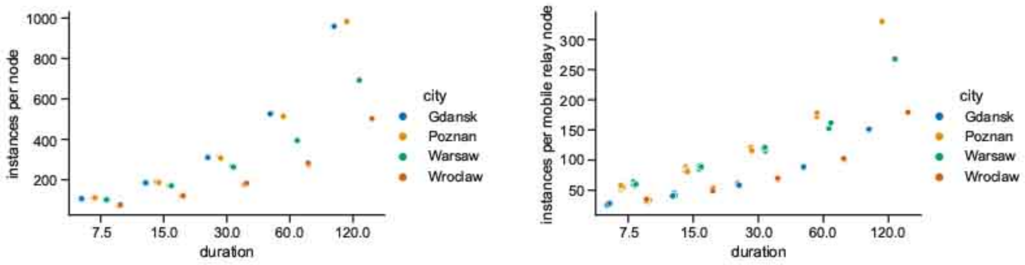


Figure 6.28 Instances per nodes in space-time connectivity graphs

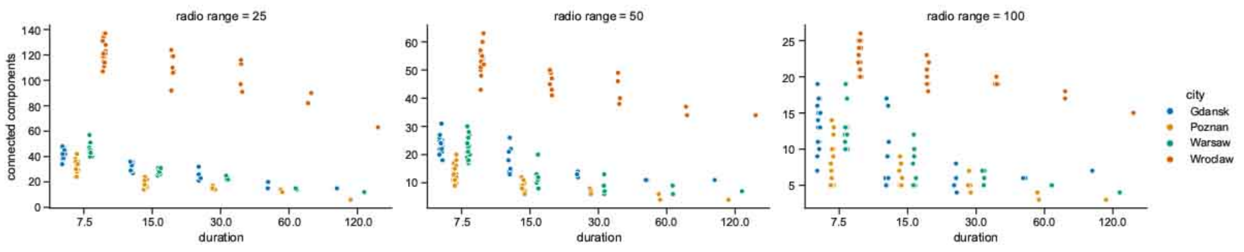


Figure 6.29 Connected components in first-contact graphs

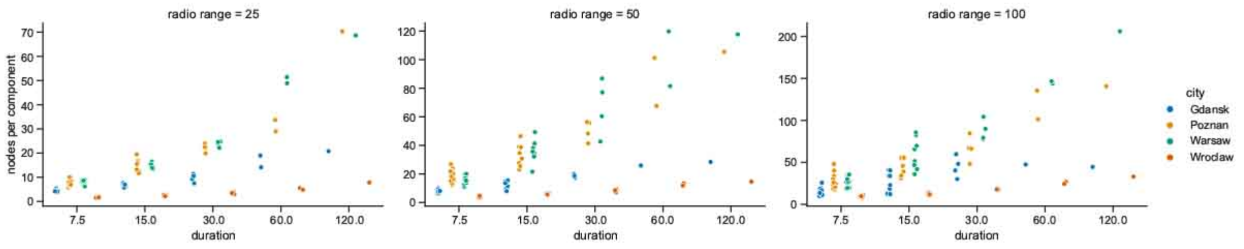


Figure 6.30 Nodes per connected component in first-contact graphs

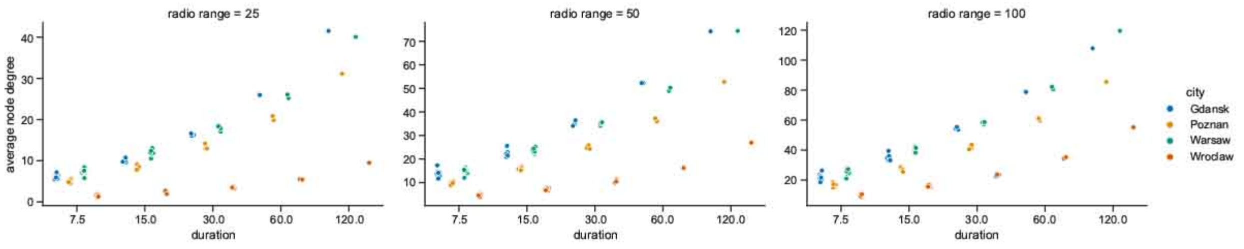


Figure 6.31 Average node degree in space-time first-contact graphs

example, the numbers of nodes for 30 minutes duration in STCG range from around 75 thousand for Gdańsk to more than 140 thousand for Warsaw. They correspond to around 240 nodes for Gdańsk and 540 nodes for Warsaw in related first-contact graphs. The numbers of instances of mobile relays are the highest in Poznań and Wrocław. It means that in those areas of interest particular mobile relays are on average present for the longest total time. Figure 6.28 shows the numbers only for space-time connectivity graphs because in first-connectivity graphs each actual device is represented by a single node.

In terms of connected components in first-contact graphs, see Figure 6.29, their numbers for Gdańsk, Poznań and Warsaw are lower than they were on average in space connectivity graphs compared in Figure 6.16. The difference grows when network duration and radio ranges increase. The same trend can be observed in Wrocław but there is a difference. At shortest durations and the smallest radio range the numbers of connected components exceed the ones in SCGs. It shows that the infrastructure in Wrocław is more fragmented and more time is needed to make the network more connected.

The opposite trends can be observed in relation to the numbers of nodes per component in Figure 6.30 and average node degrees in Figure 6.31 when compared to Figure 6.17 and Figure 6.18, respectively, which are related to these parameters of space connectivity graphs. The much higher values of space-time metrics are of key significance, because they prove that when mobile relay nodes are used to build a space-time network, the momentarily (temporarily) disconnected components (parts of the network) get connected over time and, as a result, larger and more dense time-spanning networks are constructed. Wrocław falls behind considerably but it needs to be taken into account that the underlying space connectivity graphs were of the lowest average node degrees as well.

6.4 Space-time multicast analysis

Opportunistic multicast algorithms introduced in Section 5.2 were simulated in space-time connectivity graphs analyzed in Section 6.3. In total, 16362 *space-time multicast graphs* (STMG) in *time-aggregated* form were constructed with the use of no, local, and global network knowledge. Each destination type results in different numbers of nodes that will be considered multicast destinations. In the simulated scenarios, the numbers are the lowest when the destinations include individual nodes. They increase when the nodes in particular geographical regions are targeted. The numbers increase even more when defined classes of nodes are meant to be the receivers and there are no imposed geographical boundaries. Since those potential destinations are stationary, their numbers remain constant in the modeling periods of interest for given city area, destination type and duration, as presented in Figure 6.32. In destination types defined as classes and regions they are the highest in Warsaw, followed by Poznań and Gdańsk, with Wrocław closing with the lowest numbers. In routing directed at individual nodes the number is the same for all cities.

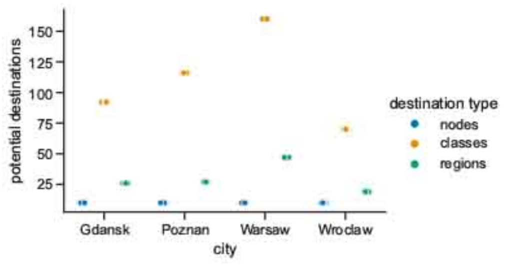


Figure 6.32 Potential multicast destinations

6.4.1 Multicast nodes

Due to highly multidimensional structure of the data related to modeled multicast graphs, different views on it have to be presented. First, the trends of numbers of nodes will be analyzed in relation to Figure 6.33. In this figure, the numbers are compared for graphs that are the multicast structures directed at reaching the nodes within given destination regions. This view is chosen because it presents the scenario which is between the one with the smallest numbers of destinations, i.e., individual nodes directed multicast, and the one with the largest numbers, i.e., classes directed multicast. Moreover, there exists a relationship between the number of destination nodes and the parameters of the resulting graphs. It will be further discussed and the reasoning will be proved valid.

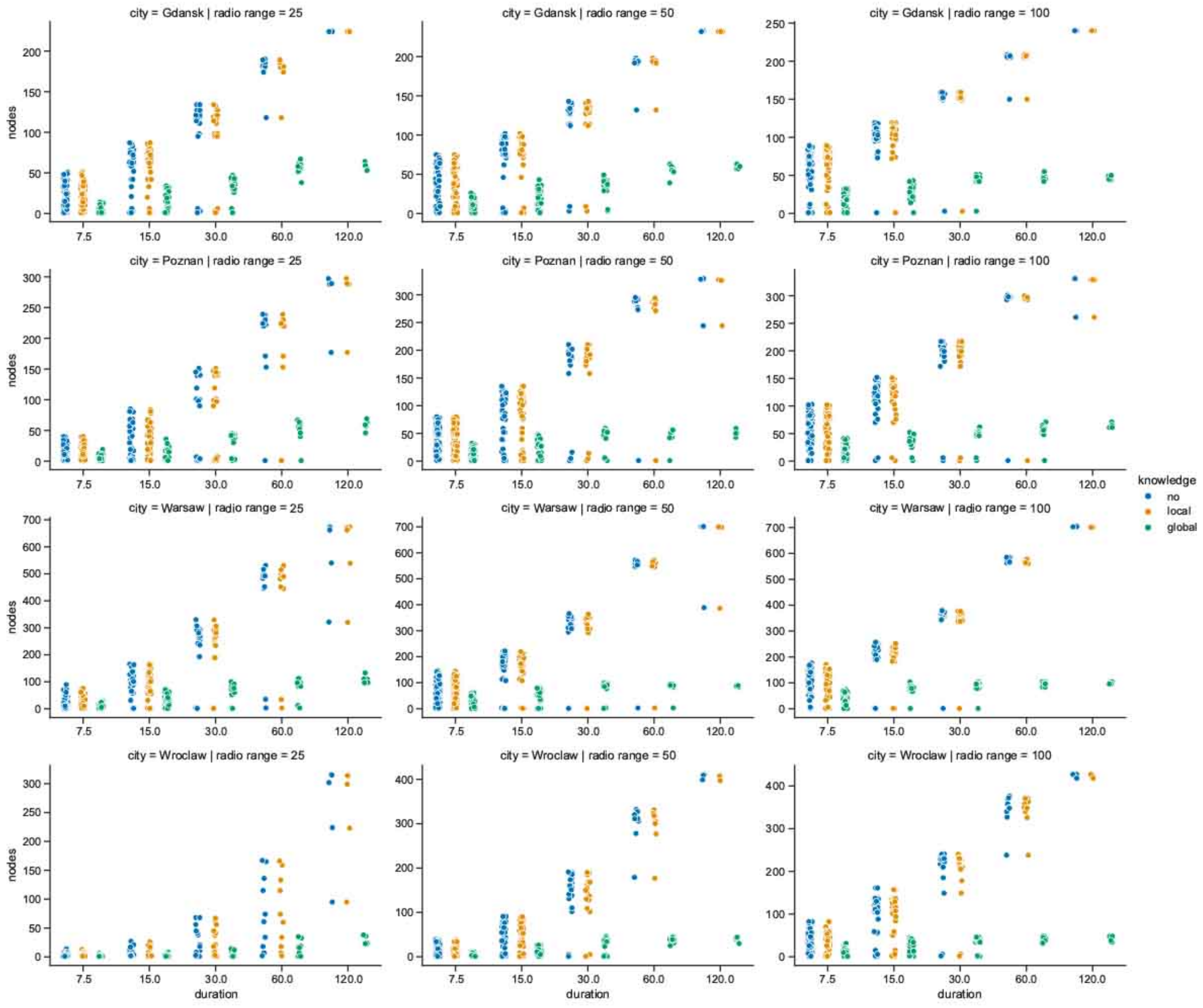


Figure 6.33 All nodes in multicast graphs with destination regions

It may seem that algorithms operating with no and local knowledge of the network result in the trees of exactly the same parameters. In most of the graphs it is true but in some differences can be noticed when closer examined. At this stage of the analysis both these network knowledge modes shall be considered as of the same qualities.

Figure 6.33 indicates that after the initial phase of increase of the numbers of nodes that follows the increase of duration and radio range, the numbers approach or reach the limits related both to the overall number of individual nodes in the area and to the number of destination nodes. The structures in each city have their own features, and at the same time, share many of the characteristics. To investigate them in more detail the networks of 30 minutes duration will be studied since this period is the middle one of the periods of interest and assures that drawn conclusions can be accordingly extended to both shorter and longer durations.

The overall number of nodes in multicast structures of 30 minutes shown in Figure 6.34 is a metric that spans three dimensions—radio range, destination type, and network knowledge. Moreover, the numbers are presented in groups related to each city. It is visible that the distributions of

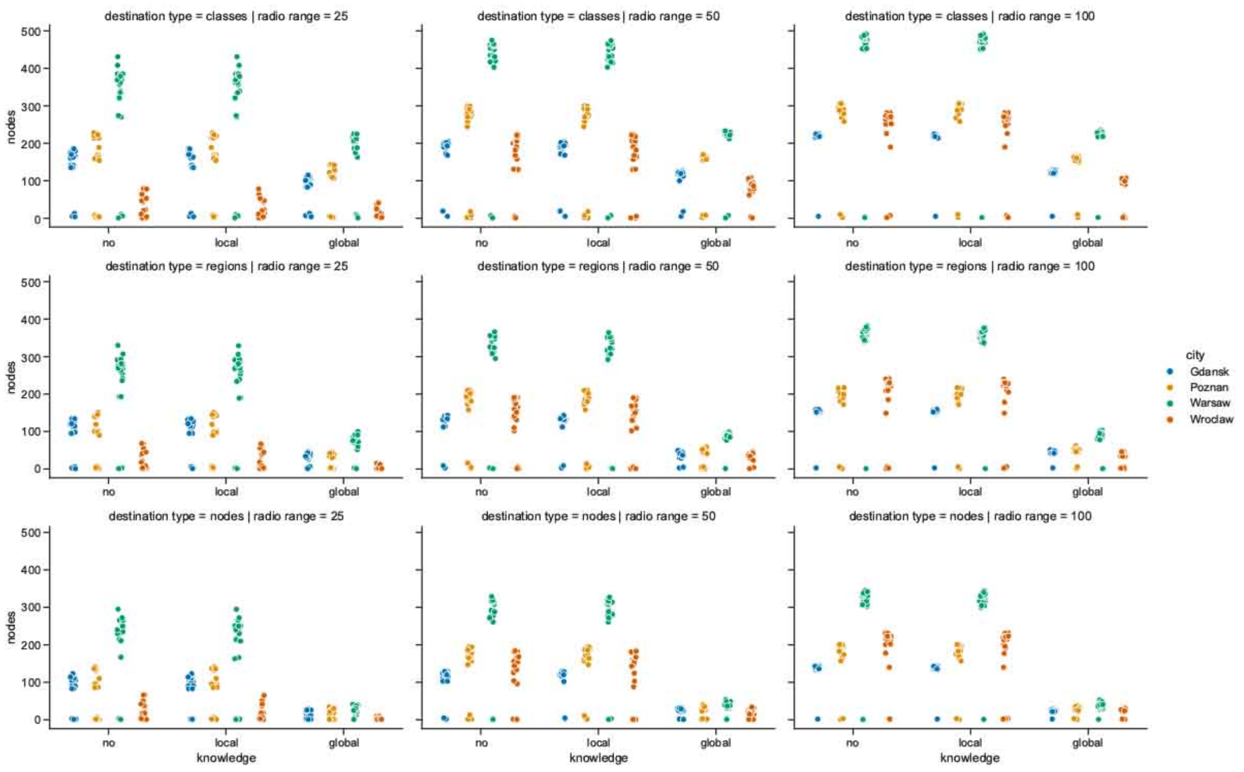


Figure 6.34 All nodes in multicast graphs of 30 minutes

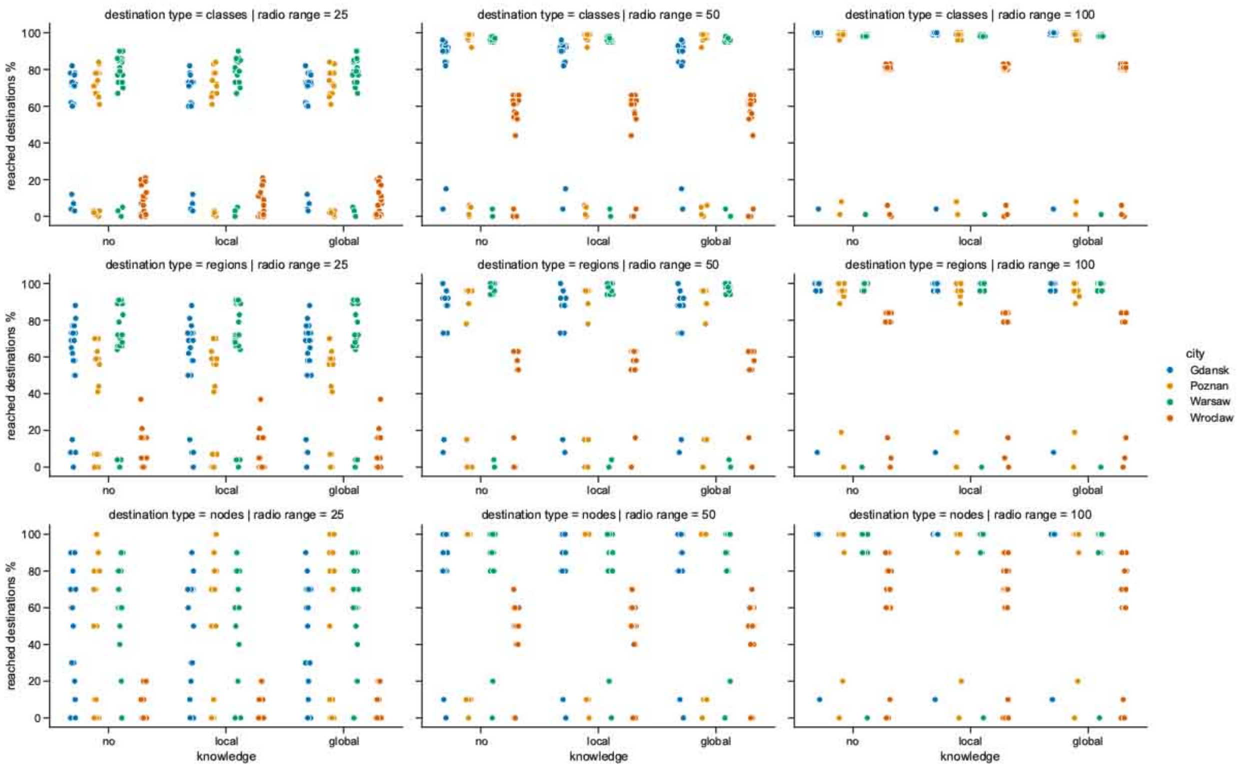


Figure 6.35 Reached to potential destinations ratio in multicast graphs of 30 minutes

city-related groups are similar to the ones already seen in relation to the space-time connectivity and first-contact graphs, for example, in Figure 6.26. It shall be noted that in each case there are a few trees composed of only single nodes. This means, in terms of multicast communication, that the source was unable to relay the message to any other node. Similarly, there are graphs that consist of only a few nodes which got connected to the tree. The most of the trees are composed of significantly more nodes. It is no surprise that when global knowledge of network topology is used, the numbers of the nodes are visibly lower since the trees are being constructed only of the nodes that lead to the destinations. The difference increases when the number of desired destination nodes decreases in comparison to the total number of nodes in the network.

The numbers of nodes in Figure 6.34 exhibit varying vertical spreads. For instance, in classes destination type and 25 meters of radio range, the metrics for Poznań are the most spread (distributed) while in Wrocław they are the most condensed. They grow and get more condensed with the increase of radio range. This observation follows the one presented so far, i.e., that network connectivity increases with increasing range, and hence, more nodes could be reached and become the members of the tree. The overall increase is, though, not as steep as in the numbers of nodes per connected component and average node degrees of space-time connectivity graphs in Figure 6.30 and 6.31. This proves that even at lower radio ranges, multicast trees of complex structure and space-time reach can get constructed. It is visible in Figure 6.35 which compares the percentages of potential multicast destinations that got connected to the trees. When destination type changes and fewer potential destinations are to be reached, the overall numbers of nodes in the trees decrease proportionally. Although, the efficacy of the tree construction, i.e., the numbers of reached destinations, remains high in each variant and city. With the lowest radio range it mostly stays between around 50 to 90 percent in Gdańsk, Poznań and Warsaw. It is visibly lower in Wrocław but when the range increases, it moves towards much higher values, still falling behind. In other cities the efficacy grows and the algorithms reach 100 percent when modeled space-time networks are interconnected well-enough.

As compared in Figure 6.36, with global knowledge the average number of nodes can range from 60 percent of the number of nodes in case of no knowledge, to as low as less than 20 percent, depending on destination type and radio range. It is related to the flooding-like nature of no and local knowledge variants which produce branches not leading to the destinations. It shall be pointed out, that local knowledge based routing tends to produce multicast structures composed of numbers of nodes similar to no knowledge operating condition. Although, there are graphs that are built with a few percent less nodes.

