electronics

MDPI

Article

Urban Delay-Tolerant Multicast Using Uncontrolled Mobile Relay

Bartosz Musznicki and Piotr Zwierzykowski *

Institute of Computer and Communication Networks, Faculty of Computing and Telecommunications, Poznan University of Technology, 60-965 Poznan, Poland
* Correspondence: piotr.zwierzykowski@put.poznan.pl

Abstract: The development of network functionalities in the urban environment is accompanied by the emergence of new publicly available data sources. They are the basis of the introduced research architecture and environment which are used to investigate the new multicast algorithms proposed in this paper. These message-oriented algorithms are primarily intended to meet the needs of opportunistic routing in heterogeneous urban sensor networks. Although, due to their generalized and protocol-agnostic design, they can be of use in other network applications and research areas. Uncontrolled mobile relay devices are the key elements of the presented delay-tolerant multicast framework. Multicast structures are modeled in four Polish cities based on open data on the location of public transportation vehicles and elements of urban infrastructure. Over 16,000 graphs were built and analyzed. It has been shown that the use of uncontrolled mobile relay enables the construction of time-spanning time-changing multicast structures. Their features are determined by the topology of a given city area, the distribution of destination nodes, as well as the number and the routes of mobile relay nodes. The efficacy and efficiency of the algorithms depend on the radio range of the nodes, maximum time span of forwarded messages, and network structure knowledge availability.

Keywords: urban sensor networks; delay-tolerant multicast; opportunistic routing; graph modeling

Citation: Musznicki, B.; Zwierzykowski, P. Urban Delay-Tolerant Multicast Using Uncontrolled Mobile Relay. *Electronics* **2024**, *13*, 510. https://doi.org/10.3390/electronics13030510

Academic Editor: Antonio J. Jara

Received: 6 December 2023
Revised: 14 January 2024
Accepted: 19 January 2024
Published: 25 January 2024

1. Introduction

A typical *wireless sensor network* (WSN) is considered to be a homogeneous multi-node multi-hop network composed of devices with sensing and communication capabilities [1]. The *unicast* is usually the most frequent form of transmission in a network of this type. A single *source* node initiates such a transmission addressed to a single *destination* node. The intermediate *relay* nodes send it further to make it reach the *sink*, which is the gateway of the sensor network leading to the external destination. A quite different relationship occurs in the case of *multicast*, where a single node initiates the transmission to reach multiple destination nodes, i.e., a multicast group. 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.

In actual urban environments, various types of networked devices with sensing capabilities are present. They are seen as the elements of a diversified *internet of things* (IoT) ecosystem. They can either be stationary (fixed), e.g., the elements of street infrastructure, and measurement equipment, or be mobile, e.g., the public transportation and service vehicles, electric rental cars, kick scooters, and smartphones. Some are constructed specifically for sensing purposes and others perform various types of measurements in addition to their main functionalities. The boundaries between the types of usually investigated networks become blurred and their actual structures become heterogeneous. Their deployment (location) can be deterministic, predictable, or random, as discussed in [2]. Examples of connected devices in large Polish cities are presented in Figure 1.

(**a**) (**b**) (**c**)

Figure 1. Examples of connected devices in cities in Poland. (**a**) Rental scooter in Gdańsk in November 2019. (**b**) Tram and electric kick scooters in Poznań in September 2019. (**c**) Electric rental car in Wrocław in December 2019.

The mobility patterns of urban nodes are usually beyond the influence of the operator of such a sensor network. Their network functions and roles are only an addition to their main purpose, e.g., personal or public transportation; although, they can be used as uncontrolled mobile relays in a *delay-tolerant network* (DTN) to enable multicast routing between otherwise disjoint stationary parts of the network, as presented in Figure 2. The relays are wireless network nodes with advanced processing and storing (caching) functionalities. Stationary relays perform *store-and-forward* operations, i.e., forward the messages instantly or store them to transmit them later, at a suitable moment. Mobile relays perform *store-carry-and-forward* functions, which means that they additionally carry stored messages in space due to their movement. Nodes of this type are frequently called data *mobile ubiquitous LAN extensions* (MULEs) [3]. In this way, data bundles such as control messages or firmware updates can be disseminated to selected sensor nodes without the need to use a fixed permanently connected distribution network. This opportunistic multicast routing can be implemented in delay-tolerant applications which do not require instant network communication and when full network coverage is physically not feasible or economically unjustified [4]. This type of communication is of particular significance in post-disaster scenarios, when stationary fixed well-connected wireless networks are unavailable and emergency alerts need to be disseminated to designated devices or interested users [5–7]. They can also be used, e.g., to update the settings of large groups of nodes in remote locations with identical configurations [8].