The distributions of the numbers of mobile relay nodes in Figure 6.37 are correlated with those of the overall numbers of nodes in Figure 6.34 and span from a few nodes to hundreds of relays. Mobile relay nodes ratios, when the baseline is the number of mobile relays in graphs constructed with no knowledge, depicted in Figure 6.38, resemble the distributions of all nodes presented in Figure 6.36. The percentages of mobile relays in the cities of interest seem to be decreasing, in most of configurations, when radio range increases—starting with Wrocław, then Warsaw, Poznań, and Gdańsk. Global knowledge based approach requires on average only around 10 to 60 percent of mobile relays needed by no and local knowledge algorithms. There are a few graphs with only single relays, i.e., the sources that did not have any neighbors. They are depicted in Figure 6.39 as the

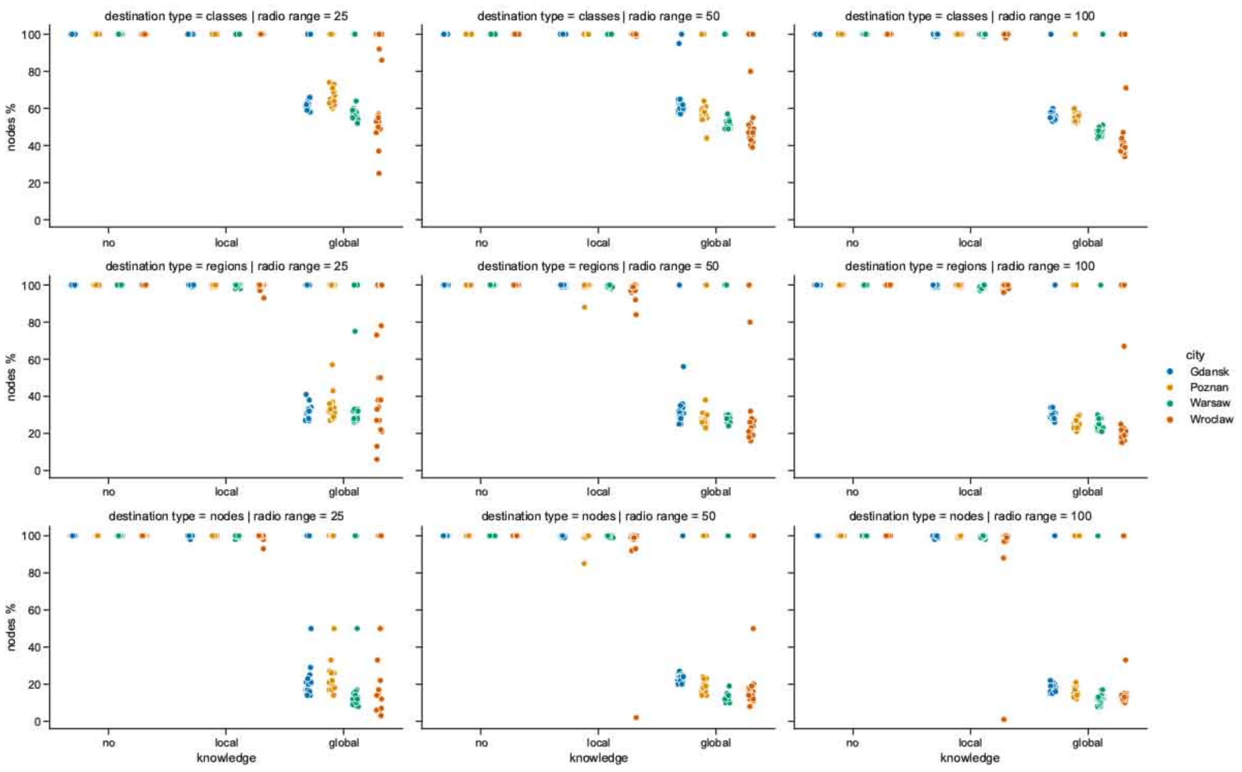


Figure 6.36 Nodes ratio to no knowledge in 30 minutes multicast graphs

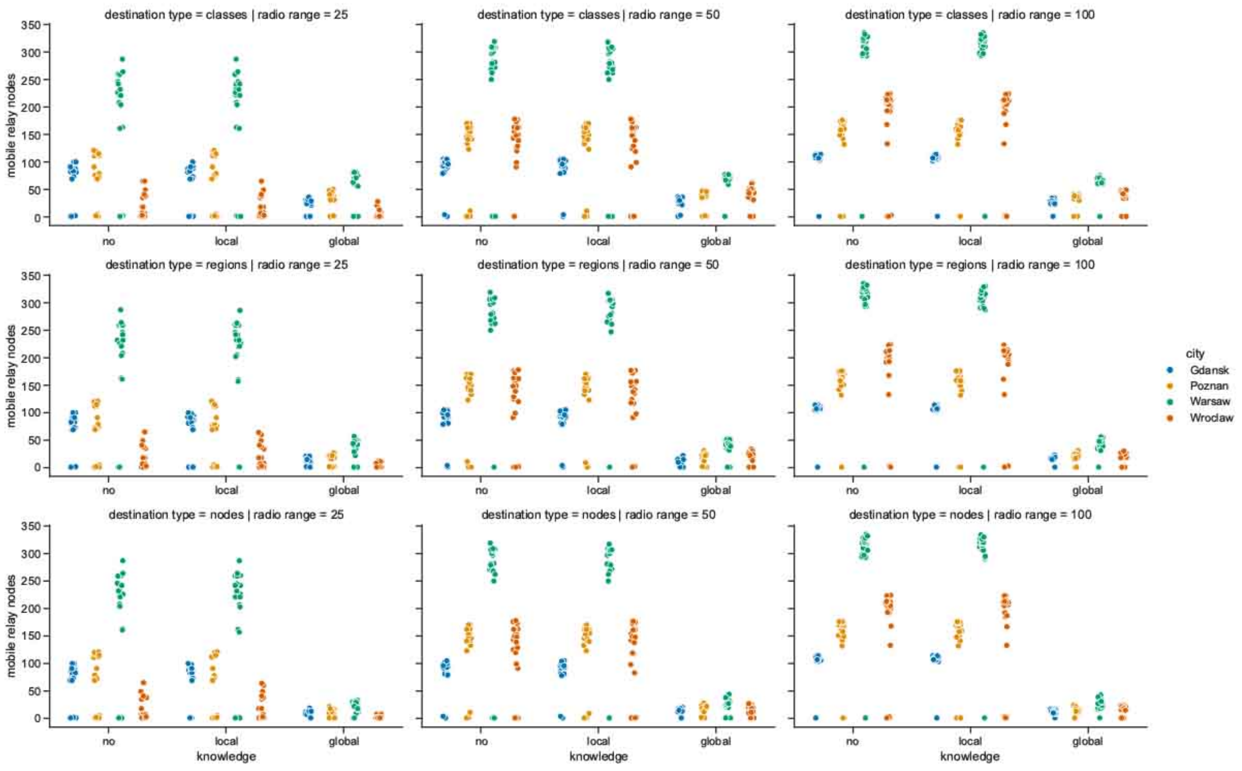


Figure 6.37 Mobile relay nodes number in 30 minutes multicast graphs

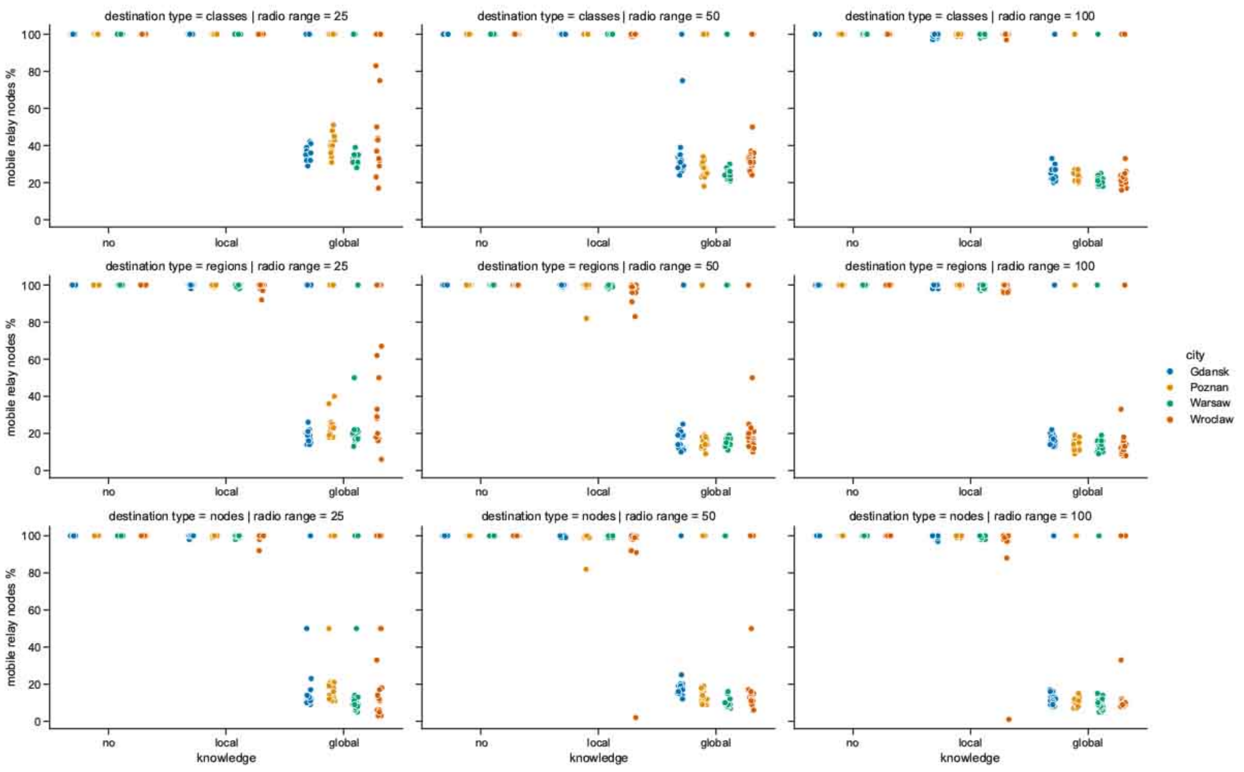


Figure 6.38 Mobile relay nodes ratio to no knowledge in multicast graphs

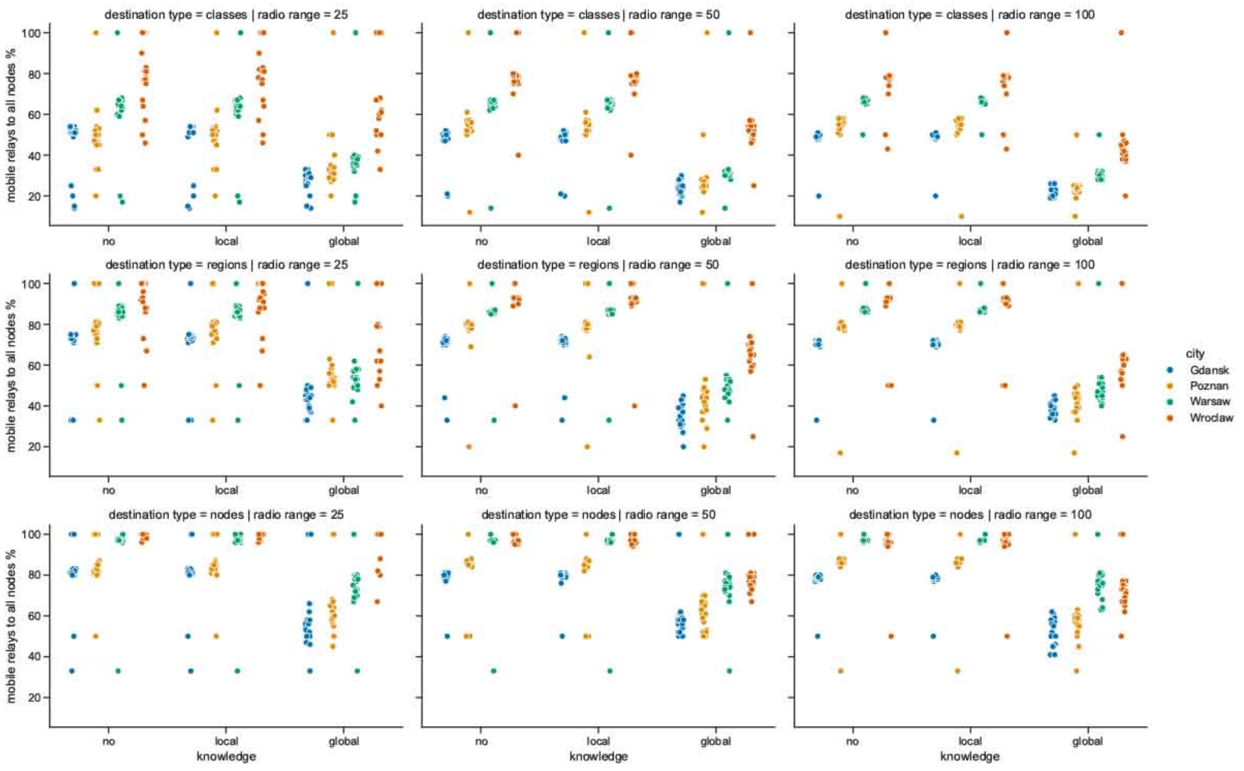


Figure 6.39 Mobile relay nodes to all nodes ratio in 30 minutes multicast graphs

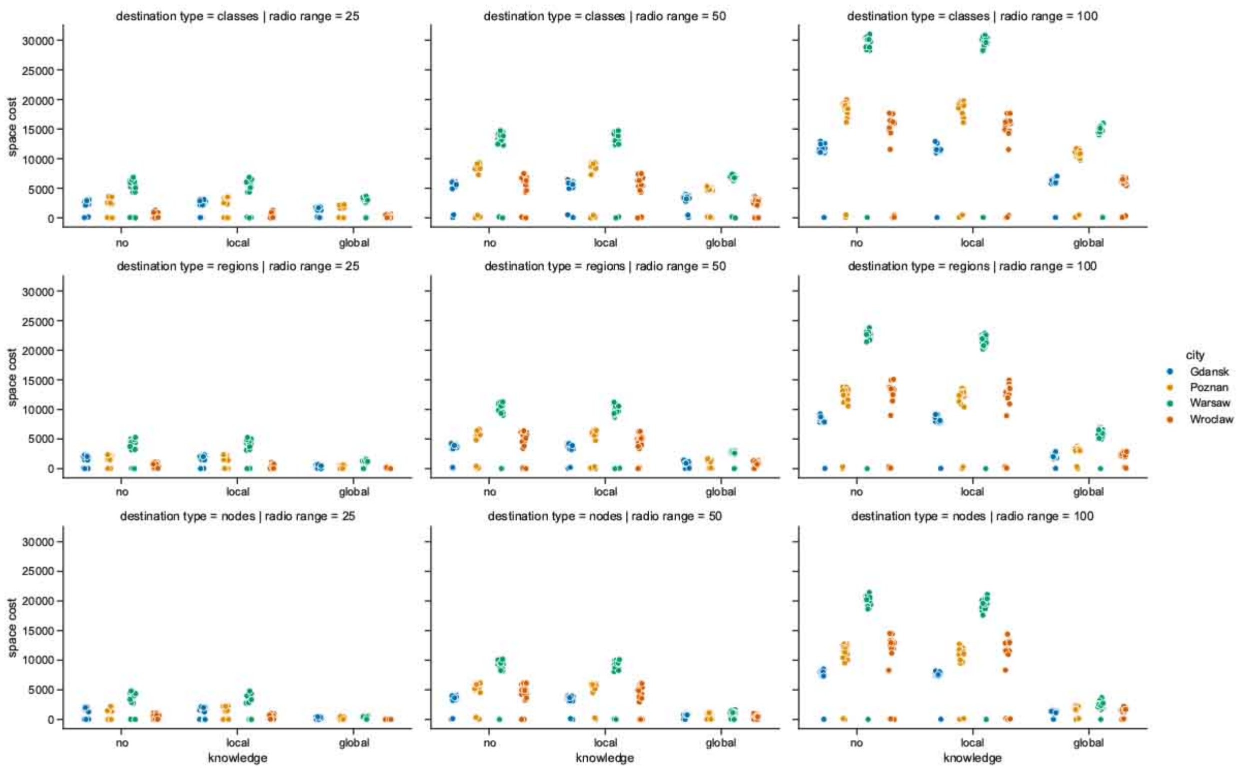


Figure 6.40 Space cost of 30 minutes multicast graphs

points at 100 percent. Otherwise, when there are more nodes in the tree, the percentages of mobile relays are lower, as compared to the numbers of all nodes in the graph. It is visible that Wrocław, despite having the lowest absolute numbers of mobile relays, is the city with the highest percentages of those in the network. Due to the geographical distribution of destination nodes and the routes of the relays it results in the lowest efficacy expressed in Figure 6.35.

6.4.2 Multicast costs

Space costs compared in Figure 6.40 are influenced mostly by the radio range. The larger the radio range, the more nodes can be directly reached, and hence, connected to the multicast structure. As seen in previous metrics, these costs are the lowest in case of global knowledge based routing with the ones of Warsaw being the highest among the cities.

Time costs visible in Figure 6.41 show correlation with the numbers of mobile relay nodes in Figure 6.37. The distributions here are although, in some cases, more vertically spread and shifted, due to the the influence of other attributes of the infrastructure.

Time spans in Figure 6.42 are the same for no and local knowledge based routing methods since they both tend to reach as far in time as possible. This causes the overhead over global knowledge related routing which grows when radio range increases and the number of destination nodes decreases. It can be also investigated in Figure 6.43 to see how much less resources-consuming is the global knowledge approach. There, the sum of all maximum time spans of each node is considered to be the overall time span cost of the structure. In Wrocław this metric is changing

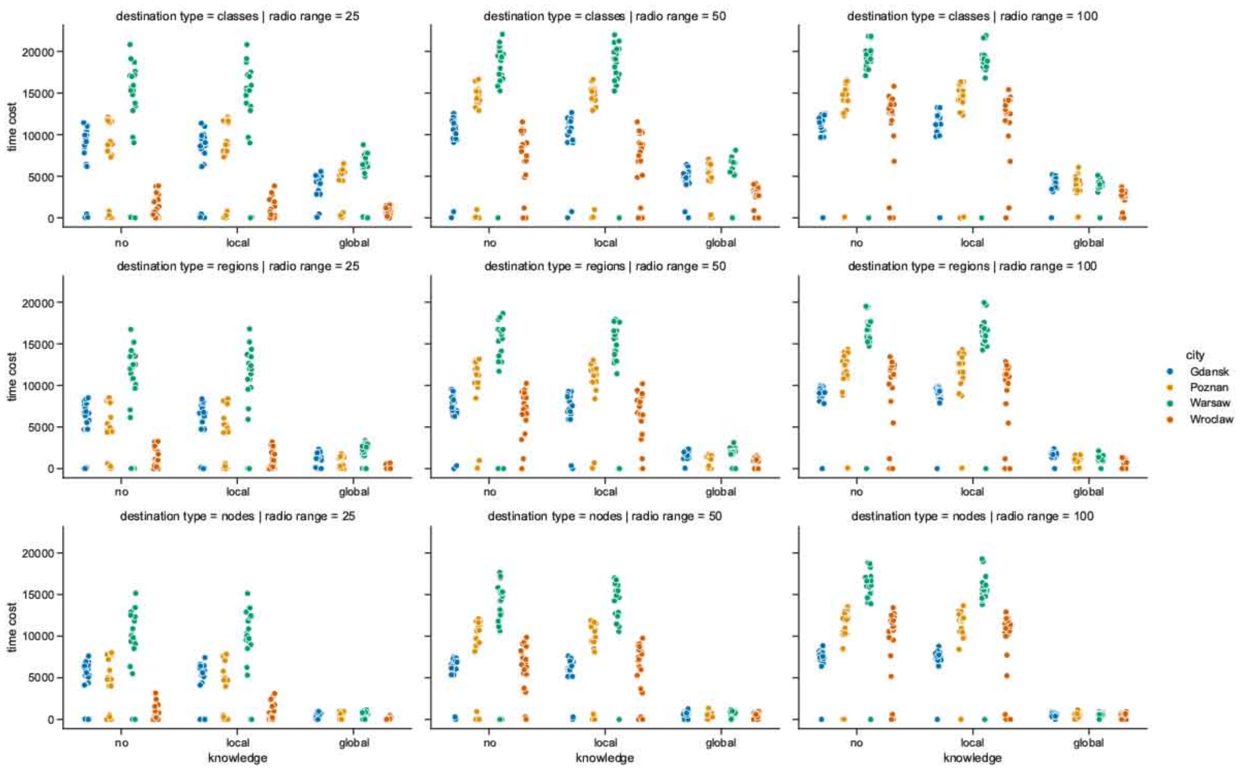


Figure 6.41 Time cost of 30 minutes multicast graphs

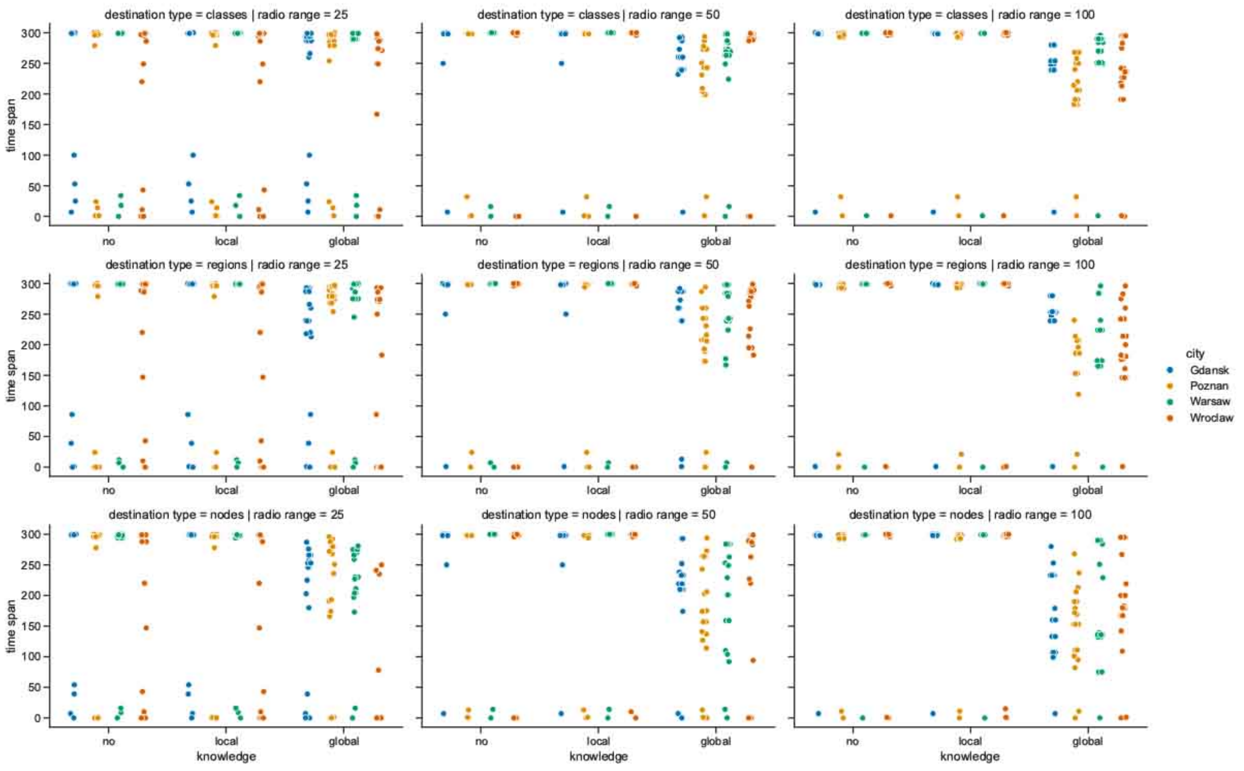


Figure 6.42 Time span of 30 minutes multicast graphs

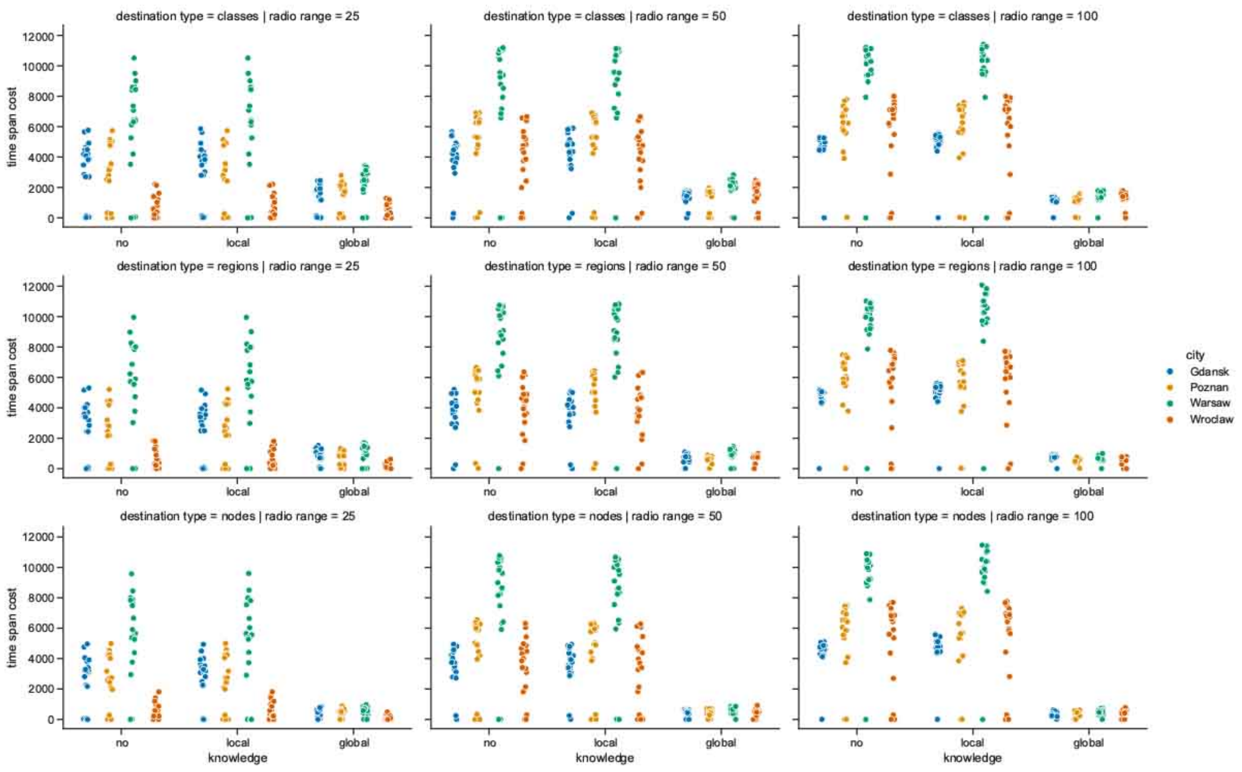


Figure 6.43 Time span cost of 30 minutes multicast graphs

the most when radio range increases. This means that when the range increases, more nodes get connected to the relays which increases time span cost.

6.4.3 Efficacy and efficiency of multicast algorithms

The relative comparisons in Figure 6.44 may seem like the ones related to nodes in Figure 6.38. Upon closer examination particular differences are revealed. Here, it is visible that, in some areas, the usage of local knowledge results in space costs that are slightly lower than when no knowledge is present. Interestingly, there are topologies which result in higher costs, in particular, the ones in Gdańsk. Moreover, the differences increase to as much as about ten percent when radio range is the largest. Space cost with global knowledge is the lowest in general, as could be expected. When it is zero percent it means that no actual multicast tree got constructed because no destinations could be reached. Conversely, when the cost is one hundred percent, the multicast graph consists of only a few edges and nodes—as do the corresponding structures constructed with no and local knowledge.

Although the influence of radio range on time costs in Figure 6.45 is much smaller than on space costs, compare with Figure 6.40, the distributions of relative time cost ratios mirror the ones of relative space costs in Figure 6.44. The groups are shifted towards lower percentages. This means that global knowledge results in multicast structures that are even more efficient in time domain than they are in space domain. It is further visible in Figure 6.46 that presents time span ratios in relation to network knowledge, as compared to no knowledge. The differences span from a few to around seventy percent. This means that multicast structures constructed with global knowledge

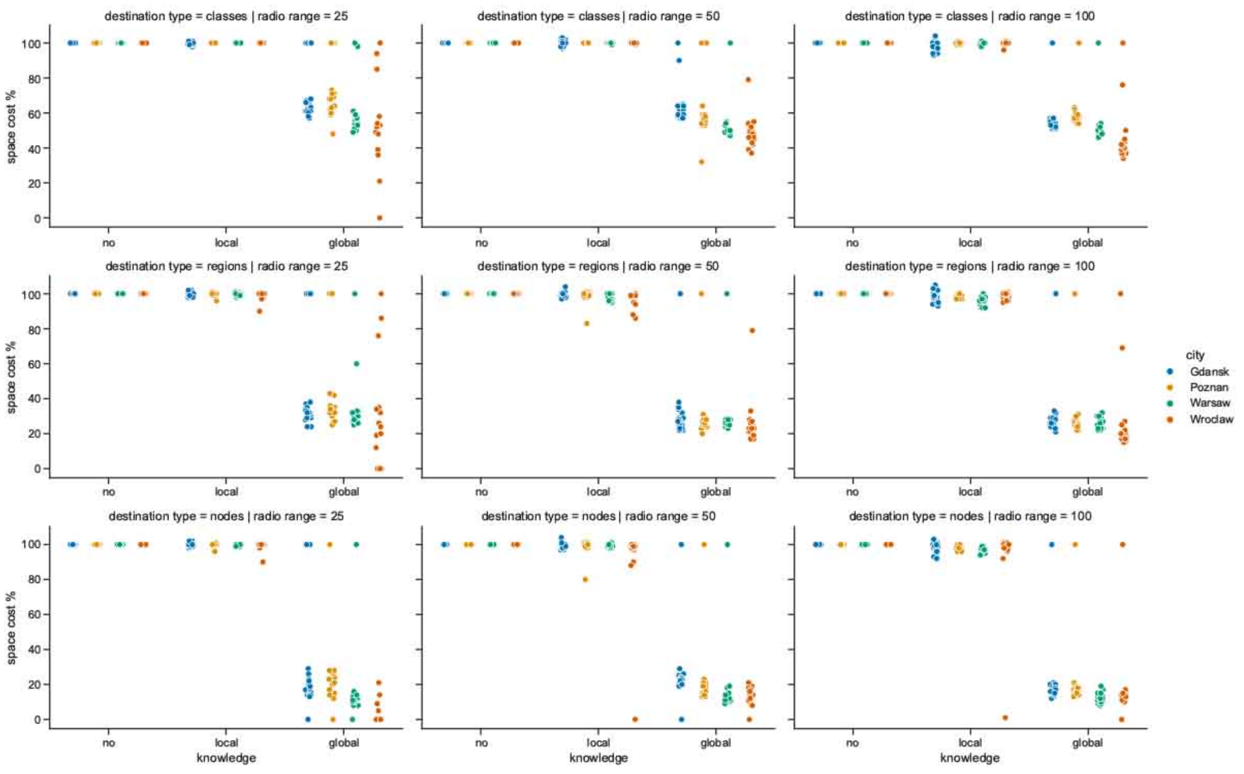


Figure 6.44 Space cost ratio to no knowledge in 30 minutes multicast graphs

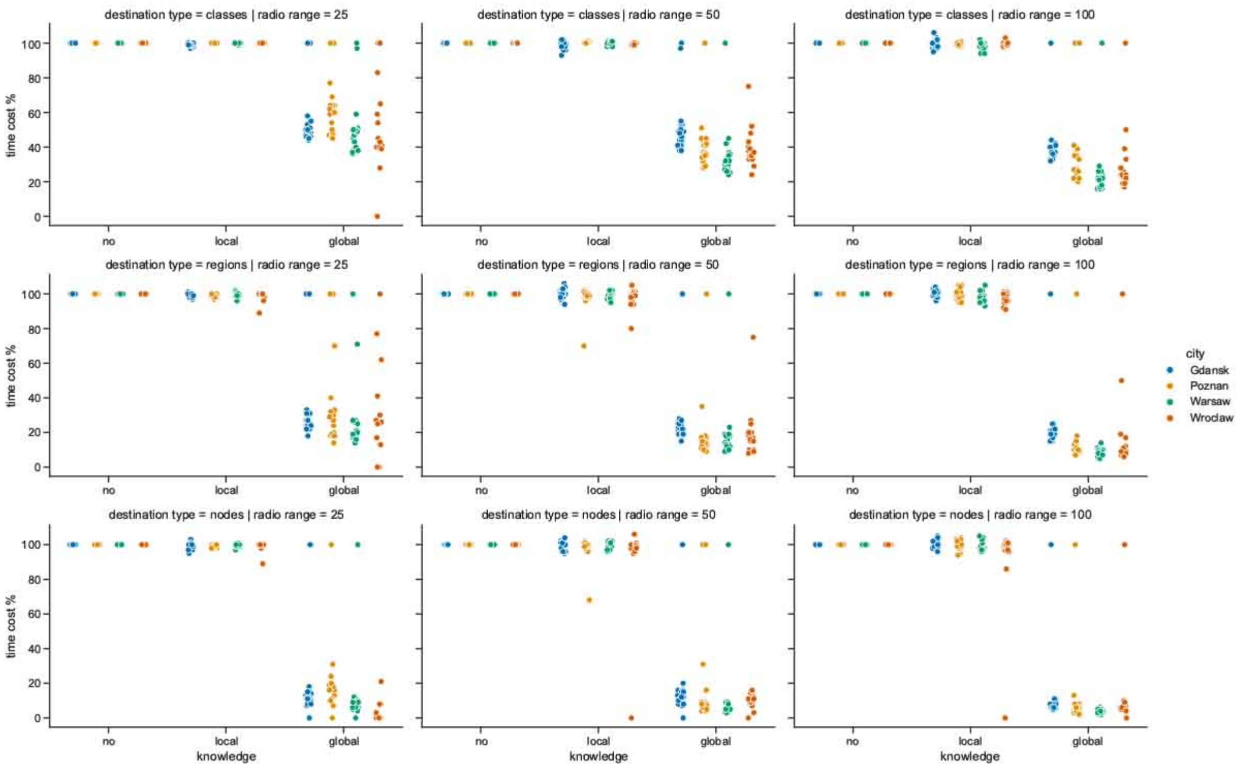


Figure 6.45 Time cost ratio to no knowledge in 30 minutes multicast graphs

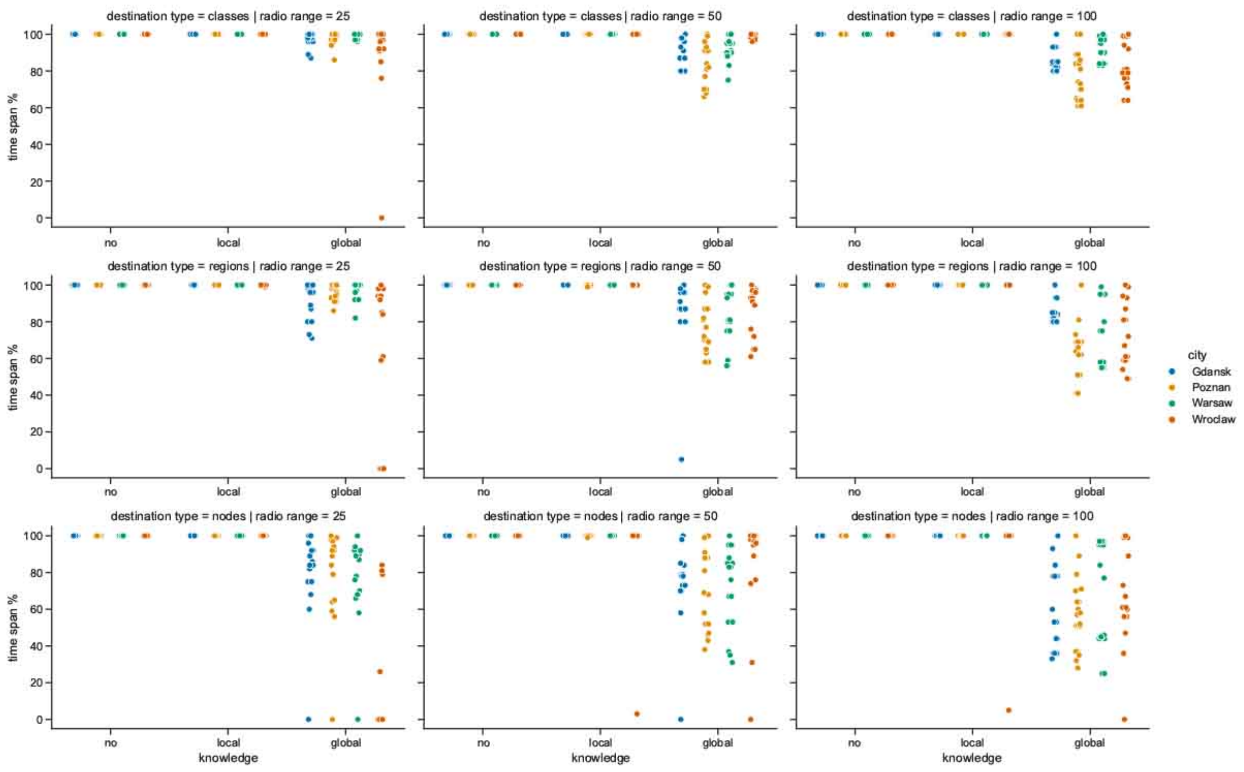


Figure 6.46 Time span ratio to no knowledge in 30 minutes multicast graphs

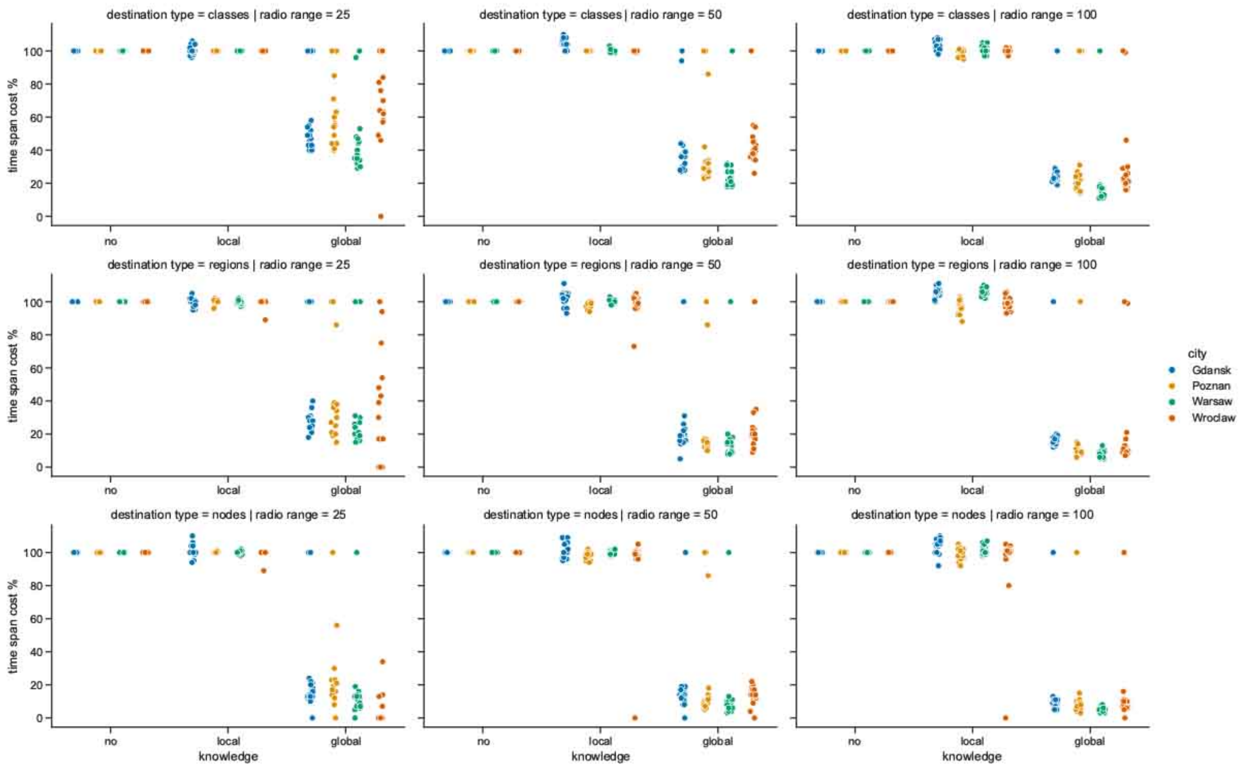


Figure 6.47 Time span cost ratio to no knowledge in 30 minutes multicast graphs

both use significantly less resources in general and require those to be utilized in shorter time periods, on average.