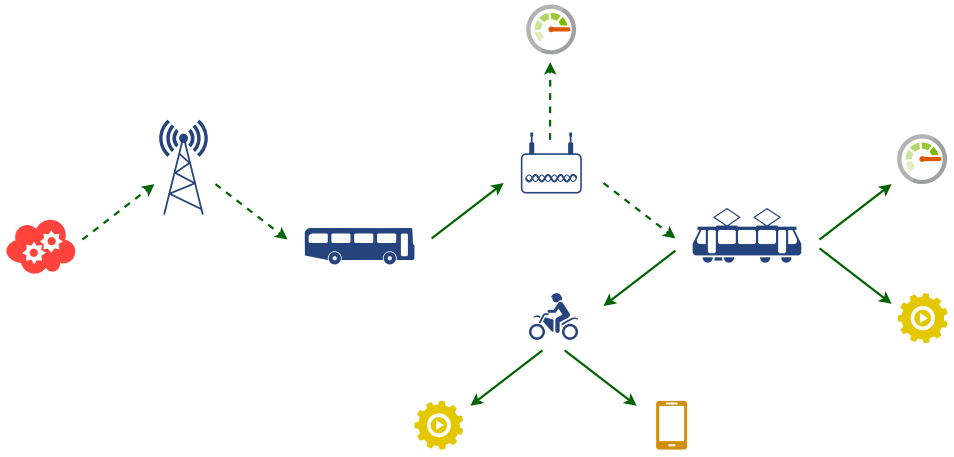


Figure 2. Delay-tolerant multicast in an urban sensor network.

Recently, an increasing number of publicly available data sources have emerged, which enable online access to the geographical location of stationary and mobile urban nodes of interest. As a result, a new open-data-based heterogeneous sensor network modeling approach was introduced by Musznicki et al. [9]. This novel methodology was then concretized and thoroughly investigated [10]. It enables the study of dynamic real-life heterogeneous network topologies constructed based on accurate node location data, instead of using synthetic models or test environments. Networks are modeled as time-spanning graphs that represent how the network changes over time. They can be then, in turn, used in the process of modeling multicast structures using multicast routing algorithms. Together with careful stationary infrastructure planning [11], this approach can lead to the efficient utilization of various types of network resources shared in an urban environment [12].

To the best knowledge of the authors, the new delay-tolerant multicast algorithms presented in this paper are the first ones designed for heterogeneous opportunistic urban sensor networks using uncontrolled mobile relays. Moreover, the presented study is the first attempt to model and analyze dynamic multicast structures in these networks as graphs based on real node location data. Although, due to their universal nature, the introduced methods could also be used in other types of wireless networks and research areas.

First, design considerations are discussed in Section 2. The family of new generalized opportunistic multicast message-oriented algorithms is defined in Section 3. The model of the related *delay-tolerant multicast router* (DTMR) is discussed in Section 4. Then, the simulation and analysis methodology is presented in Section 5. There, the complete multicast routing modeling architecture and flow are defined. Example multicast structures are modeled as graphs in four large Polish cities to show the feasibility of the method. Next, the results of the multidimensional simulation study of more than 16,000 modeled multicast graphs are presented in Section 6. The study is summarized in Section 7.

2. Delay-Tolerant Multicast Design Considerations

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

2.1. Protocol Agnosticism

Protocol and technology agnosticism go hand in hand with an algorithmic approach and graph-based network structure abstraction. Therefore, the presented multicasting methods do not follow, extend, or mimic any particular existing routing protocols. The data structures and procedures are as simple and as generic as possible. The key elements are presented in an algorithmic way inspired by popular set theory notation and the *JavaScript Object Notation* (JSON) structure [13]. 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 [8]. Similarly, a *destination* node or nodes can be called an endpoint [14].