Based on the already analyzed parameters, it could be predicted that time span cost ratios will be the lowest for global knowledge, as presented in Figure 6.47. When radio range increases they tend to get lower, as well as, when the numbers of destination nodes decrease. The local knowledge graph variants diverge from the baseline set with no knowledge—performing slightly better in some topologies and worse in others.

Core multicast structure

The relative distributions of core nodes numbers in Figure 6.48 demonstrate the influence of area the algorithms operate in, and even more, of the number of destination nodes and radio range. For no and local knowledge they are distributed and virtually identical. The numbers for global knowledge are the reference. The structures constructed with global knowledge consist either of only core multicast nodes, i.e., there are no stub relays that do not lead to destination nodes or of no core nodes at all. Core nodes in no and local knowledge variants are, on average, between around 20 and 60 percent of the totality of nodes. The more destination nodes are to be reached, the higher the ratios.

Although the comparisons of core space cost ratios in Figure 6.49 and core time costs ratios in Figure 6.50 closely resemble the one of core nodes ratios in Figure 6.48, they are presented to show that they are highly correlated. Core time cost ratios appear to be somewhat lower, on average,

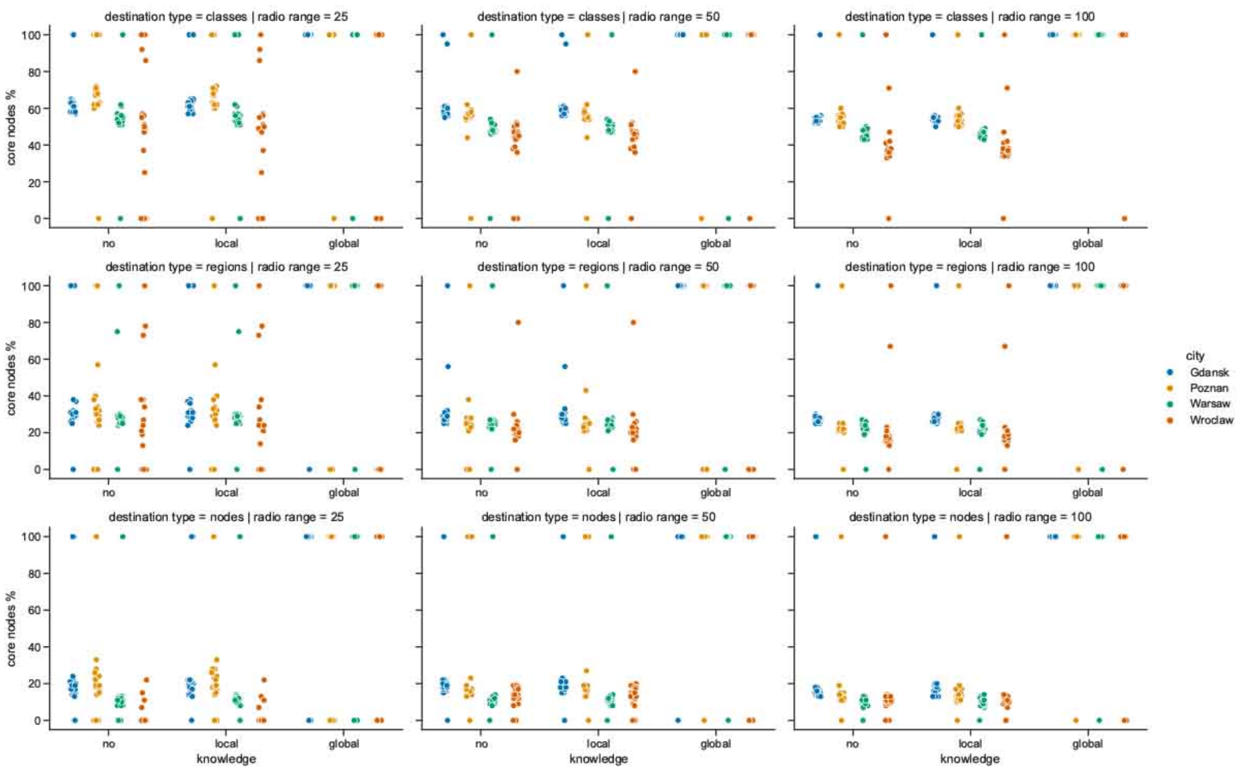


Figure 6.48 Core nodes ratio in 30 minutes multicast graphs

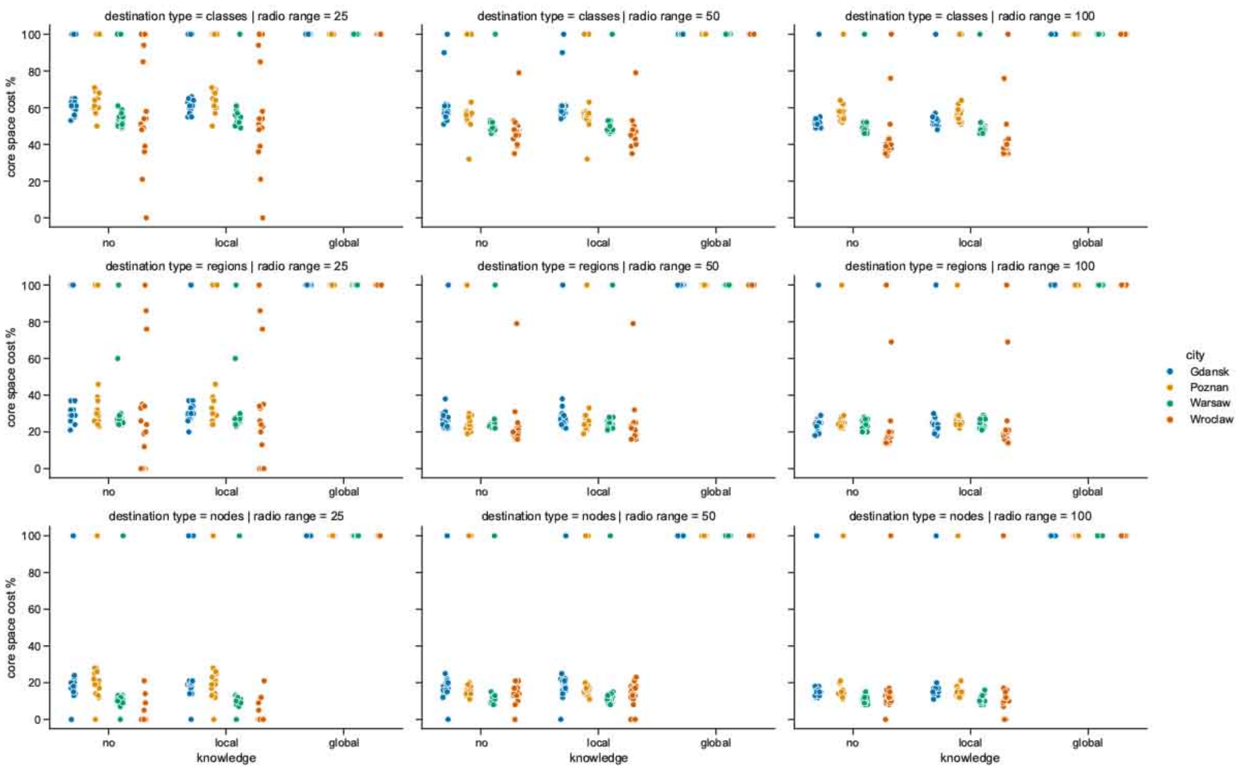


Figure 6.49 Core space cost ratio in 30 minutes multicast graphs

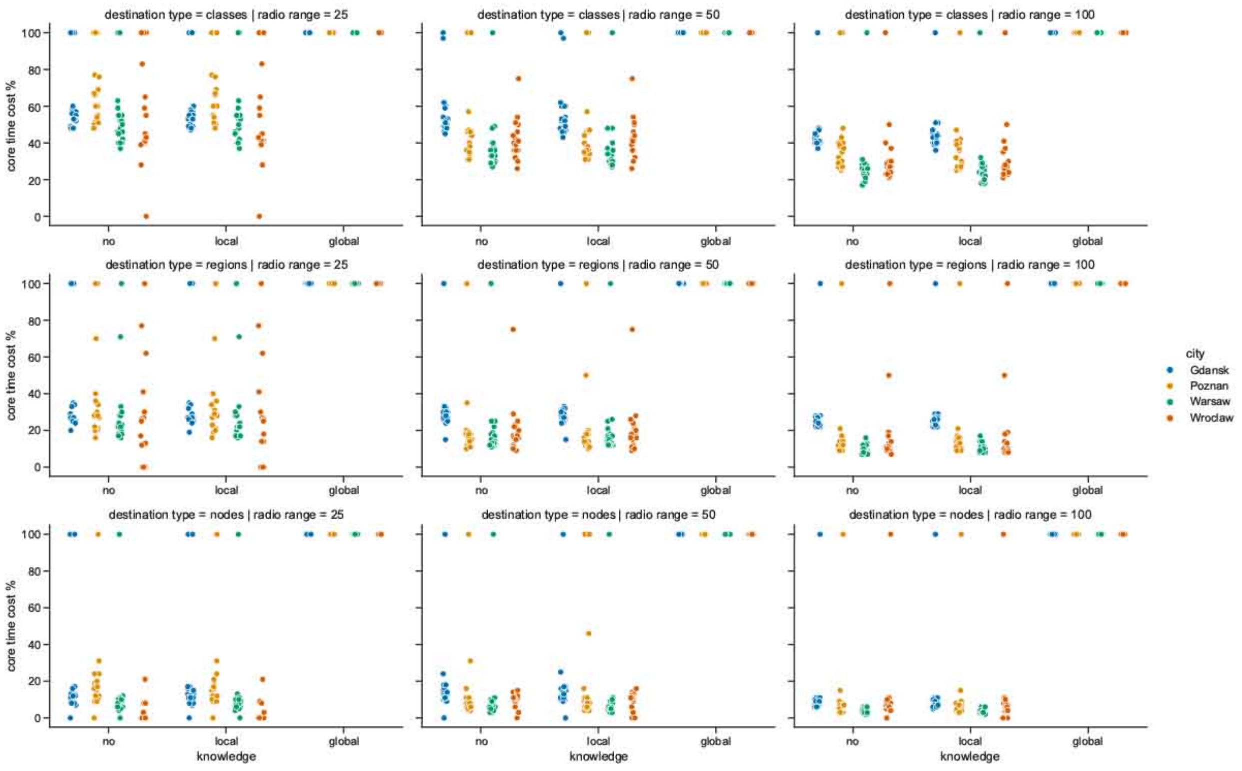


Figure 6.50 Core time cost ratio in 30 minutes multicast graphs

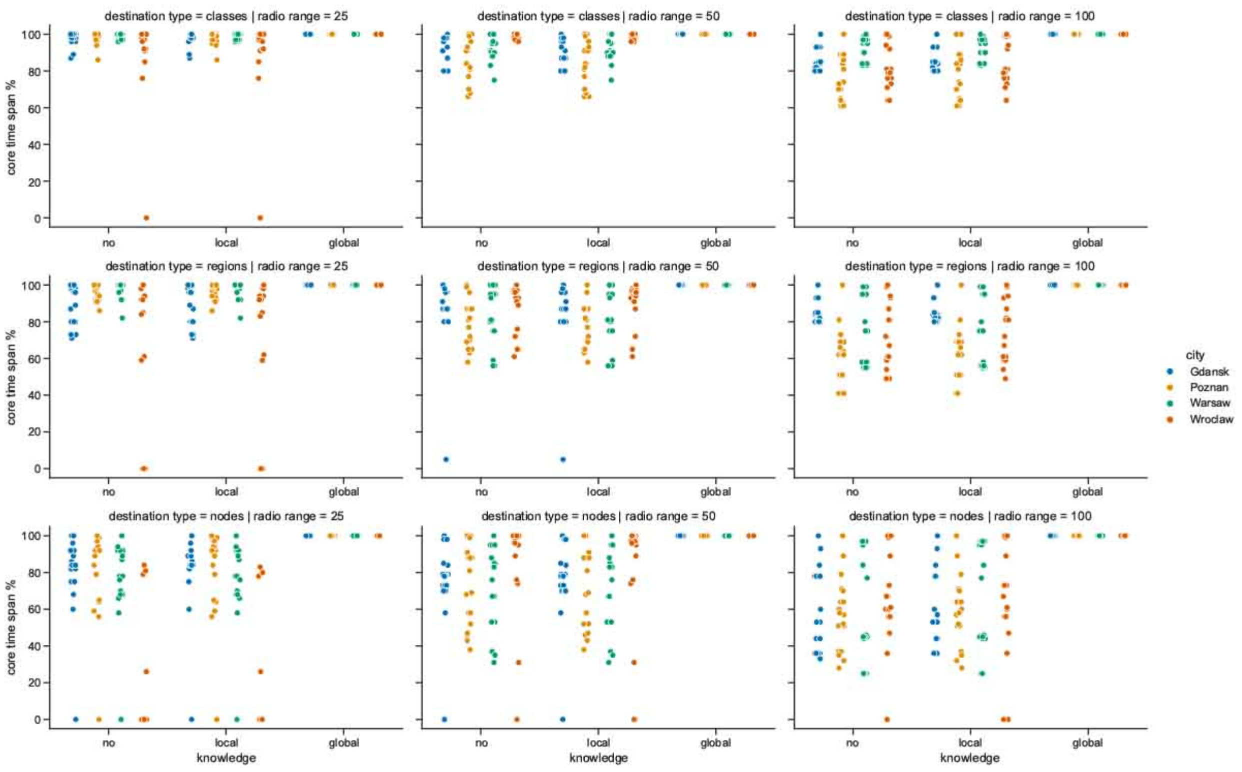


Figure 6.51 Core time span ratio in 30 minutes multicast graphs

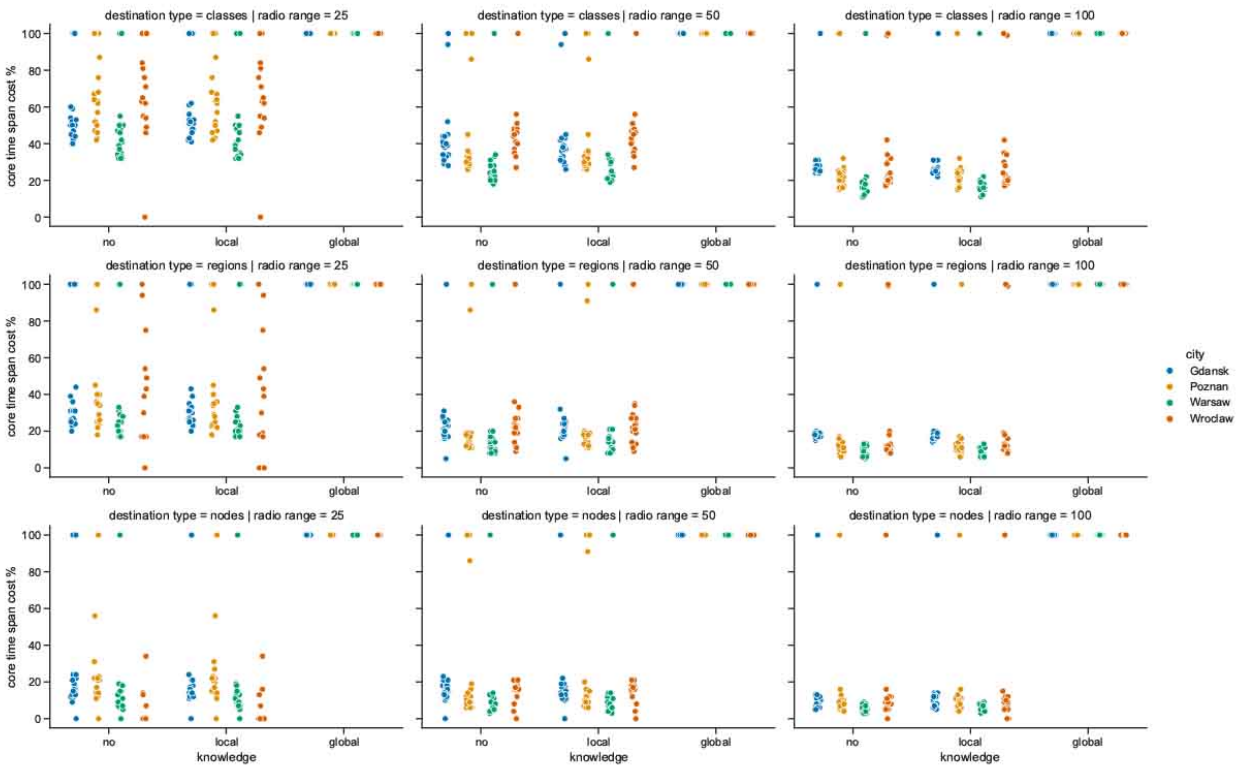


Figure 6.52 Core time span cost ratio in 30 minutes multicast graphs

than the core space cost ratios. It is similar to time cost ratios in Figure 6.45, where the more centrally located groups of values for global knowledge represent on average lower percentages than their counterparts related to space cost ratios.

Time span of the core multicast structure, see Figure 6.51, in case of no and local knowledge ranges between one hundred and eighty percent to even as low as twenty percent. The lower the ratio, the higher the core time span overhead compared to global knowledge. It grows as the radio range increases and is more distributed for lower numbers of destination nodes.

The distributions of core time span cost ratios in Figure 6.52 are most similar to the ones of core time cost ratios in Figure 6.50. Although, they are more dispersed and shifted towards larger values for larger numbers of destinations. With only a few destination nodes an opposite shift direction is visible. With the increase of radio range the average core time cost ratio drops. It decreases even more when the number of destinations goes down with changing destination type from classes through regions to individual nodes. This means that when radio range is the highest and the lowest number of destinations are to be reached, the largest numbers of furthest reaching time edges are redundant since they do not lead to the destinations.

6.4.4 Mobile relay nodes influence

The connectivity, efficacy and efficiency of multicast construction depends heavily on the presence of mobile relay nodes. Therefore, this section discusses the influence of the number of mobile relay nodes on key metrics. To focus the analysis, only the graphs constructed with local knowledge are considered. As already presented, scenarios of no network knowledge result in similar metrics values and distributions. Global knowledge cases are proportionally more effective and less resource consuming, as can be concluded based on Figures 6.38 and 6.39, to construct multicast structures with lower numbers of mobile relay nodes involved.

Space costs diagrams in Figure 6.53 show almost linear increase trend in relation to the number of mobile relay nodes included in multicast graphs. The numbers increase also when radio range and numbers of destination nodes increase, which is accompanied by more visible separate grouping of space costs of different cities. Warsaw is the city of the largest space costs in every scenario. The costs of Poznań are in many cases a little higher than those of Wrocław. In Wrocław they are on average the lowest ones for the lowest radio range and the lowest numbers of mobile relays. When radio range is the highest and the lowest number of destination nodes are to be reached, the cost and numbers of involved mobile relays tend to be usually second largest after Warsaw. Space costs of Gdańsk graphs are the lowest and the most grouped when radio range is the largest.

Time costs also exhibit linearly increasing trend related to the number of mobile relay nodes visible in Figure 6.54. Here, the groups of values related to each city are more easily noticeable. This shows that the increase in radio range causes time costs to be less dispersed for given area and infrastructure. The distributions of time span costs in Figure 6.55 are similar to time cost distributions. The differences are hardly distinguishable for the lowest radio range. For larger ranges the differences become more visible. The obvious distinguishing feature are the absolute values which are almost two times higher in case of time cost.

The relationships between mobile relay nodes and ratios related to the core part of multicast structures with destination regions and local knowledge are presented in Figure 6.56 grouped by radio range. There, the correlation between the ratios of nodes, space costs, time costs and time span costs is clearly visible. The ratios are the most dispersed when only a number of mobile relay nodes are part of multicast graphs. For larger numbers the ratios are more grouped in case of 50 and 100 meters radio range and do not exceed 50 percent. More mobile relays are included in the structures as well. The cost ratios decrease with the increase of mobile relay nodes. Core time span ranges at higher ratio levels, i.e., from about 50 to 100 percent in most cases. The percentage of reached destinations increases almost linearly when the number of mobile relay grows from zero to one hundred nodes with 25 meters radio range. Then it reaches around 75 percent regardless of further increase of the number of mobile relays in Gdańsk, Poznań and Warsaw. When radio range is larger, almost or exactly 100 percent of destinations are reached in those city areas. Despite the ratios in Wrocław being lower, they increase substantially, reaching the level of around 80 percent with the largest radio range.