Some well-known mechanisms can be noticed in introduced multicast algorithms, e.g., elements of epidemic routing [15], the usage of message *time-to-live* (TTL) [16], and use of periodic beacons (beacon messages) in greedy geographic forwarding [17]. The other presented mechanisms have not been used before. On the one hand, they are dictated by the heterogeneous nature of the modeled urban sensor networks, i.e., various classes and different roles of nodes. On the other hand, the presented multicast framework combines related procedures tailored to perform different forms of multicasting.

2.2. Uncontrolled Mobile Relays

At the heart of the designed algorithms is the use of uncontrolled relay nodes, i.e., routers that move independently from the influence of the routing algorithms they are running [18]. 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 the desired coverage and connectivity multiple uncontrolled relay nodes should be used, especially in vast urban areas with a sparse structure and a limited number of stationary relay nodes.

2.3. Heterogeneous Networks

An 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 network, 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 a modular and scalable design of the routing procedures to enable them to be applied in the scope fitting the needs and capabilities of the particular node class.

2.4. Network Utilization and Delivery

A fundamental feature of opportunistic heterogeneous urban networks is the high dynamics of the network topology. Frequent changes take place not only in the structure of connections but also in the number and classes of nodes available at a given moment in the area of interest. Therefore, some nodes, i.e., stationary or the ones with 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 the implementation of 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 carried and forwarded opportunistically by multiple relays, is a viable method to overcome relay node failures and network discontinuities to ensure the required multicast delivery rate. When designing and implementing algorithms of this family, though, a loop and network saturation avoidance mechanism needs to be implemented. The ones that are introduced to the presented algorithms are the use of time-to-live counters, 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 a given message.

2.5. 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 the cases of the presented algorithms. The reasoning behind this is that many current sensing nodes are, in fact, integrated parts of more advanced devices with high processing and storage capabilities, such as smartphones, onboard computers, and fixed power-grid- or solar-panel-connected measurement stations. In the 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. Similarly,

it is assumed that each node can have its own unique radio connectivity capabilities and coverage, also changing in time, depending on its configuration and location, as well as the propagation conditions and interference. These aspects are out of the scope of the discussed high-level multicast message dissemination design.

2.6. Node Location

The geographic location of the node is expressed as a pair of latitude and longitude coordinates. And yet, any other coordinate and location system could be used if it better met the needs of a particular application. Therefore, location capabilities need to be available to mobile nodes. These 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 not be equipped with positioning hardware themselves but programmed with their location measured when the devices are deployed (installed).

2.7. Next Hops and Destinations

The term *next hop* means the node, i.e., device or routing process, which is planned to receive the message next. Such nodes can be the destinations of a 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. The destination nodes can be stationary and mobile, depending on the application and node classes present in the network.

2.8. Generic and External Elements

To keep the 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 the 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 the current state of the router and all the 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 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 the current node are passed to an external higher-level *processPayload()* procedure.

2.9. Time Complexity

The presentation of each multicast algorithm includes a definition of time complexity. The use of data structures implemented using hash tables is assumed, and hence, average-case complexity $\mathcal{O}(1)$ is relevant for all basic data creation, read, update, and deletion operations. They include, e.g., accessing an element of a simple set, or including an element to a more advanced dictionary-like keyed structure. Therefore, the impact of this type of operation is omitted and the presented time complexities focus on the very essence of the algorithms.

3. Delay-Tolerant Multicast Framework

In this section, three new heterogeneous opportunistic DTN multicast algorithms for dynamic urban environments are presented. While sharing the 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 even be seen as nine different algorithms. For simplicity and to ease the implementation, they are 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 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 the 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 1–3 is an iteration over all *storedMessages* aimed at transmission of those, following the set conditions and steps, updating the 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 the target geographic region. Nodes do not join a multicast group or subscribe to a 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 the broadcast nature of the 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 the current node.

A message-forwarding flowchart related to a single iteration is depicted in Figure 3. The purpose of this swimlane diagram is to be a high-level overview of the steps, common elements, and relationships between the 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 the no-knowledge opportunistic multicast, is also the common part of the local- and global-knowledge-based algorithms. Each algorithm begins at the 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 of the algorithm. A rhombus is related to a decision step while a rhomboid represents data-related operations. A plain rectangle indicates a set of operations (a procedure). A rectangle with column and row heading lines represents a foreach loop for traversing a given set of data. The entry point of such a loop is indicated by an empty arrowhead and the exit point is marked with an empty square. The elements are grouped into overarching procedures with light- and dark-gray backgrounds.