To highlight the influence of mobile relays in different city areas, the data presented in Figure 6.56 is showed in Figure 6.57 grouped in city-related sets of diagrams. In Gdańsk the number of mobile relays slightly exceeds 100 nodes. The distribution of ratios seems only a little more compressed than in Warsaw, but in fact, it is much more compressed, since in Warsaw the number of nodes spans over three times wider range of mobile relays. In Poznań and Wrocław the full span starting at zero mobile relays can be observed. With more than around ten mobile relay nodes the cost ratios

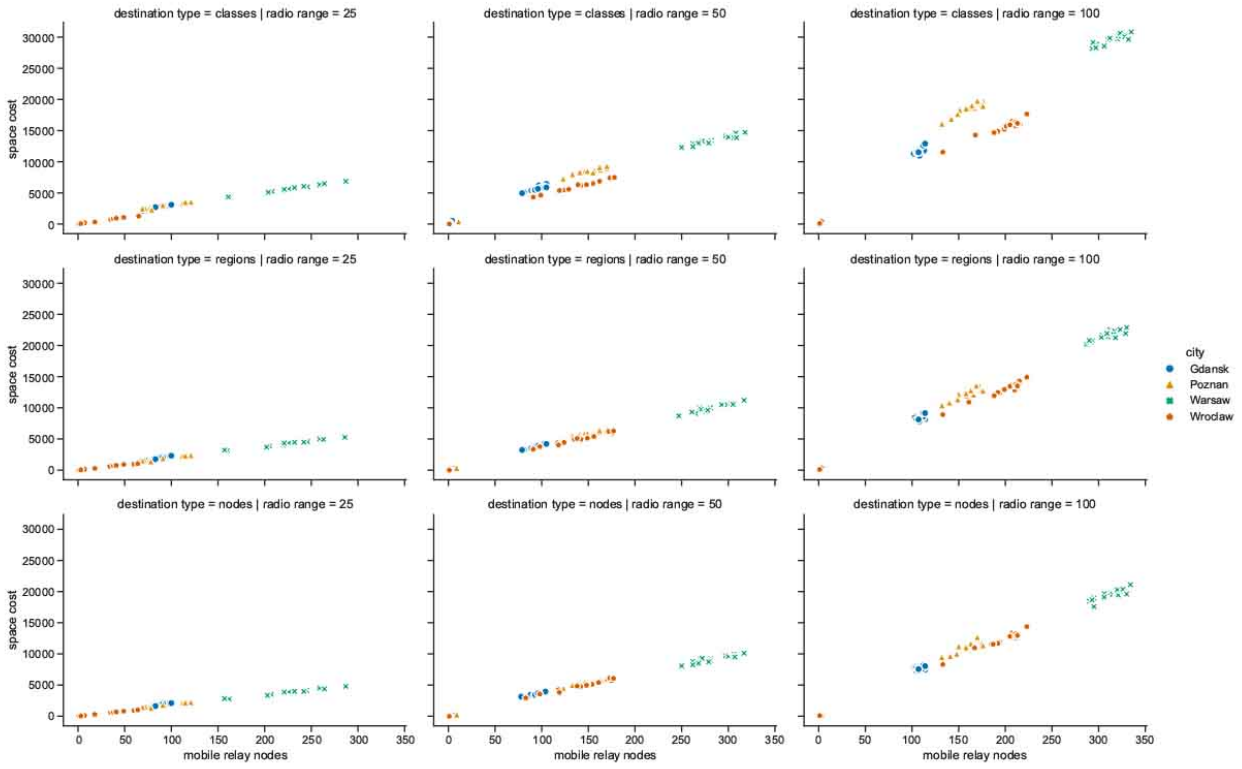


Figure 6.53 Mobile relays influence on space cost of 30 min graphs with local knowledge

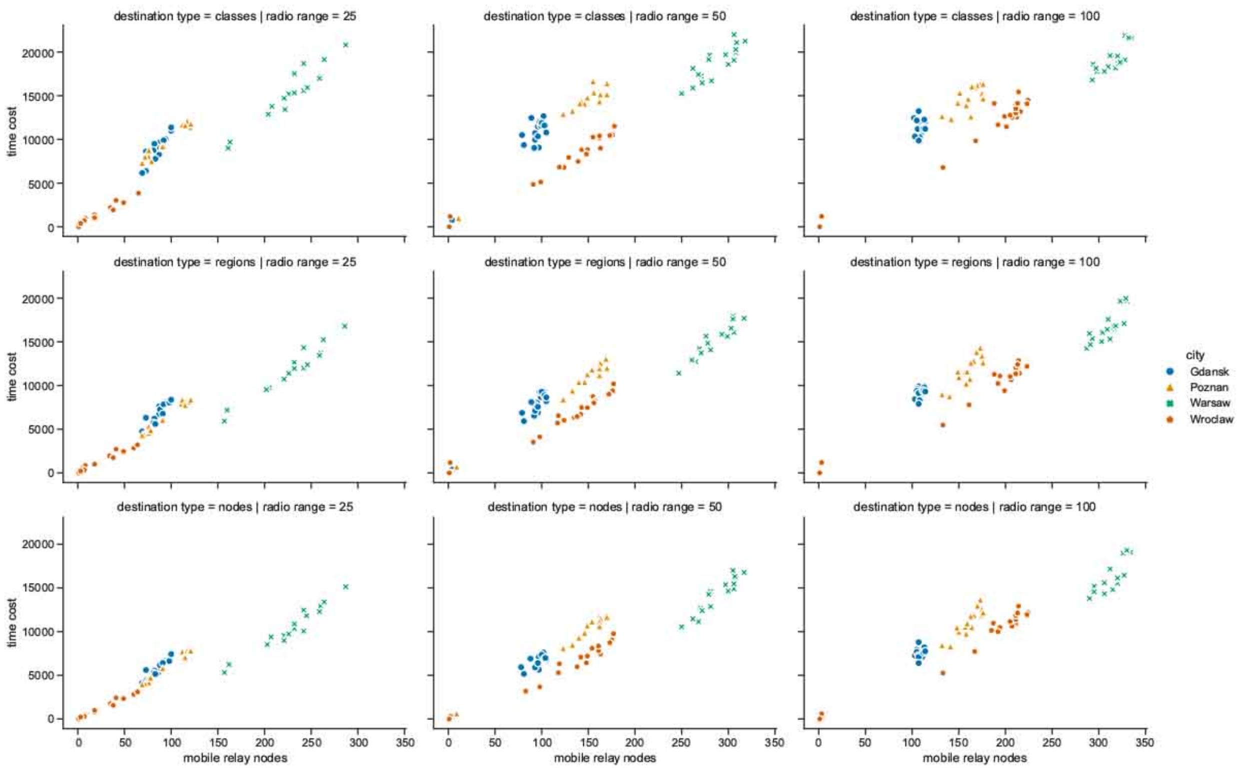


Figure 6.54 Mobile relays influence on time cost of 30 min graphs with local knowledge

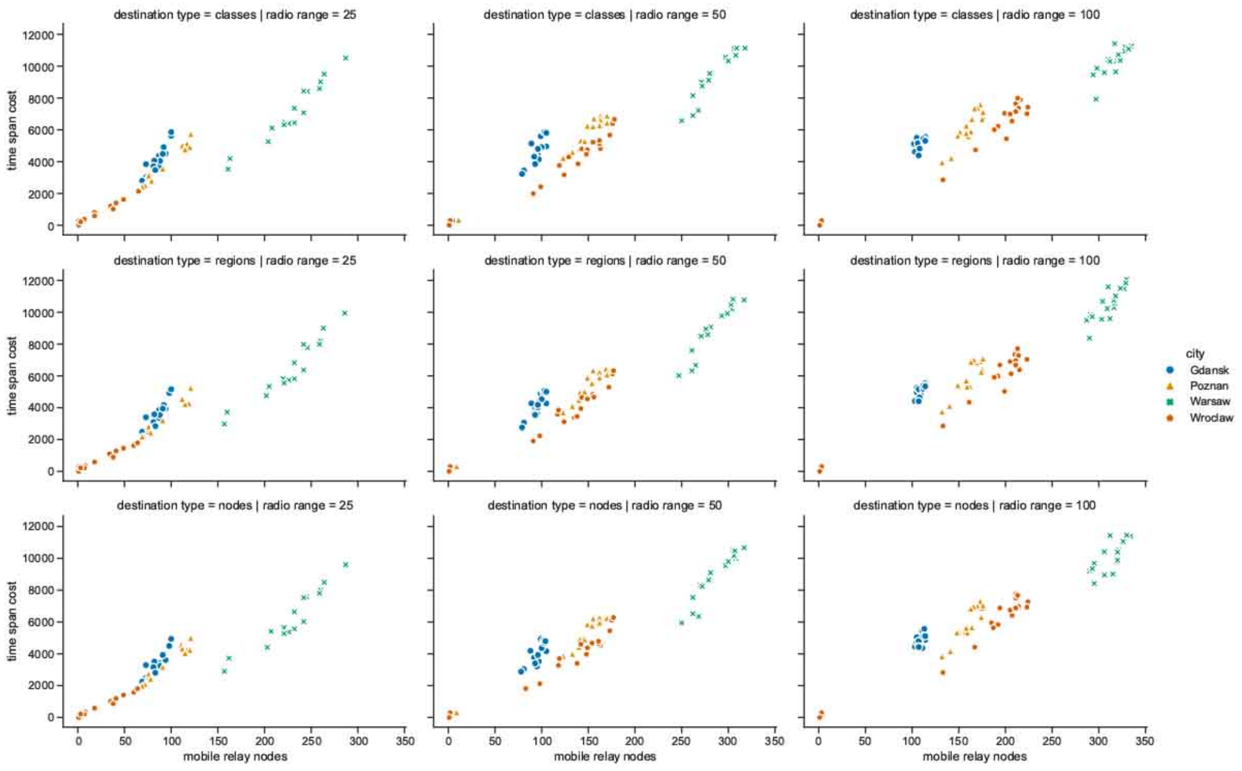
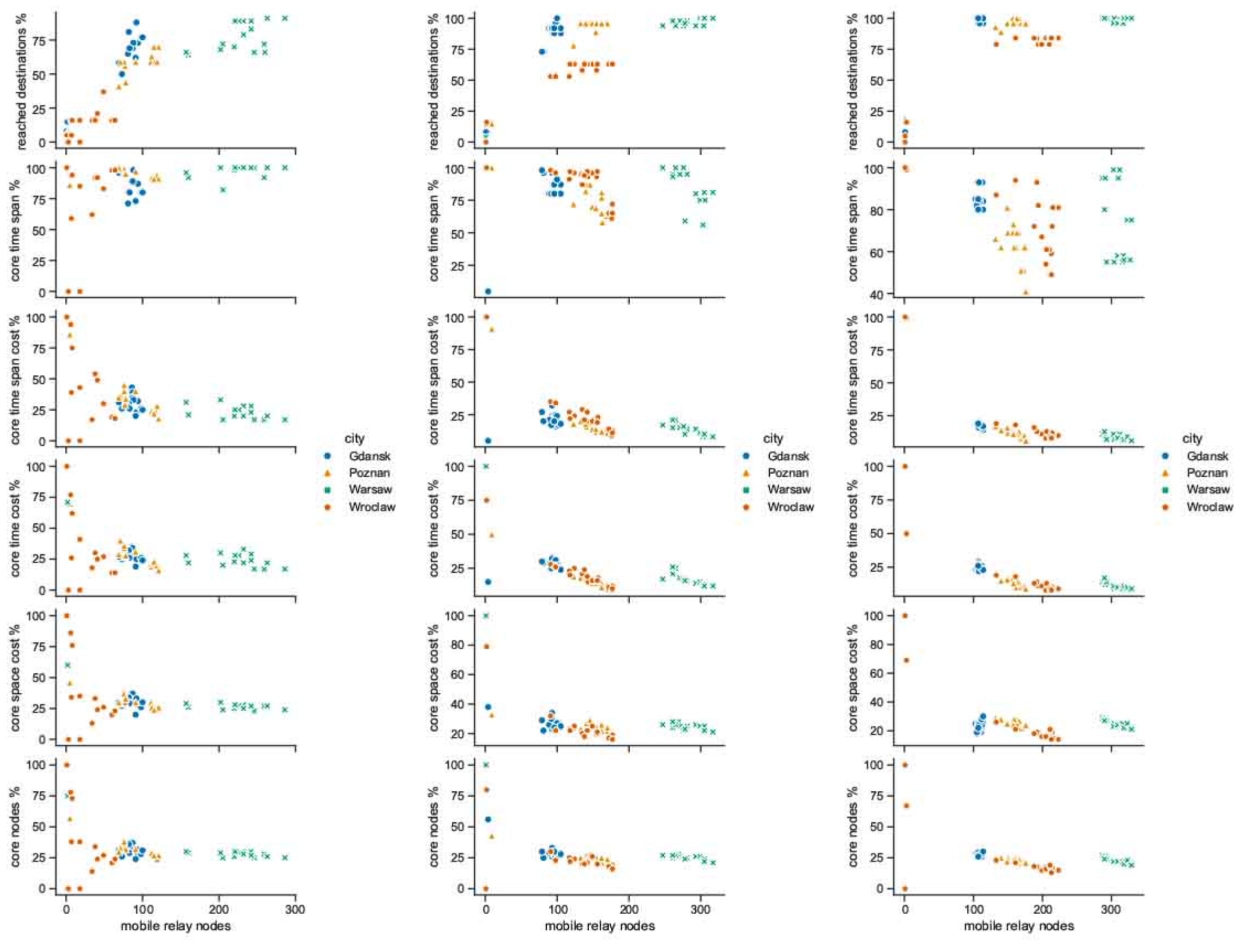


Figure 6.55 Mobile relays influence on time span cost of 30 min graphs with local knowledge

start to follow a decreasing linear trend getting denser with the increase of mobile relays number. It is visible that increase in radio range in all cities causes the increase of mobile relay nodes included into core multicast structure. Moreover, the increase of radio range from 50 to 100 meters brings only a few percent improvement in Gdańsk, Poznań and Warsaw. The same increase in Wrocław results in much higher growth in the percentage of reached destinations starting at around 60 and reaching 80 percent. Core time span ratio appears to be more related to radio range than to the number of mobile relays, decreasing slightly in some cases with the increase of the number of nodes.

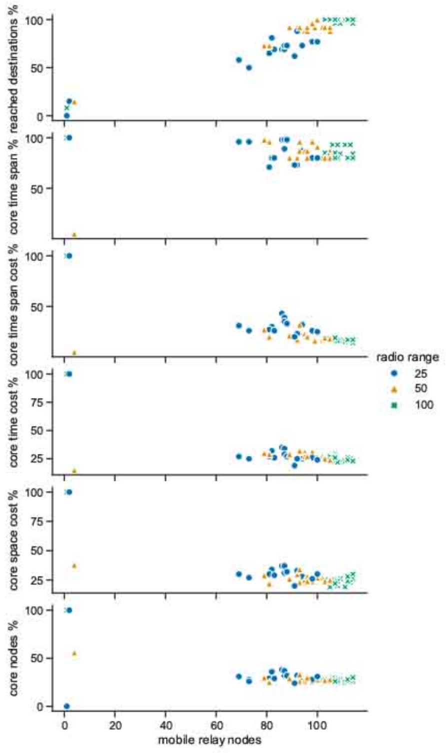
6.4.5 Duration influence

Apart from the choice of the algorithm that will be used, the key controllable factor that influences the efficacy of introduced multicasting methods is the duration of the network, i.e., the maximum life-span of message. While radio range may or may not be controllable by the operator of the network, the desired duration is the aspect that can be directly influenced. Therefore, the most informative performance-related metrics are compared for the areas of interest against both duration

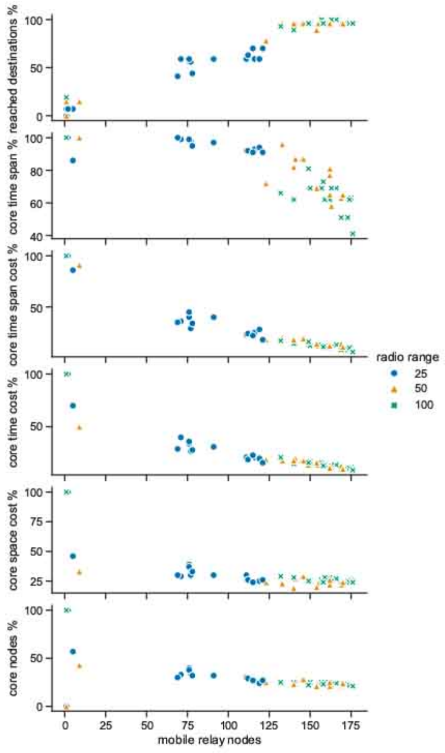


(a) 25 m radio range (b) 50 m radio range (c) 100 m radio range

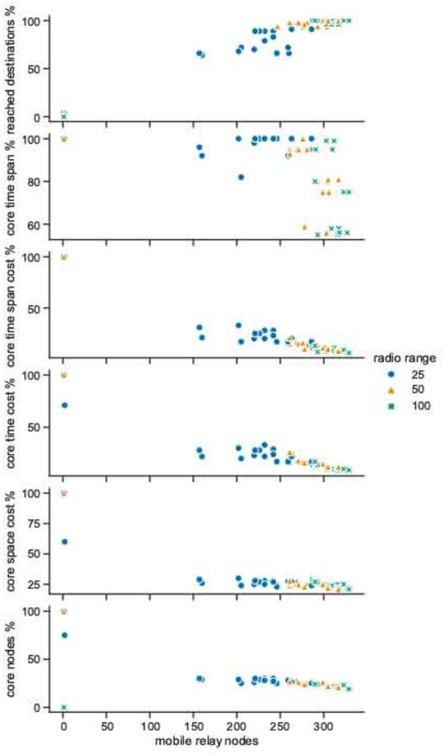
Figure 6.56 Mobile relays and radio range influence on multicast core of 30 minutes graphs with destination regions and local knowledge



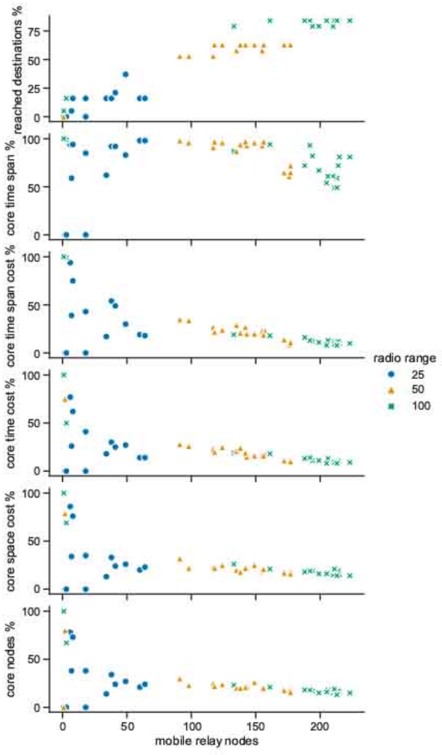
(a) Gdańsk



(b) Poznań



(c) Warsaw



(d) Wrocław

Figure 6.57 Mobile relays and city area influence on multicast core of 30 min graphs

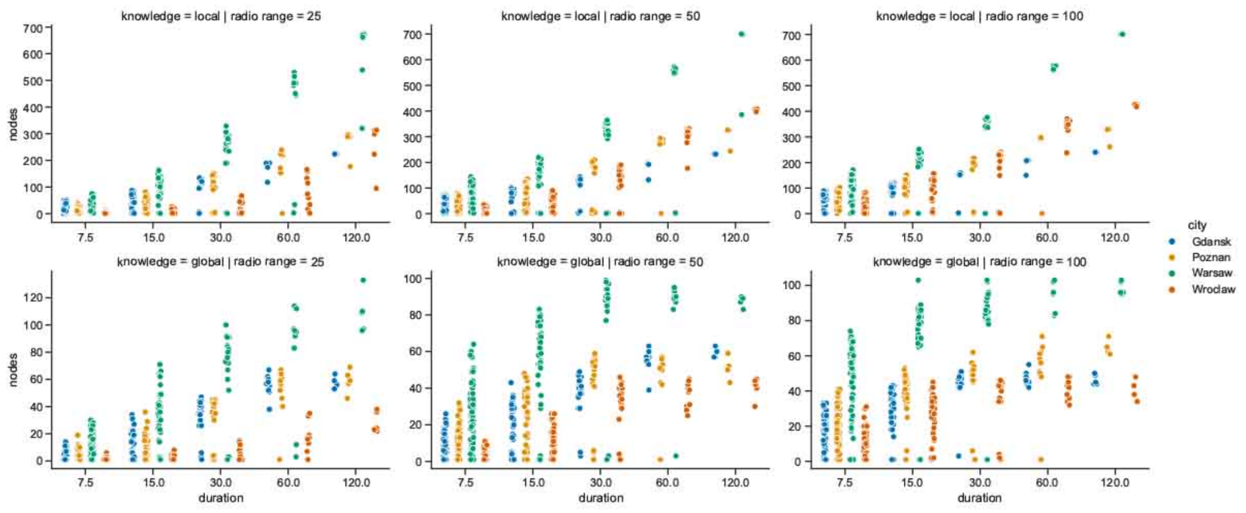


Figure 6.58 All nodes in multicast graphs with destination regions and local or global knowledge

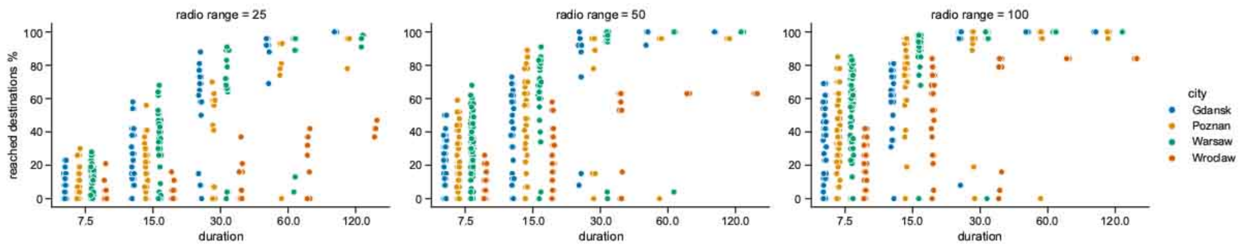


Figure 6.59 Reached to potential destinations ratio in multicast graphs with destination regions and local knowledge

and radio range. They are presented for multicasting with local and global knowledge aimed at destination regions which results in moderate numbers of potential destination nodes. Figure 6.58 shows that not only the numbers of nodes increase with the increase of network duration and radio range but also the values become less spread. It is important to note that Wrocław is the area with the most significant relative node numbers increases when radio range grows. In case of global knowledge the numbers do not increase as dynamically as with local knowledge. Moreover, in all areas they are on average between two to five times lower than when local knowledge is available, and hence, also when no network knowledge is used. It is visible that the increase of duration from 60 to 120 minutes with global knowledge does not translate into significant increases in the numbers of nodes in the multicast structures. When also the radio range is the largest, the numbers start to converge on a steady level already in 30 minutes networks.

As it has been shown in more detail in Figure 6.35, each multicasting knowledge mode results in virtually the same delivery ratios, i.e., the percentages of reached destinations. In Figure 6.59 it is visible that in Gdańsk, Poznań and Warsaw the level of 90 to 100 percent of reached destinations can be achieved in most cases when network duration is at least 60 minutes. When radio range increases the duration of only 30 minutes might also be considered satisfactory in some scenarios. Lower durations result in more distributed, and hence, less predictable delivery ratios. In general, Warsaw

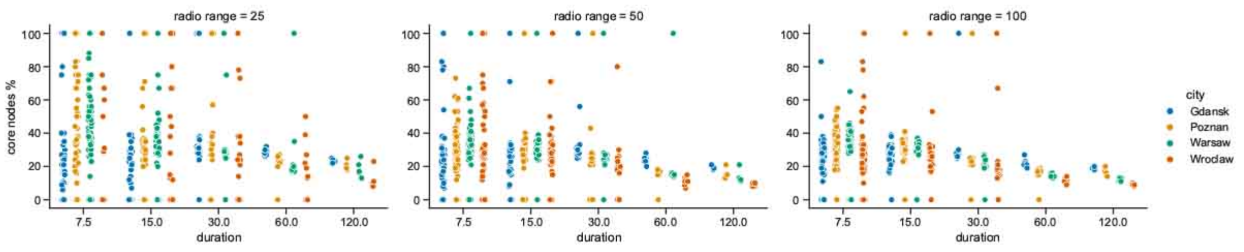


Figure 6.60 Core nodes ratio in multicast graphs with destination regions and local knowledge

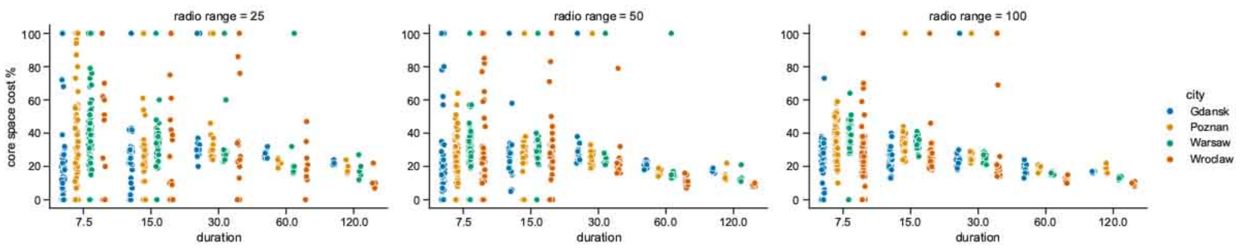


Figure 6.61 Core space cost ratio in multicast graphs with destination regions and local knowledge

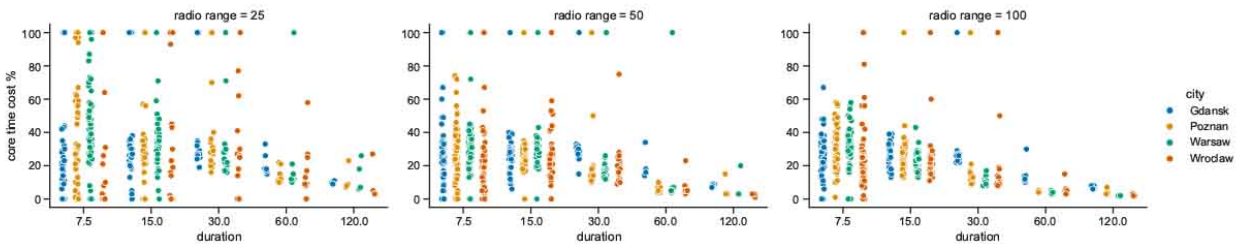


Figure 6.62 Core time cost ratio in multicast graphs with destination regions and local knowledge

is the area that approaches the highest reached destination ratios. Wrocław is staying behind in all variants. Constructed multicast structures do not even reach half of the destinations when radio range is the lowest. When radio range increases to 50 meters, a little over 60 percent of destinations can be reached when the duration is at least 60 minutes. Further radio range growth increases the numbers of reached nodes even at lower durations.

With the shortest network duration in Figure 6.60 the highest and the most distributed numbers of core nodes ratios in local knowledge based graphs can be observed. It means that when the network lasts for the shortest time, and therefore, consists of lower numbers of nodes, the structures are of the highest similarity to the solutions constructed with global knowledge. When duration increases, i.e., the messages are allowed do be distributed in the network over a longer period, the overhead also increases in no and local knowledge variants. It is caused by the flooding-based nature of those approaches. If the duration is set to 30 minutes the numbers of core nodes drop to around 20–30 percent and decline a few more with duration increase. The ratio is zero percent when the graph has no nodes that lead to at least one destination. One hundred percent mean that local knowledge based multicast structure consists only of nodes leading to the destinations. It can happen

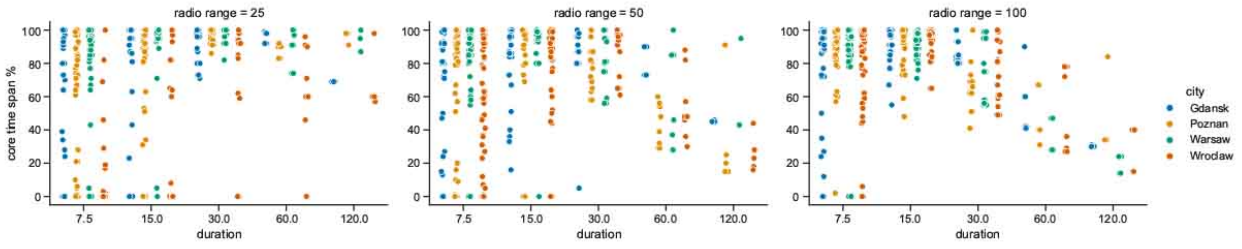


Figure 6.63 Core time span ratio in multicast graphs with destination regions and local knowledge

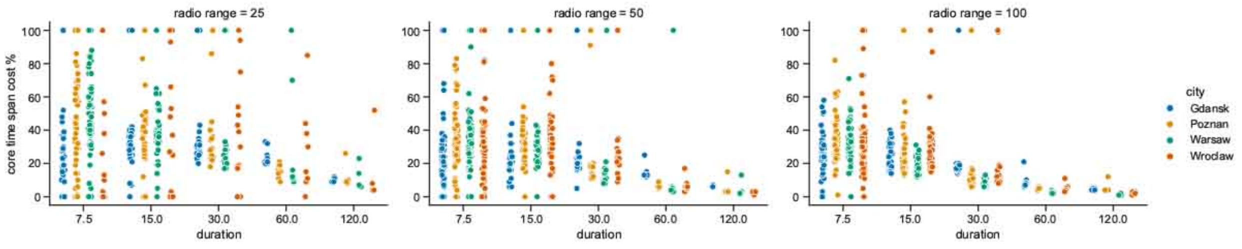


Figure 6.64 Core time span cost ratio in multicast graphs with destination regions and local knowledge

either when the graph is small and reaches only a number of potential destinations or the algorithm managed to reach all of these.

Both the core space cost ratio in Figure 6.61 and core time cost ratio in Figure 6.62 get less vertically distributed when the duration increases. Radio range growth entails the decrease of the part of these costs that are related to core multicast structures built with local knowledge. This trend is visible more in case of core time cost ratio. It informs that when the network is excessively expanded over time, the time cost becomes to be mostly the overhead that could be avoided by tuning the network. This also shows the obvious advantage of using global knowledge when it is available—because then the structure consists only of core nodes.

Core time span ratios in Figure 6.63 inform that with the durations of 30 minutes or lower the core parts of the multicast structure span in most cases entire or almost entire graphs. When the duration and radio range increase it is becoming more and more apparent that the spacial and temporal span of the network becomes excessive. Core time span cost ratios presented in Figure 6.64 express the distributions and trends correlated with the ones of core time cost ratios in Figure 6.62.

6.5 Observations

Presented study used network modeling methods introduced in Section 4 and simulated multicasting algorithms defined in Section 5. The heterogeneous structures were constructed based on open data on the infrastructure and transportation of four Polish urban areas in the cities of Gdańsk, Poznań, Warsaw and Wrocław. No comparable sets of open data sources related to other cities were discovered.

Space connectivity analysis shows that:

- Large-scale network modeling with *space connectivity graphs* (SCG) and multi-dimensional analysis can be conducted based on open data;
- Unique topology and infrastructure features of each urban area influence the networks that get constructed;
- The largest overall numbers of nodes occurred in Warsaw, Poznań, Gdańsk, and Wrocław, respectively. Interestingly, it is a different order than the one of city population densities, i.e., Warsaw, Wrocław, Poznań, and Gdańsk;
- The nodes are the most densely and evenly distributed in Warsaw and Poznań. These areas are also of the largest numbers of stationary destinations. The highest numbers of stationary relays are present in Gdańsk and Poznań;
- Mobile relays occur in the areas of interest in diverse numbers and distributions. The largest numbers of mobile relay nodes were recorded in Warsaw, followed by Poznań, Wrocław, and Gdańsk. When compared with the overall number of public transport day routes, which is the highest in Warsaw, then Wrocław, Gdańsk, and Poznań, an important conclusion can be drown. Not only the number of routes influences the numbers of modeled mobile nodes but also travel frequency, street topology and stops density;
- Different percentages of mobile nodes were present in the cities. In Warsaw and Wrocław the ratios as high as 30 percent were observed on average, compared to the total numbers of nodes. A several less mobile relays were present in Poznań, with only a few percent traversing the area in Gdańsk;
- Changing radio range of the nodes influences modeled networks to different extent, depending on node distribution. Instantaneous topologies are more partitioned for lower radio ranges. In more fragmented areas, i.e., Gdańsk and Wrocław, the increase of radio range is required to get more nodes connected. Network structures modeled in Wrocław are the most sparse locally, i.e., express the lowest numbers of nodes per connected component. The numbers of components, are though, the lowest in Gdańsk and the highest in Warsaw;
- Radio range has an effect on the costs of space connectivity graphs and related minimum and maximum spanning forests. It is shown that when radio coverage increases also the numbers of edges and costs of the structures increase. These increases are of exponential nature, and therefore, radio parameters of designed networks have to be planned with care. In this way, overutilization of the wireless medium, as well as, computing and storage resources of the nodes can be avoided;
- The area in Poznań is the easiest one to achieve high level of connectivity at the lowest radio ranges and cost—due to significant number of stationary relays and high density of routes of numerous mobile nodes.

Space-time connectivity analysis proves that:

- *Space-time connectivity graphs* (STCG) which consist of as many as hundreds of thousands of nodes can be modeled based on *space connectivity lists* (SCL) and used as the intermediate structure in space-time network modeling;
- *First-contact graphs* (FCG) are valuable compact-form indicators that capture how space-time networks develop over time and how the adjacencies occur for the first time;
- The structures get more complex when network duration and radio coverage grows. It means that topology and connectivity increase as well;
- The sets of mobile nodes present in the areas change over time—some enter and some leave at different moments. In a space-time connectivity graph there are more unique space nodes (individual devices) than in a single space connectivity graph. As a result, the numbers of mobile relays in first-contact graphs are at least a few times higher than in related space connectivity graphs;
- The highest numbers of mobile nodes are present in STCGs related to Warsaw. Poznań pursues with more mobile nodes than the ones modeled for Wrocław. The situation is quite the opposite in case of FCGs in these two cities, which is caused by mobile nodes playing more centralized role in the connectivity modeled in Wrocław. Gdańsk is the area of the lowest numbers of mobile relay nodes;
- The highest numbers of instances of mobile relay nodes occur in Poznań and Wrocław. It means that mobile relays are on average present for the longest total time in these areas;
- The numbers of connected components in FCGs in Gdańsk, Poznań, and Warsaw are on average lower than they are in SCGs. The disparity increases with the increase of network duration and radio range. Similar trend can be noticed in Wrocław, however, there is a difference. The shortest durations and radio ranges result in the numbers of connected components exceeding the ones for SCGs. It suggests, that the infrastructure is more fragmented in Wrocław, and hence, more time is required to achieve higher network connectivity;
- The average node degrees and the numbers of connected components are much higher in first-contact graphs than they are in space connectivity graphs. This proves that using mobile relays to construct space-time networks enables the momentarily disconnected parts of the network to be connected over time. As a result, larger and denser time-spanning topologies are constructed.

Space-time multicast analysis indicates that:

- in general:
  - Algorithms designed for *delay tolerant multicast router* (DTMR) can be used in multicast communication in urban environment and enable modeling of related graph structures;

  - The efficacy and efficiency of multicast algorithms are related to the presence, number, distribution, and movement of relay nodes;

- in terms of structural complexity:
  - In some topologies, the randomly selected mobile source does not establish any space connections over time, and therefore, is unable to relay the message to other nodes. There are also some multicast trees composed only of a handful of nodes. Although, the most of modeled graphs consist of significantly more nodes;
  - The increase of radio range causes more nodes to be directly reached as the members of multicast tree. Although, even at lower radio ranges, space-time multicast graphs of complex structure and reach can get constructed;
  - The area in Wrocław is the one with the most significant relative node numbers growth when radio range increases;

- in terms of mobile relays:
  - The highest numbers of mobile relay nodes in multicast graphs were observed in Warsaw, followed by Poznań or Wrocław, with the fewest present in Gdańsk. In spite of having the lowest absolute numbers of mobile relays, Wrocław is the city with the highest percentages of these in time-spanning topologies;
  - Global knowledge based multicasting approach utilizes on average only around 10 to 60 percent of mobile relays involved when no and local knowledge algorithms are used. The percentage of used mobile relay nodes usually declines when radio range grows;

- in terms of costs:
  - Space costs are the lowest in case of global knowledge based routing. The ones of Warsaw are the highest among them in the areas of interest. These costs increase almost linearly with the increase of the number of mobile relays. Also, the larger the radio range, the higher the space cost of multicast graphs. It indicates the related increasing power use caused by radio transmissions;
  - Time costs are positively correlated with the numbers of mobile relays. Therefore, the more mobile relays involved, the more computing resources of the network will be used;
  - Time spans are equal for no and local network knowledge routing methods because they both propagate the message as far in time as possible. This is the reason for the overhead over global knowledge based routing. It grows when radio range increases and the number of destination nodes decreases. Global knowledge approach can be significantly less resources-consuming and minimize the overall use of storage capacity of the relays;
  - Time span cost grows when radio range increases because more nodes get connected to single relays. When this cost gets higher, message buffers of the relays are occupied longer;

  - Time cost and time span cost become mostly the overhead when no and local knowledge networks are excessively expanded over time;

- in terms of performance:

  - No and local network knowledge result in multicast structures of identical or similar graph parameters. Out of the two, local knowledge routing should be preferred when available and lower utilization of radio communication medium is desired;
  - The graphs constructed with no and local knowledge consist of, on average, between around 40 and 80 percent of stub relays that do not lead to multicast destination nodes. This is due to greedy opportunistic network flooding-related nature of the algorithms;
  - Global knowledge multicasting involves, in general, significantly fewer relay nodes and uses them shorter. Transmission medium usage is also expected to be lower. Therefore, the metrics get to be as low as 10 to 80 percent of those for no and local knowledge, depending on the scenario and metric. The difference grows when the number of destination nodes decreases in relation to the total number of nodes in the network;
  - Each network knowledge mode results in practically the same delivery ratios, i.e., the numbers of reached multicast destinations;
  - Radio range of 50 meters and maximum message distribution time (duration) set to 30 minutes by the means of using the *time-to-live* (TTL) parameter can be considered as the probable optimal settings for investigated urban areas. When higher delivery ratio is required or area of interest gets vaster, the duration has to be extended or the range (coverage) increased, when feasible.

In total, over 60 thousand graphs were built and enabled the study of modeled network topologies and multicast communication structures that use both stationary and uncontrolled mobile relay nodes. Well-known and newly-introduced graph-theory metrics were presented. Key features, trends, and relationships were analyzed, compared and discussed showing the usability of introduced algorithms.

Chapter 7

Summary

The evolution of wireless sensor network concept in relation to structures of sensing capabilities deployed in city space creates new challenges and opportunities. For this reason, the types and characteristics of actual urban sensor networks are discussed in this dissertation. Crucial routing research problems are identified as opportunistic routing, data aggregation, data offloading, with the fundamental issue being topology control and modeling. Practical implications are discussed based on professional and research experience of the author. It is indicated that implemented networks tend to be based on ready-made and easily available components and proven systems, not necessarily and solely designed for typical sensor networks. Therefore, research should be conducted in close relation to technological advancement.

The types of node deployment schemes observed in urban environment are analyzed. It is shown that, despite of the fact that topology control can be executed as a complex design problem, in actual networks it is usually reduced to, first and foremost, node deployment. In practice, the predominant model of implementation appears to be empirical, i.e., draws past experience, from experiments, and best practices, while more advanced aspects are omitted. As a result, deterministic and random node distribution types can be distinguished.

Investigation of emerging urban node location data sources leads to the conclusion that they can be used in sensor network modeling. As a result, innovative data gathering, processing, and opportunistic network modeling architecture and algorithms are presented. The method is proved feasible and practical using open data. Presentation of this novel approach opens up new possibilities for the research of key topology and routing problems in real-life urban sensor networks. Now, these networks can be modeled and studied as momentary and time-spanning graphs based on exact time-changing locations and attributes of large numbers of heterogeneous sensing devices.

Presented delay tolerant multicast algorithms, aimed at heterogeneous urban network structures, are designed with practicality in mind. They are protocol agnostic, modular and introduced in a form that should be convenient to understand and implement. The high-level aspects of network utilization and delivery ratio, computing power, storage and wireless medium usage are considered, while their details are pushed aside to focus the matter on routing aspects. The methods are presented as the family of related modular components and procedures. The key differentiators are the usage of no, local, or global topology knowledge available to the nodes, as well as, the definition of multicast destinations as single nodes, particular geographical regions, or selected node classes. In this way, multiple multicasting scenarios can be addressed and the new ones may be developed, if the need arises. Furthermore, a generalized model of a complete delay-tolerant routing process implementation is presented in the form of a technology-agnostic router.

Multi-dimensional analysis of introduced algorithms is conducted based on publicly available data for four Polish cities. They are studied in custom-developed simulation research environment related to presented network modeling architecture. More than 60 thousand graphs are constructed and examined with numerous metrics. The most significant characteristics, trends, and relationships are discussed. Moreover, it is shown, that each urban area of interest has its own unique features that influence the resulting network connectivity graphs and related time-spanning multicast trees. Also, the presence of mobile relays is clearly visible as the key factor which influences network connectivity with store-carry-and-forward functionality, bridging (connecting) the otherwise disjoint network areas. Similarly, the ability to use global or local topology knowledge positively affects overall network utilization, at the expense of more capabilities and message processing resources required in involved nodes, as compared to no network knowledge scenarios.