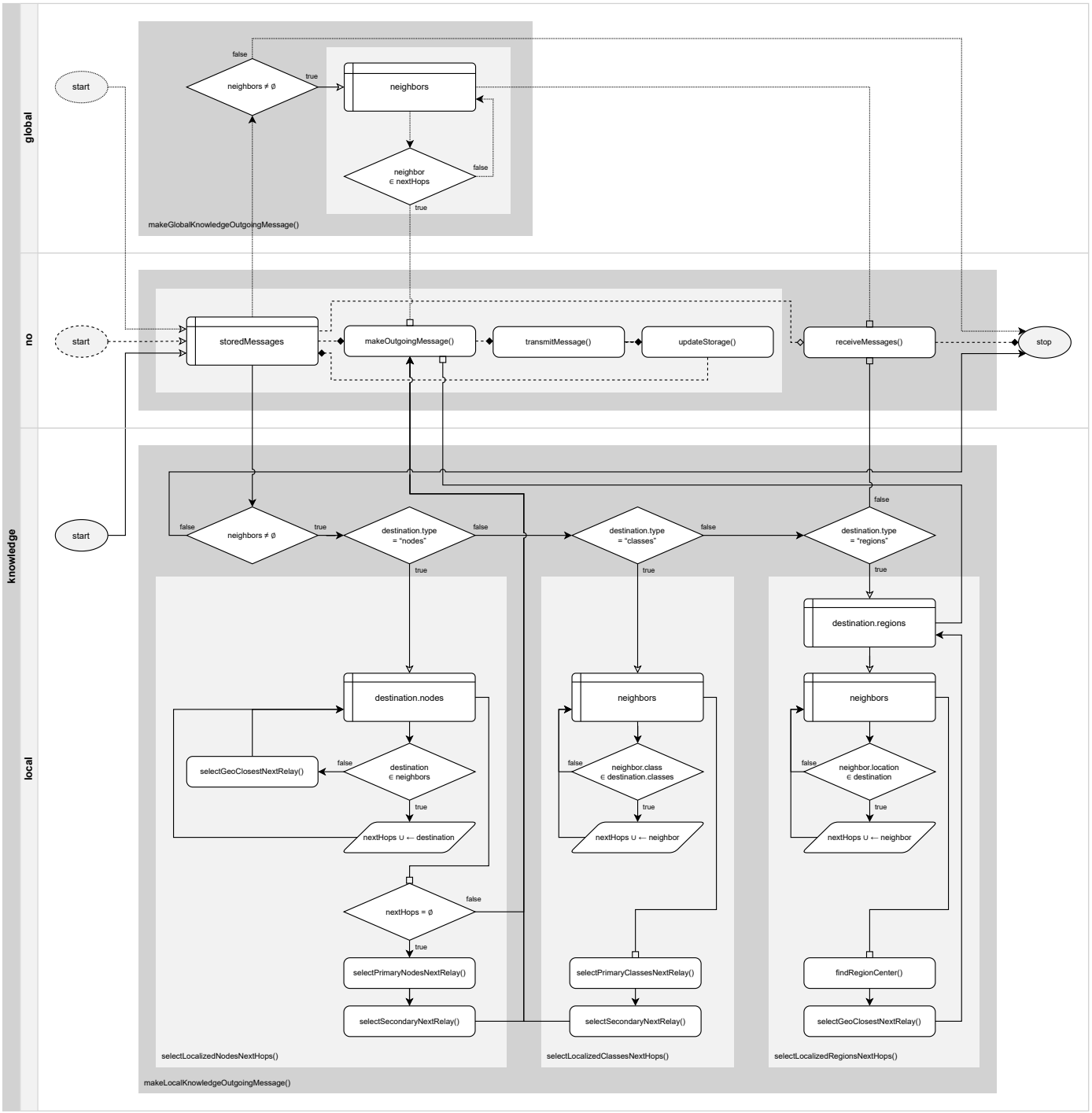


Figure 3. Delay-tolerant multicast forwarding flowchart.

3.1. No-Knowledge Opportunistic Multicast

Algorithm 1, *no-knowledge opportunistic multicast* (NKOM), is the most simplistic one in the framework. Procedure *makeNoKnowledgeOutgoingMessage*, at its core makes an *outgoingMessage* based on *storedMessage*. Due to no knowledge about its surroundings being available to the node, each stored message is transmitted in every algorithm iteration (time slot). Then, the message storage is updated and the *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 is implemented [19]. Finally, internal (originated by the 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 the well-known *universally unique identifier* (UUID)-generation methods can be used [20]. Moreover, each *message* includes the list of nodes *visited* by this message and the 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 the beaconless and no-network-knowledge nature, it is also potentially has the greatest power consumption and radio medium utilization. Albeit, thanks to its simplicity, it can be used as a 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.

Algorithm 1: No-knowledge opportunistic multicast

```
1 foreach storedMessage ∈ storedMessages do
2     outgoingMessage ← makeNoKnowledgeOutgoingMessage(storedMessage)
3     if outgoingMessage ≠ ∅ 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 ← makeOutgoingMessage(storedMessage)
2 output outgoingMessage
```

Procedure updateStorage(storedMessage)

Input:
storedMessage // message stored in current node

```
1 storedMessage.ttl −← 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 \← storedMessage
```

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*

3.2. Local Knowledge Opportunistic Multicast

Algorithm 2, *local knowledge opportunistic multicast* (LKOM), is a significant extension of the NKOM algorithm. The changes are due to the usage of locally available real-time knowledge about the current network neighborhood of the node. Therefore, Procedure *makeLocalKnowledgeOutgoingMessage* is able to take advantage of more sophisticated localized message-forwarding techniques tailored to the destination types. Together with the network structure dynamics caused by the movement of mobile nodes, they enable the multicast message to avoid becoming stuck in a local optimum with no progress toward the desired destinations. The *outgoingMessage* creation process will be started only if there are currently neighbors present and *nextHops* were selected by one of the greedy procedures discussed in the next sections.

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 the 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 2: 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 ← ∅                                   // set outgoing message to an empty set
2  neighbors ← router.neighbors                          // get the set of current neighbors
3  if neighbors ≠ ∅ then
4      nextHops ← ∅                                      // 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 ← selectLocalizedNodesNextHops(storedMessage)
           // destination type set to node classes
8          case "classes" do
9              nextHops ← selectLocalizedClassesNextHops(storedMessage)
           // destination type set to regions
10         case "regions" do
11             nextHops ← selectLocalizedRegionsNextHops(storedMessage)
12     if nextHops ≠ ∅ then
           // prepare outgoing message
13         outgoingMessage ← makeOutgoingMessage(storedMessage, nextHops)
14 output outgoingMessage
```

3.2.1. Localized Next Hops to Destination Nodes

Procedure *selectLocalizedNodesNextHops* determines which of the current *neighbors* should receive the *storedMessage* and designates them as a set of *nextHops*. For each *destination* node of the *storedMessage* a lookup for a *nextHop* is performed. If *destination* is one of the *neighbors*, then it is 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 the *destinations*, it means that no *destination* is a direct neighbor and the *location* of the *destination* nodes was unknown. In such a case, a 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* was selected, a *secondaryNextHop*, following Procedure *selectSecondaryNodesNextRelay*, will be determined as the current node neighbor which is the geographically furthest one from *primaryNextHop*. In this way, the chances of delivery of the *storedMessage* is 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 loop and unnecessary transmission medium usage and computation-related power consumption at the receiving side.

Procedure selectLocalizedNodesNextHops(storedMessage)

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

```
nextHops ← ∅                                                    // initialize next hops as an empty set
neighbors ← router.neighbors                                    // get the set of current neighbors
destinations ← storedMessage.destination.nodes                  // get the set of destination nodes
foreach destination ∈ destinations do
    nextHop ← ∅                                                 // initialize next hop as an empty set
    // set destination as next hop if it is a neighbor
    if destination ∈ neighbors then
        nextHop ← destination
    else
        // determine closest next hop if destination node location is known
        if destination.location ≠ ∅ then
            nextHop ← selectGeoClosestNextRelay(destination.location)
    if nextHop ≠ ∅ and nextHop ∉ (storedMessage.visited ∪ nextHops) then
        nextHops ∪← nextHop
// select primary and secondary next hop if no next hops were found
if nextHops = ∅ then
    // select primary next hop (relay)
    primaryNextHop ← selectPrimaryNodesNextRelay()
    if primaryNextHop ≠ ∅ and primaryNextHop ∉ storedMessage.visited then
        nextHops ∪← primaryNextHop
    // select secondary next hop (relay) if primary was selected
    if primaryNextHop ≠ ∅ then
        secondaryNextHop ← selectSecondaryNextRelay(primaryNextHop)
        if secondaryNextHop ≠ ∅ and secondaryNextHop ∉ storedMessage.visited then
            nextHops ∪← secondaryNextHop
output nextHops
```

Procedure selectGeoClosestNextRelay(location)

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