Main contributions of presented work are as follows:

1. Overview of actual urban sensor networks types and empirical analysis of node deployment schemes;
2. Method for conversion of gathered infrastructure data into network knowledge:
   (a) characteristics of data sources;
   (b) network modeling architecture based on real-life open data;
   (c) spatiotemporal network visualization method;
3. Urban sensor network graph-based modeling algorithms:
   (a) space connectivity modeling;
   (b) space-time connectivity modeling;
   (c) first-contact modeling;
4. Urban delay tolerant multicast algorithms using uncontrolled mobile relays:
   (a) no knowledge opportunistic multicast;
   (b) local knowledge opportunistic multicast;
   (c) global knowledge opportunistic multicast;
5. Simulation study of introduced models and algorithms:
   (a) methodology for multidimensional modeling and analysis;
   (b) simulation architecture and custom-developed research environment;
   (c) comparative investigation and observations for four Polish cities.

Suggested directions for further research include the following areas:

1. Urban sensor network modeling:
   (a) Development of a reliable parametric network topology generator based on long term observations of open data and timetables;
   (b) Construction of movement traces based on gathered node location open data;
   (c) Use of introduced modeling architecture as an element of stationary relays deployment planning;
   (d) Use of presented modeling algorithms in other fields:
      i. social trends analysis;
      ii. multi-criteria route optimization for public service vehicles;
      iii. urban planning of bicycle paths, street infrastructure etc.;
2. Multicast using uncontrolled mobile relay:
   (a) Introduction of real-time fallback mechanisms for global and local network knowledge algorithms, e.g., to switch from global to local knowledge routing when globally determined or predicted path proves to be unavailable while routing;
   (b) Modeling the usage of acknowledgments and retransmissions;
   (c) Implementation of advanced buffer and duty-cycle management, as well as, radio spectrum and power-preserving schemes;
   (d) Making routing decision in relation to node movement direction and speed, as well as, the history of previous and predicted contacts;
   (e) Introduction of machine learning based optimizations;
3. Graph-based study of presented algorithms:
   (a) Investigation of extended scale and scope:
      i. more data sources, including the closed ones, when available;
      ii. more areas of different locations and sizes, e.g., suburbs, small cities, countrysides, etc.;
      iii. different periods of interest;
      iv. various shapes of destination regions;
   (b) Advanced radio connectivity modeling:
      i. use of actual connectivity data, when available;
      ii. use of complex radio coverage models;
   (c) Different sets of multicast parameters and destinations:
      i. solution-related node classes and roles;
      ii. regions-based multicast to nodes of desired classes;

4. Network traffic flow-based study of introduced multicast algorithms:
    (a) Analysis based on the usage of discrete-event simulation:
        i. data routing in a network simulator with radio propagation, communication and power consumption models (e.g., OMNeT++ [210] or ns-3 [211]);
        ii. node movement determined by a road traffic simulator (e.g., SUMO [212]) exposed by a vehicular network simulation framework (e.g., Veins [213]);
        iii. use of an opportunistic DTN simulator (e.g., The ONE [214]);
    (b) Implementation and verification of the algorithms in a *software-defined network* (SDN) testbed.

As summarized, new algorithms were introduced that enable modeling of multicast communication in urban wireless sensor networks in which uncontrolled mobile relay node is used. Therefore, the objective of the dissertation was achieved. Moreover, novel opportunistic network modeling and investigation approach was developed and proved valid in implemented research environment. Directions for further research are suggested as well.

References

[1] B. Musznicki. Empirical Approach in Topology Control of Sensor Networks for Urban Environment. *Journal of Telecommunications and Information Technology*, (1):47–57, 2019.

[2] B. Musznicki, M. Piechowiak, and P. Zwierzykowski. Modeling Real-Life Urban Sensor Networks Based on Open Data. *Sensors*, 22(23), 2022.

[3] B. Musznicki and P. Zwierzykowski. Survey of Simulators for Wireless Sensor Networks. *International Journal of Grid and Distributed Computing*, 5(3):23–50, September 2012.

[4] K. Holger and A. Willig. *Protocols and Architectures for Wireless Sensor Networks.* John Wiley & Sons Ltd., Chichester, West Sussex, England, 2005.

[5] I. Akyildiz and M. C. Vuran. *Wireless Sensor Networks.* John Wiley & Sons Ltd., Chichester, West Sussex, England, August 2010.

[6] L. Atzori, A. Iera, and G. Morabito. From “Smart Objects” to “Social Objects”: The Next Evolutionary Step of the Internet of Things. *IEEE Communications Magazine*, 52(1):97–105, January 2014.

[7] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys Tutorials*, 16(1):414–454, 2014.

[8] M. Díaz, C. Martín, and B. Rubio. State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. *Journal of Network and Computer Applications*, 67:99–117, 2016.

[9] R. Faludi. *Building Wireless Sensor Networks: with ZigBee, XBee, Arduino, and Processing.* O’Reilly Media, Inc., Sebastopol, CA, USA, 2010.

[10] S. Ferdoush and X. Li. Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications. *Procedia Computer Science*, 34:103–110, 2014.

[11] F. Leccese, M. Cagnetti, and D. Trinca. A Smart City Application: A Fully Controlled Street Lighting Isle Based on Raspberry-Pi Card, a ZigBee Sensor Network and WiMAX. *Sensors*, 14(12):24408–24424, 2014.

[12] C. P. Kruger, A. M. Abu-Mahfouz, and G. P. Hancke. Rapid prototyping of a wireless sensor network gateway for the internet of things using off-the-shelf components. In *IEEE International Conference on Industrial Technology (ICIT 2015)*, pages 1926–1931, March 2015.

[13] A. D. Deshmukh and U. B. Shinde. A low cost environment monitoring system using raspberry Pi and arduino with Zigbee. In *International Conference on Inventive Computation Technologies (ICICT 2016)*, volume 3, pages 1–6, August 2016.

[14] M. Głąbowski, B. Musznicki, P. Nowak, and P. Zwierzykowski. An In-depth Discussion of Challenges Related to Solving Shortest Path Problems Using ShortestPathACO Based Algorithms. In J. Świątek, L. Borzemski, A. Grzech, and Z. Wilimowska, editors, *Information Systems Architecture and Technology; Knowledge Based Approach to the Design, Control and Decision Support*, pages 77–88. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, Poland, 2013.

[15] A. Gupta and R. K. Jha. A Survey of 5G Network: Architecture and Emerging Technologies. *IEEE Access*, 3:1206–1232, 2015.

[16] K. Kowalik, D. Dudek, M. Kołodziejski, B. Musznicki, E. Grzybek, and J. Jarzina. Lessons Learned from WiMAX Deployment at INEA. *Journal of Telecommunications and Information Technology*, (3):34–41, September 2014.

[17] B. Musznicki, K. Kowalik, P. Kołodziejski, and E. Grzybek. Mobile and Residential INEA Wi-Fi Hotspot Network. In *13th International Symposium on Wireless Communication Systems (ISWCS 2016)*, Poznań, Poland, 20–23 September 2016.

[18] A. Kliks, B. Musznicki, K. Kowalik, and P. Kryszkiewicz. Perspectives for Resource Sharing in 5G Networks. *Telecommunication Systems*, 11 December 2017.

[19] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[20] C. P. Kruger and G. P. Hancke. Implementing the Internet of Things Vision in Industrial Wireless Sensor Networks. In *12th IEEE International Conference on Industrial Informatics (INDIN 2014)*, pages 627–632, July 2014.

[21] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.

[22] A. Kos, V. Milutinović, and A. Umek. Challenges in wireless communication for connected sensors and wearable devices used in sport biofeedback applications. *Future Generation Computer Systems*, 92:582–592, 2019.

[23] L. Manjakkal, S. Mitra, Y. R. Petillot, J. Shutler, E. M. Scott, M. Willander, and R. Dahiya. Connected Sensors, Innovative Sensor Deployment, and Intelligent Data Analysis for Online Water Quality Monitoring. *IEEE Internet of Things Journal*, 8(18):13805–13824, 2021.

[24] T. Tang and A. T.-K. Ho. A path-dependence perspective on the adoption of Internet of Things: Evidence from early adopters of smart and connected sensors in the United States. *Government Information Quarterly*, 36(2):321–332, 2019.

[25] M. S. Sheikh, J. Liang, and W. Wang. A Survey of Security Services, Attacks, and Applications for Vehicular Ad Hoc Networks (VANETs). *Sensors*, 19(16), 2019.

[26] F. Kurugollu, S. H. Ahmed, R. Hussain, F. Ahmad, and C. A. Kerrache. Vehicular Sensor Networks: Applications, Advances and Challenges. *Sensors*, 20(13), 2020.

[27] Z. Afzal and M. Kumar. Security of Vehicular Ad-Hoc Networks (VANET): A survey. *Journal of Physics: Conference Series*, 1427(1):012015, jan 2020.

[28] A. Rahim, Z. Khan, F. T. B. Muhaya, M. Sher, and T.-H. Kim. Sensor Based Framework for Secure Multimedia Communication in VANET. *Sensors*, 10(11):10146–10154, 2010.

[29] M. Noor-A-Rahim, Z. Liu, H. Lee, M. O. Khyam, J. He, D. Pesch, K. Moessner, W. Saad, and H. V. Poor. 6G for Vehicle-to-Everything (V2X) Communications: Enabling Technologies, Challenges, and Opportunities. *Proceedings of the IEEE*, 110(6):712–734, 2022.

[30] D. Rathee, S. Rangi, P. Chakarvarti, and V. Singh. Recent trends in Wireless Body Area Network (WBAN) research and cognition based adaptive WBAN architecture for healthcare. *Health Technol.*, pages 1–6, 05 2014.

[31] M. Yaghoubi, K. Ahmed, and Y. Miao. Wireless Body Area Network (WBAN): A Survey on Architecture, Technologies, Energy Consumption, and Security Challenges. *Journal of Sensor and Actuator Networks*, 11(4), 2022.

[32] L. Tong, Q. Zhao, and S. Adireddy. Sensor networks with mobile agents. In *IEEE Military Communications Conference, 2003. MILCOM 2003.*, volume 1, pages 688–693 Vol.1, 2003.

[33] K. Ma, Y. Zhang, and W. Trappe. Managing the Mobility of a Mobile Sensor Network Using Network Dynamics. *IEEE Transactions on Parallel and Distributed Systems*, 19(1):106–120, 2008.

[34] B. P. Santos, O. Goussevskaia, L. F. Vieira, M. A. Vieira, and A. A. Loureiro. Mobile Matrix: Routing under mobility in IoT, IoMT, and Social IoT. *Ad Hoc Networks*, 78:84–98, 2018.

[35] W. Wang, V. Srinivasan, and K.-C. Chua. Using Mobile Relays to Prolong the Lifetime of Wireless Sensor Networks. In *11th Annual International Conference on Mobile Computing and Networking*, MobiCom '05, page 270–283, New York, NY, USA, 2005. Association for Computing Machinery.

[36] T. Kotsilieris and G. Karetsos. Prolonging the Lifetime of Two-Tiered Wireless Sensor Networks with Mobile Relays. *ISRN Sensor Networks*, 2013(11), 2013.

[37] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling and Analysis of a Three-tier Architecture for Sparse Sensor Networks. *Ad Hoc Networks*, 1(2):215–233, 2003.

[38] B. Musznicki and P. Zwierzykowski. Performance Evaluation of Flooding Algorithms for Wireless Sensor Networks Based on EffiSen: The Custom-Made Simulator. In A.-S. K. Pathan, M. M. Monowar, and S. Khan, editors, *Simulation Technologies in Networking and Communications: Selecting the Best Tool for the Test*, pages 433–458. CRC Press, Taylor & Francis Group, USA, 2015.

[39] M. Hu, Z. Zhong, M. Ni, and A. Baiocchi. Design and Analysis of A Beacon-Less Routing Protocol for Large Volume Content Dissemination in Vehicular Ad Hoc Networks. *Sensors*, 16(11), 2016.

[40] P. Jadhav and R. Satao. A Survey on Opportunistic Routing Protocols for Wireless Sensor Networks. *Procedia Computer Science*, 79:603–609, 2016. Proceedings of International Conference on Communication, Computing and Virtualization (ICCCV) 2016.

[41] L. Pelusi, A. Passarella, and M. Conti. Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks. *Comm. Mag.*, 44(11):134–141, nov 2006.

[42] L. Torgerson, S. C. Burleigh, H. Weiss, A. J. Hooke, K. Fall, D. V. G. Cerf, K. Scott, and R. C. Durst. Delay-Tolerant Networking Architecture. RFC 4838, April 2007.

[43] J. Leguay, A. Lindgren, J. Scott, T. Friedman, and J. Crowcroft. Opportunistic Content Distribution in an Urban Setting. In *Proceedings of the 2006 SIGCOMM workshop on Challenged networks*, pages 205–212, 2006.

[44] T.-M. Pham and S. Fdida. DTN Support for News Dissemination in an Urban Area. In J. Domingo-Pascual, P. Manzoni, S. Palazzo, A. Pont, and C. Scoglio, editors, *NETWORKING 2011*, pages 120–133, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[45] S. Cabrero, R. García, X. G. Pañeda, and D. Melendi. Understanding Opportunistic Networking for Emergency Services: Analysis of One Year of GPS Traces. In *10th ACM MobiCom Workshop on Challenged Networks*, CHANTS '15, page 31–36, New York, NY, USA, 2015. Association for Computing Machinery.

[46] P. Cruz, R. S. Couto, and L. H. M. Costa. An algorithm for sink positioning in bus-assisted smart city sensing. *Future Generation Computer Systems*, 93:761–769, 2019.

[47] Y. Zguira, H. Rivano, and A. Meddeb. Internet of Bikes: A DTN Protocol with Data Aggregation for Urban Data Collection. *Sensors*, 18(9), 2018.

[48] Cisco. Cisco Visual Networking Index, Global Mobile Data Traffic Forecast Update, 2015-2020 White Paper. Document ID: 958959758. 2016.

[49] I. Park, D. Kim, and D. Har. MAC Achieving Low Latency and Energy Efficiency in Hierarchical M2M Networks With Clustered Nodes. *IEEE Sensors Journal*, 15:1657–1661, 03 2015.

[50] M. Bonola, L. Bracciale, P. Loreti, R. Amici, A. Rabuffi, and G. Bianchi. Opportunistic communication in smart city: Experimental insight with small-scale taxi fleets as data carriers. *Ad Hoc Networks*, 43:43–55, 2016.

[51] CRAWDAD – Dataset of mobility traces of taxi cabs in Rome, Italy. https://crawdad.org/coppe-ufrj/RioBuses/.

[52] D. S. Dias, L. H. M. Costa, and M. D. de Amorim. Data offloading capacity in a megalopolis using taxis and buses as data carriers. *Vehicular Communications*, 14:80–96, 2018.

[53] Data.Rio open data portal. https://www.data.rio.

[54] CRAWDAD – Dataset of mobility traces of buses in Rio de Janeiro, Brasil, 19 March 2018. https://crawdad.org/coppe-ufrj/RioBuses/.

[55] D. Chaładyniak and J. Grzybowski. Wybrane metody diagnozowania nieprawidłowości działania sieci teleinformatycznych. *Zeszyty Naukowe Warszawskiej Wyższej Szkoły Informatyki*, nr 8:61–76, 2012.

[56] Ł. Skibniewski and J. Furtak. Zdalne Laboratorium Sieciowe. *Biuletyn Instytutu Automatyki i Robotyki*, 18:1–20, 2012.

[57] M. Stasiak and M. Michalski. Algorytmy Wspomagające Projektowanie Pierścieniowych Sieci Optycznych. In *Poznańskie Warsztaty Telekomunikacyjne (PWT 2003)*, Poznań, Poland, 11–12 December 2003.

[58] P. Santi. *Topology Control in Wireless Ad Hoc and Sensor Networks*. John Wiley & Sons Ltd, Chichester, West Sussex, England, 2005.

[59] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Topology Management for Sensor Networks: Exploiting Latency and Density. In *3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc 2002)*, pages 135–145, New York, NY, USA, 2002. ACM.

[60] M. Younis, I. F. Senturk, K. Akkaya, S. Lee, and F. Senel. Topology management techniques for tolerating node failures in wireless sensor networks: A survey. *Computer Networks*, 58:254–283, 2014.

[61] G. Sosnowski. Przegląd algorytmów dynamicznego zarządzania topologią w bezprzewodowych, ruchomych sieciach ad hoc. In *Poznańskie Warsztaty Telekomunikacyjne (PWT 2005)*, Poznań, Poland, 8–9 December 2005.

[62] J. Zhao and G. Cao. Robust Topology Control in Multi-Hop Cognitive Radio Networks. *IEEE Transactions on Mobile Computing*, 13(11):2634–2647, November 2014.

[63] K. M. amd Do-Sik Yoo, W. Lee, and S.-J. Oh. Receiver Cooperation in Topology Control for Wireless Ad-Hoc Networks. *IEEE Transactions on Wireless Communications*, 14(4):1858–1870, April 2015.

[64] M. A. Labrador and P. M. Wightman. *Topology Control in Wireless Sensor Networks: With a Companion Simulation Tool for Teaching and Research*. Springer Netherlands, 2009.

[65] A. A. Aziz, , Y. A. Sekercioglu, P. Fitzpatrick, and M. Ivanovich. A Survey on Distributed Topology Control Techniques for Extending the Lifetime of Battery Powered Wireless Sensor Networks. *IEEE Communications Surveys Tutorials*, 15(1):121–144, first quarter 2013.

[66] M. Li, Z. Li, and A. V. Vasilakos. A Survey on Topology Control in Wireless Sensor Networks: Taxonomy, Comparative Study, and Open Issues. *Proceedings of the IEEE*, 101(12):2538–2557, December 2013.

[67] E. Niewiadomska-Szynkiewicz, P. Kwaśniewski, and I. Windyga. Comparative Study of Wireless Sensor Networks Energy-Efficient Topologies and Power Save Protocols. *Journal of Telecommunications and Information Technology*, (3):68–75, 2009.

[68] Y. Huang, J.-F. Martínez, V. H. Díaz, and J. Sendra. A Novel Topology Control Approach to Maintain the Node Degree in Dynamic Wireless Sensor Networks. *Sensors*, 14(3):4672–4688, 2014.

[69] I. Yoon, D. K. Noh, and H. Shin. Energy-Aware Hierarchical Topology Control for Wireless Sensor Networks with Energy-Harvesting Nodes. *International Journal of Distributed Sensor Networks*, 2015.

[70] B. Chen and L. li Wang. An Interference Prediction-Based Topology Control Algorithm for 3-D Wireless Sensor Networks. *Journal of Computational Information Systems*, 7(4):1198–1205, April 2011.

[71] S. S. Dhillon and K. Chakrabarty. *Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks*, volume 3. IEEE, 2003.

[72] J. Ai and A. A. Abouzeid. Coverage by directional sensors in randomly deployed wireless sensor networks. *Journal of Combinatorial Optimization*, 11(1):21–41, 2006.

[73] B. Musznicki, M. Tomczak, and P. Zwierzykowski. Dijkstra-based Localized Multicast Routing in Wireless Sensor Networks. In *International Symposium on Communication Systems, Networks and Digital Signal Processing 2012 (CSNDSP)*, Poznań, Poland, 18–20 July 2012.

[74] M. E. M. Campista and M. G. Rubinstein. *Advanced Routing Protocols for Wireless Networks*. John Wiley & Sons, Chichester, West Sussex, England, 2014.

[75] Q. Mamun. A Qualitative Comparison of Different Logical Topologies for Wireless Sensor Networks. *Sensors*, 12(11):14887–14913, 2012.

[76] X. Liu. A Survey on Clustering Routing Protocols in Wireless Sensor Networks. *Sensors*, 12(8):11113–11153, 2012.

[77] M. Głąbowski, B. Musznicki, P. Nowak, and P. Zwierzykowski. Shortest Path Problem Solving Based on Ant Colony Optimization Metaheuristic. *International Journal of Image Processing & Communications, Special Issue: Algorithms and Protocols in Packet Networks*, 17(1–2):7–17, December 2012.

[78] M. Stein, T. Petry, I. Schweizer, M. Brachmann, and M. Mühlhäuser. Topology Control in Wireless Sensor Networks: What Blocks the Breakthrough? In *IEEE 41st Conference on Local Computer Networks (LCN 2016)*, pages 389–397, November 2016.

[79] R. N. Murty, G. Mainland, I. Rose, A. R. Chowdhury, A. Gosain, J. Bers, and M. Welsh. CitySense: An Urban-Scale Wireless Sensor Network and Testbed. In *2008 IEEE Conference on Technologies for Homeland Security*, pages 583–588, 2008.

[80] D. R. Moogk. Minimum Viable Product and the Importance of Experimentation in Technology Startups. *Technology Innovation Management Review*, 2(3):23, 2012.