```
closestNextRelay ← ∅
minDistance ← ∞
neighbors ← router.neighbors                                    // get the set of current neighbors
foreach neighbor ∈ neighbors do
    if neighbor.role = "relay" then
        distance ← geoDistance(location, neighbor.location)
        // set current neighbor as secondary next relay
        if distance < minDistance then
            closestNextRelay ← neighbor
            minDistance ← distance
output closestNextRelay
```

Procedure selectPrimaryNodesNextRelay

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

```
primaryNextRelay ← ∅
maxNeighborsNumber ← −∞
neighbors ← router.neighbors                          // get the set of current neighbors
foreach neighbor ∈ neighbors do
    if neighbor.role = "relay" then
        // set current neighbor as primary next relay
        if neighbor.nodes.number > maxNeighborsNumber then
            primaryNextRelay ← neighbor
            maxNeighborsNumber ← neighbor.nodes.number
output primaryNextRelay
```

Procedure selectSecondaryNextRelay(primaryNextRelay)

Input:
primaryNextRelay // primary next relay

Output:
secondaryNextRelay // the furthest relay node from primary next relay

```
secondaryNextRelay ← ∅
maxDistance ← −∞
neighbors ← router.neighbors                          // get the set of current neighbors
// select secondary relay node which is the furthest one from from primary next relay
foreach neighbor ∈ (neighbors \ primaryNextRelay) do
    if neighbor.role = "relay" then
        distance ← geoDistance(primaryNextRelay, neighbor)
        // set current neighbor as secondary next relay
        if distance > maxDistance then
            secondaryNextRelay ← neighbor
            maxDistance ← distance
output secondaryNextRelay
```

3.2.2. Localized Next Hops to Destination Classes

Procedure *selectLocalizedClassesNextHops* determines the current *nextHops* of a *storedMessage* destined for the desired *classes*. First, each current direct neighbor that belongs to one of the destination *classes* is added to the set of *nextHops*. Then, additionally, a *primaryNextHop* is selected in Procedure *selectPrimaryClassesNextRelay*. This next hop is one of the relay nodes with the largest number of destination classes neighbors, i.e., highest classes node degree. In such a manner, the *storedMessage* is forwarded to the relay which is currently able to reach the largest number of other nodes of the destination classes of interest. Next, if *primaryNextHop* was chosen, Procedure *selectSecondaryNodesNextRelay* 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 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 the routing loop avoidance mechanisms is implemented and resource utilization is lowered.

Procedure selectLocalizedClassesNextHops(storedMessage)

Input:
storedMessage // stored message

Output:
nextHops // set of next hops

nextHops ← ∅ // initialize next hops as an empty set
neighbors ← *router.neighbors* // get the set of current neighbors
destinations ← *storedMessage.destination.classes* // get the set of destination classes
// select direct next-hop neighbors that belong to a destination class
foreach *neighbor* ∈ *neighbors* **do**
    if *neighbor* ∉ *storedMessage.visited* **then**
        if *neighbor.class* ∈ *destinations* **then**
            nextHops ∪← *neighbor.id* // add neighbor to the set of next hops
// select primary next-hop relay
primaryNextHop ← *selectPrimaryClassesNextRelay*(*destinations*)
if *primaryNextHop* ≠ ∅ **and** *primaryNextHop* ∉ (*storedMessage.visited* ∪ *nextHops*) **then**
    nextHops ∪← *primaryNextHop*
// select secondary next hop (relay) if primary was selected
if *primaryNextHop* ≠ ∅ **then**
    secondaryNextHop ← *selectSecondaryNextRelay*(*primaryNextHop*)
    if *secondaryNextHop* ≠ ∅ **and** *secondaryNextHop* ∉ (*storedMessage.visited* ∪ *nextHops*) **then**
        nextHops ∪← *secondaryNextHop*
output *nextHops*

Procedure selectPrimaryClassesNextRelay(destinations)

Input:
destinations // set of destination classes

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

primaryNextRelay ← ∅
maxNeighborsNumber ← −∞
neighbors ← *router.neighbors* // get the set of current neighbors
foreach *neighbor* ∈ *neighbors* **do**
    if *neighbor.role* = "*relay*" **then**
        destinationNeighborsNumber ← 0
        // check if neighbor has neighbors belonging to destination classes
        foreach *class* ∈ *neighbor.nodes.classes* **do**
            if *class.name* ∈ *destinations* **then**
                // add the number of neighbor's neighbors of given destination class
                destinationNeighborsNumber + ← *class.number*
        // set current neighbor as primary next relay when it has more destination classes neighbors
        if *destinationNeighborsNumber* > *maxNeighborsNumber* **then**
            primaryNextRelay ← *neighbor*
            maxNeighborsNumber ← *destinationNeighborsNumber*
output *primaryNextRelay*

3.2.3. Localized Next Hops to Destination Regions

Procedure *selectLocalizedRegionsNextHops* chooses the *nextHops* of a *storedMessage* which lead to nodes in the desired destination *regions*. This particular type of multicasting is usually called *geocasting* [21]. For each *destination* region it is checked if each *neighbor* is located within this region. If it is, then it is set as the 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 the *destination* region, to improve message dissemination coverage.

Procedure selectLocalizedRegionsNextHops(storedMessage)

Input:
storedMessage // stored message

Output:
nextHops // set of next hops

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

3.3. Global Knowledge Opportunistic Multicast

Algorithm 3, *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 the 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.

This seemingly idealistic scenario is designed for applications characterized by high or complete predictability of the node locations. This 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 [2]. It is also related to the abstract concept of *knowledge oracles*, which provide the nodes with the information on possible contacts in the 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 [22].

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 the *"nodes"* type is used. This approach is based on the observation that since the 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.

Algorithm 3: Global knowledge opportunistic multicast