[81] P. Walkowiak, R. Szalski, B. Musznicki, D. Dudek, K. Kowalik, and P. Zwierzykowski. Evaluation of CARMNET System in INEA HOTSPOT Network. In *IEICE Information and Communication Technology Forum (ICTF 2014)*, Poznań, Poland, 28–30 May 2014.

[82] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees. Deploying a Wireless Sensor Network on an Active Volcano. *IEEE Internet Computing*, 10(2):18–25, March 2006.

[83] H. Zhang and J. C. Hou. Is Deterministic Deployment Worse than Random Deployment for Wireless Sensor Networks? In *25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, pages 1–13. IEEE, 2006.

[84] S. N. Simić and S. Sastry. Distributed Environmental Monitoring Using Random Sensor Networks. In F. Zhao and L. Guibas, editors, *Information Processing in Sensor Networks: Second International Workshop (IPSN 2003)*, pages 582–592, Palo Alto, CA, USA, 22–23 April 2003. Springer Berlin Heidelberg.

[85] K. Römer and F. Mattern. The Design Space of Wireless Sensor Networks. *Wireless Communications, IEEE*, 11(6):54–61, 2004.

[86] A. Howard, M. J. Matarić, and G. S. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots*, 13(2):113–126, 2002.

[87] H. Gao, S. Utecht, G. Patrick, G. Hsieh, F. Xu, H. Wang, and Q. Li. High Speed Data Routing in Vehicular Sensor Networks. *Journal of Communications*, 5(3):181–188, 2010.

[88] M. Wooldridge and N. R. Jennings. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10:115–152, 6 1995.

[89] U. Lee and M. Gerla. A survey of urban vehicular sensing platforms. *Computer Networks*, 54(4):527–544, 2010.

[90] *IEEE Standard for Information technology, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 29 March 2012. IEEE Std 802.11™-2012.

[91] K. Kowalik, A. Kliks, B. Musznicki, M. Kołodziejski, and P. Kryszkiewicz. Observation of WiMAX Radio Parameters to Enhance Spectrum Utilisation in Mixed Environment. *Journal of Telecommunications and Information Technology*, (1):42–50, 2018.

[92] H. Conceição, M. Ferreira, and J. Barros. On the Urban Connectivity of Vehicular Sensor Networks. In *International Conference on Distributed Computing in Sensor Systems (DCOSS 2008)*, pages 112–125. Springer-Verlag Berlin Heidelberg, 2008.

[93] M. Boban, T. T. Vinhoza, M. Ferreira, J. Barros, and O. K. Tonguz. Impact of Vehicles as Obstacles in Vehicular Ad Hoc Networks. *Selected Areas in Communications, IEEE Journal on*, 29(1):15–28, 2011.

[94] A. Cardote, S. Sargento, and P. Steenkiste. On the connection availability between relay nodes in a VANET. In *IEEE GLOBECOM Workshops (GC Wkshps 2010)*, pages 181–185. IEEE, 2010.

[95] C. Ameixieira, A. Cardote, F. Neves, R. Meireles, S. Sargento, L. Coelho, J. Afonso, B. Areias, E. Mota, R. Costa, et al. HarborNet: A Real-World Testbed for Vehicular Networks. *Communications Magazine, IEEE*, 52(9):108–114, 2014.

[96] Creating The World's Largest Network of Connected Vehicles for Smart Cities. https://veniam.com/wp-content/uploads/2016/02/PortoCaseStudy1.pdf.

[97] *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements*, 15 July 2010. IEEE Std 802.11p™-2010.

[98] P. Santos, T. Calçada, A. Sá, D. Guimaraes, T. Condeixa, C. Penichet, S. Sargento, A. Aguiar, and J. Barros. Demo Abstract: Experiments On Using Vehicles As Data Mules For Data Collection From Urban Sensors. In *12th European Conference on Wireless Sensor Networks (EWSN 2015)*, pages 17–18, Porto, Portugal, 9–11 February 2015.

[99] CityMobil2 Experience and Recommendations. http://www.citymobil2.eu/en/upload/Deliverables/PU/CityMobil2bookletwebfinal_17112016.pdf.

[100] A. Alessandrini, A. Cattivera, C. Holguin, and D. Stam. *Road Vehicle Automation*, chapter CityMobil2: Challenges and Opportunities of Fully Automated Mobility, pages 169–184. Springer International Publishing, Switzerland, 2014.

[101] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis. Design and Deployment of Industrial Sensor Networks: Experiences from a Semiconductor Plant and the North Sea. In *3rd International Conference on Embedded Networked Sensor Systems (SenSys 2005)*, pages 64–75, New York, NY, USA, 2005. ACM.

[102] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.

[103] G. Carles and J. Paradells. Wireless Home Automation Networks: A Survey of Architectures and Technologies. *IEEE Communications Magazine*, 48(6):92–101, June 2010.

[104] T. Zachariah, N. Klugman, B. Campbell, J. Adkins, N. Jackson, and P. Dutta. The Internet of Things Has a Gateway Problem. In *16th International Workshop on Mobile Computing Systems and Applications (HotMobile 2015)*, pages 27–32, New York, NY, USA, 2015. ACM.

[105] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke. Smart Grid Technologies: Communication Technologies and Standards. *IEEE transactions on Industrial informatics*, 7(4):529–539, 2011.

[106] L. Quan-Xi and L. Gang. Design of remote automatic meter reading system based on ZigBee and GPRS. In *3rd International Symposium on Computer Science and Computational Technology (ISCSCT 2010)*, volume 2, pages 186–189, 2010.

[107] M. Piechowiak, P. Zwierzykowski, and B. Musznicki. LoRaWAN Metering Infrastructure Planning in Smart Cities. *Applied Sciences*, 13(14), 2023.

[108] COST Action 231 Final Report. http://www.lx.it.pt/cost231/final_report.htm.

[109] scikit-learn Clustering documentation. https://scikit-learn.org/stable/modules/clustering.html#k-means.

[110] Cisco openBerlin Innovation Center. https://www.cisco.com/c/m/de_de/innovationcenter/berlin.html.

[111] *ZigBee Specification*, 7 September 2012. ZigBee Document 053474r20.

[112] *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS – Access networks – In premises networks – Short range narrow-band digital radiocommunication transceivers – PHY, MAC, SAR and LLC layer specifications*, 2015. Recommendation ITU-T G.9959.

[113] Mobility Database. https://database.mobilitydata.org.

[114] GZM – bus GPS locations. https://otwartedane.metropoliagzm.pl/dataset/lokalizacje-autobusow-ztm.

[115] R. T. Fielding and R. N. Taylor. Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, 2002.

[116] The Things Stack – Packet Broker Mapper. https://www.thethingsindustries.com/docs/getting-started/packet-broker/api/.

[117] Airly Developer – Documentation. https://developer.airly.org/en/docs.

[118] Syngeos API. https://syngeos.pl/api/.

[119] GIOŚ Air Quality portal – Measurement data archives. https://powietrze.gios.gov.pl/pjp/archives.

[120] GIOŚ Air Quality portal – Application Programming Interface. https://powietrze.gios.gov.pl/pjp/content/api?lang=en.

[121] Smart City Poznań. https://www.poznan.pl/mim/smartcity/api-dane-przestrzenne,p,25877,38305.html.

[122] Warsaw open data. https://api.um.warszawa.pl.

[123] CKAN – The Open Source Data Portal Software. https://ckan.org.

[124] Open Gdańsk. https://ckan.multimediagdansk.pl.

[125] Wrocław – Open data. https://www.wroclaw.pl/open-data/.

[126] CKAN DataStore extension. https://docs.ckan.org/en/2.9/maintaining/datastore.html.

[127] On the threshold of a breakthrough. Shared mobility in Poland. https://smartride.pl/wp-content/uploads/2020/02/Raport_Shared_Mobility_2019_PL_maly.pdf.

[128] blinkee.city. https://blinkee.city.

[129] Bolt – Scooter rental. https://bolt.eu/en/scooters/.

[130] Poznań City Bike – How it works? https://poznanskirower.pl/en/polski-jak-to-dziala/.

[131] Traficar – How it works? https://www.traficar.pl/how.

[132] take&drive. https://takeanddrive.eu.

[133] Get public information. https://www.gov.pl/web/gov/uzyskaj-informacje-publiczna.

[134] D. Crockford. The application/json Media Type for JavaScript Object Notation (JSON). RFC 4627, IETF, 7 2006.

[135] Y. Shafranovich. Common Format and MIME Type for Comma-Separated Values (CSV) Files. RFC 4180, October 2005.

[136] Protocol Buffers. https://developers.google.com/protocol-buffers/.

[137] Open Gdańsk – GPS positions of the vehicles. https://ckan.multimediagdansk.pl/dataset/tristar/resource/0683c92f-7241-4698-bbcc-e348ee355076.

[138] ZTM Poznań – For developers – GTFS-RT. https://www.ztm.poznan.pl/pl/dla-deweloperow/gtfsRtFiles.

[139] H. Butler, M. Daly, A. Doyle, S. Gillies, T. Schaub, and S. Hagen. The GeoJSON Format. RFC 7946, August 2016.

[140] GTFS Realtime Overview. https://developers.google.com/transit/gtfs-realtime.

[141] Open Gdańsk – GTFS-RT resources. https://ckan.multimediagdansk.pl/dataset/tristar/resource/976e1fd1-73d9-4237-b6ba-3c06004d1105.

[142] Data elements and interchange formats – Information interchange – Representation of dates and times. Technical report, International Organization for Standardization, 2004.

[143] Linux manual page – time(2). https://man7.org/linux/man-pages/man2/time.2.html.

[144] Wrocław open data – Wrocław City Bike stations. https://www.wroclaw.pl/open-data/dataset/nextbikesoap_data/resource/42eea6ec-43c3-4d13-aa77-a93394d6165a.

[145] Wrocław City Bike. https://wroclawskirower.pl/en/.

[146] Department of Defense World Geodetic System 1984: Its Definition and Relationships with Local Geodetic Systems. Second Edition. Technical report, Defense Mapping Agency, 1 September 1991.

[147] Open Gdańsk – public vahicles list. https://ckan.multimediagdansk.pl/dataset/tristar/resource/fff34d32-885d-4622-a9a2-c2d18ccf68c1.

[148] Open Gdańsk – timetables. https://ckan.multimediagdansk.pl/dataset/tristar/resource/a023ceb0-8085-45f6-8261-02e6fcba7971.

[149] ZTM Poznań – For developers – GTFS timetables. https://www.ztm.poznan.pl/pl/dla-deweloperow/gtfsFiles.

[150] Open Gdańsk – positions of ticket machines. https://ckan.multimediagdansk.pl/dataset/tristar/resource/af7bf4a9-e62e-4af2-906a-fa27c2532dfd.

[151] Poznań – positions of parking meters. https://www.poznan.pl/mim/plan/map_service.html?mtype=pub_transport&co=parking_meters.

[152] Warsaw open data – public vehicle positions – API documentation. https://api.um.warszawa.pl/files/9fae6f84-4c81-476e-8450-6755c8451ccf.pdf.

[153] BusLive. https://buslive.pl.

[154] Open Gdańsk – list of bus stops. https://ckan.multimediagdansk.pl/dataset/tristar/resource/4c4025f0-01bf-41f7-a39f-d156d201b82b.

[155] NetworkX – Network Analysis in Python. https://networkx.org.

[156] OpenStreetMap. https://www.openstreetmap.org/copyright.

[157] F. Harary and G. Gupta. Dynamic graph models. *Mathematical and Computer Modelling*, 25(7):79–87, 1997.

[158] A. Ferreira. On models and algorithms for dynamic communication networks: The case for evolving graphs. In *Proceedings of ALGOTEL 2002*. sn, 2002.

[159] A. Li, S. P. Cornelius, Y.-Y. Liu, L. Wang, and A.-L. Barabási. The fundamental advantages of temporal networks. *Science*, 358(6366):1042–1046, 2017.

[160] S. Merugu, M. H. Ammar, and E. W. Zegura. Routing in space and time in networks with predictable mobility. Technical report, Georgia Institute of Technology, 2004.

[161] M. Huang, S. Chen, Y. Zhu, B. Xu, and Y. Wang. Topology Control for Time-Evolving and Predictable Delay-Tolerant Networks. In *2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems*, pages 82–91. IEEE, 2011.

[162] B. George and S. Shekhar. Time-aggregated graphs for modeling spatio-temporal networks. In *Journal on Data Semantics XI*, pages 191–212. Springer, 2008.

[163] D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 504–513, 2000.

[164] P. Holme and J. Saramäki. Temporal Networks. *Physics reports*, 519(3):97–125, 2012.

[165] H. Wu, J. Cheng, S. Huang, Y. Ke, Y. Lu, and Y. Xu. Path Problems in Temporal Graphs. *Proceedings of the VLDB Endowment*, 7(9):721–732, 2014.

[166] P. Flocchini, B. Mans, and N. Santoro. Exploration of Periodically Varying Graphs. In *International Symposium on Algorithms and Computation*, pages 534–543. Springer, 2009.

[167] N. Masuda and R. Lambiotte. *A guide to temporal networks.* World Scientific, 2016.

[168] P. Holme. Modern temporal network theory: a colloquium. *The European Physical Journal B*, 88(9):1–30, 2015.

[169] P. Holme and J. Saramäki. *Temporal network theory*, volume 2. Springer, 2019.

[170] Y. Wang, Y. Yuan, Y. Ma, and G. Wang. Time-dependent graphs: Definitions, applications, and algorithms. *Data Science and Engineering*, 4(4):352–366, 2019.

[171] C. C. Robusto. The cosine-haversine formula. *The American Mathematical Monthly*, 64(1):38–40, 1957.

[172] S. Huang, A. W.-C. Fu, and R. Liu. Minimum Spanning Trees in Temporal Graphs. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 419—430, New York, NY, USA, 2015. Association for Computing Machinery.

[173] M. Piechowiak and P. Zwierzykowski. Simulations of the MAC Layer in the LoRaWAN Networks. *Journal of Telecommunications and Information Technology*, (2):22–27, 2020.

[174] M. Piechowiak and P. Zwierzykowski. Efficiency Analysis of Multicast Routing Algorithms in Large Networks. In *International Conference on Networking and Services (ICNS '07)*, pages 101–101, 2007.

[175] M. Piechowiak and P. Kotlarz. Network topology models for telecommunication and automation networks. *Image Processing & Communications*, Vol. 15, no 2:47–53, 2010.

[176] Poznan.pl – Znamy liczbę mieszkańców Poznania. https://www.poznan.pl/mim/info/news/znamy-liczbe-mieszkancow-poznania,188075.html.

[177] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[178] M. Piechowiak, M. Stasiak, and P. Zwierzykowski. Analysis of the Influence of Group Members Arrangement on the Multicast Tree Cost. In *2009 Fifth Advanced International Conference on Telecommunications*, pages 429–434, 2009.

[179] M. Piechowiak and P. Zwierzykowski. Performance of Fast Multicast Algorithms in Real Networks. In *EUROCON 2007 - The International Conference on "Computer as a Tool"*, pages 956–961, 2007.

[180] M. Głąbowski, B. Musznicki, P. Nowak, and P. Zwierzykowski. Review and Performance Analysis of Shortest Path Problem Solving Algorithms. *International Journal On Advances in Software*, 7(1 & 2):20–30, 2014.

[181] B. Musznicki, M. Tomczak, and P. Zwierzykowski. Geographic Dijkstra-based Multicast Algorithm for Wireless Sensor Networks. *International Journal of Image Processing & Communications, Special Issue: Algorithms and Protocols in Packet Networks*, 17(1–2):33–46, December 2012.

[182] M. Głąbowski, B. Musznicki, P. Nowak, and P. Zwierzykowski. An Algorithm for Finding Shortest Path Tree Using Ant Colony Optimization Metaheuristic. In R. S. Choraś, editor, *Image Processing and Communications Challenges 5*, volume 233 of *Advances in Intelligent Systems and Computing*, pages 317–326. Springer International Publishing, Switzerland, 2014.

[183] E. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959.

[184] K.-S. Wong and T.-C. Wan. Current State of Multicast Routing Protocols for Disruption Tolerant Networks: Survey and Open Issues. *Electronics*, 8(2), 2019.

[185] S. Burleigh, K. Fall, and E. J. Birrane. Bundle Protocol Version 7. RFC 9171, January 2022.

[186] A. Vahdat and D. Becker. Epidemic Routing for Partially Connected Ad Hoc Networks. Technical report cs-2000-06, Duke University, 2000.

[187] D. Braginsky and D. Estrin. Rumor Routing Algorthim for Sensor Networks. WSNA '02, pages 22—-31, New York, NY, USA, 2002. Association for Computing Machinery.

[188] J. A. Sanchez, P. M. Ruiz, J. Liu, and I. Stojmenovic. Bandwidth-Efficient Geographic Multicast Routing Protocol for Wireless Sensor Networks. *IEEE Sensors Journal*, 7(5):627–636, 2007.

[189] S. Basagni, A. Carosi, and C. Petrioli. Controlled vs. Uncontrolled Mobility in Wireless Sensor Networks: Some performance insights. In *2007 IEEE 66th Vehicular Technology Conference*, pages 269–273. IEEE, 2007.

[190] P. J. Leach, R. Salz, and M. H. Mealling. A Universally Unique IDentifier (UUID) URN Namespace. RFC 4122, July 2005.

[191] W. Zhao, M. Ammar, and E. Zegura. Multicasting in Delay Tolerant Networks: Semantic Models and Routing Algorithms. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-Tolerant Networking*, WDTN '05, pages 268—-275, New York, NY, USA, 2005. Association for Computing Machinery.

[192] V. Gunturi, S. Shekhar, and A. Bhattacharya. Minimum Spanning Tree on Spatio-Temporal Networks. In P. G. Bringas, A. Hameurlain, and G. Quirchmayr, editors, *Database and Expert Systems Applications*, pages 149–158, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[193] E. Grigorescu, Y.-S. Lin, and K. Quanrud. Online Directed Spanners and Steiner Forests. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2021.

[194] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56–61, 2010.

[195] M. L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

[196] Poznań – positions of public transport stops. http://www.poznan.pl/mim/plan/map_service.html?mtype=pub_transport&co=cluster.

[197] Poznań – positions of ticket machines. http://www.poznan.pl/mim/plan/map_service.html?mtype=pub_transport&co=class_objects&class_id=4000.

[198] Wrocław open data – positions of public transporation vehicles. https://www.wroclaw.pl/open-data/dataset/lokalizacjapojazdowkomunikacjimiejskiejnatrasie_data.

[199] Wrocław open data – Vozilla – city electric car rental – parking lots. https://www.wroclaw.pl/open-data/dataset/wykaz-miejsc-parkingowych-miejskiej-wypozyczalni-samochodow-elektrycznych-vozillaa.

[200] Gdańsk w liczbach – liczba mieszkańców Gdańska. https://www.gdansk.pl/gdansk-w-liczbach/mieszkancy,a,108046.

[201] Geoportal Krajowy Na Mapie. https://geoportal-krajowy.pl.

[202] Gdańsk Municipal Transport Authority – Timetables. https://ztm.gda.pl/rozklady.

[203] Poznań Municipal Transport Company – Timetable. https://www.mpk.poznan.pl/en/timetable/.

[204] Statystyka Warszawy – Miasto Warszawa. https://um.warszawa.pl/statystyka-warszawy-2022.

[205] Warsaw Public Transport – Timetables. https://www.wtp.waw.pl/en/timetables/.

[206] Statistical Office in Wroclaw – Population. https://wroclaw.stat.gov.pl/en/zakladka2/.

[207] Wrocław Municipal Transport Company – Timetable. https://www.wroclaw.pl/komunikacja/rozklady-jazdy.

[208] H. Karvonen, C. Pomalaza-Ráez, K. Mikhaylov, M. Hämäläinen, and J. Iinatti. Experimental Performance Evaluation of BLE 4 Versus BLE 5 in Indoors and Outdoors Scenarios. In *Advances in Body Area Networks I*, pages 235–251. Springer International Publishing, 2019.

[209] A. E. Ferreira, F. M. Ortiz, L. H. M. Costa, B. Foubert, I. Amadou, and N. Mitton. A study of the LoRa signal propagation in forest, urban, and suburban environments. *Annals of Telecommunications*, 75:333–351, 2020.

[210] A. Varga. *A Practical Introduction to the OMNeT++ Simulation Framework*, pages 3–51. Springer International Publishing, 2019.

[211] G. F. Riley and T. R. Henderson. *The ns-3 Network Simulator*, pages 15–34. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[212] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic Traffic Simulation using SUMO. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[213] C. Sommer, R. German, and F. Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, January 2011.

[214] A. Keränen, J. Ott, and T. Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA, 2009. ICST.