```
foreach storedMessage ∈ storedMessages do
    outgoingMessage ← makeGlobalKnowledgeOutgoingMessage(storedMessage)
    if outgoingMessage ≠ ∅ then
        transmitMessage(outgoingMessage)
        updateStorage(storedMessage)
receiveMessages()
```

Procedure makeGlobalKnowledgeOutgoingMessage(storedMessage)

```
Input:
storedMessage                                              // stored message
Output:
outgoingMessage                                            // outgoing message
outgoingMessage ← ∅                          // set outgoing message to an empty set
neighbors ← router.neighbors                 // get the set of current neighbors
if neighbors ≠ ∅ then
    nextHops ← storedMessage.nextHops                      // get the set of next hops
    foreach neighbor ∈ router.neighbors do
        if neighbor ∈ nextHops then
            outgoingMessage ← makeOutgoingMessage(storedMessage, nextHops)
            // break the loop because at least one next-hop neighbor exists
            break
output outgoingMessage
```

3.4. Observations

The 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 operate 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 the JSON structure and the flowcharts 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 the 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 the network topology dynamics, is a method for local optima avoidance.

Since the presented routing mechanisms are designed for delay-tolerant networks, a message is not only simply forwarded from node to node but it is also required 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 the desired delivery rate, all related aspects and parameters have to be thoroughly considered and carefully planned.

4. Delay-Tolerant Multicast Router

The algorithms introduced in Section 3, 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, though, more routing and storage management aspects related to message transmission in such a structure, not necessarily 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. The generality of the description is intentionally maintained so as to not obscure the concepts and keep them comprehensible to all interested parties. The implementation details will vary depending on the actual software and hardware platforms.

The upper bound of time complexity of a single iteration of the router depends on the enabled knowledge mode. It is also related to the numbers of stored and incoming messages, neighbors (in the case of LKOM and GKOM), and maximum number of destination elements among the 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: $\mathcal{O}(|storedMessages| \cdot |neighbors| + |incomingMessages|)$.

A general flowchart of the proposed delay-tolerant multicast router is depicted in Figure 4, following the convention described in Section 3. Six main procedures are marked with light- and dark-gray backgrounds. Further components are presented as respective blocks. For details, please refer to Sections 4.1 and 4.2.

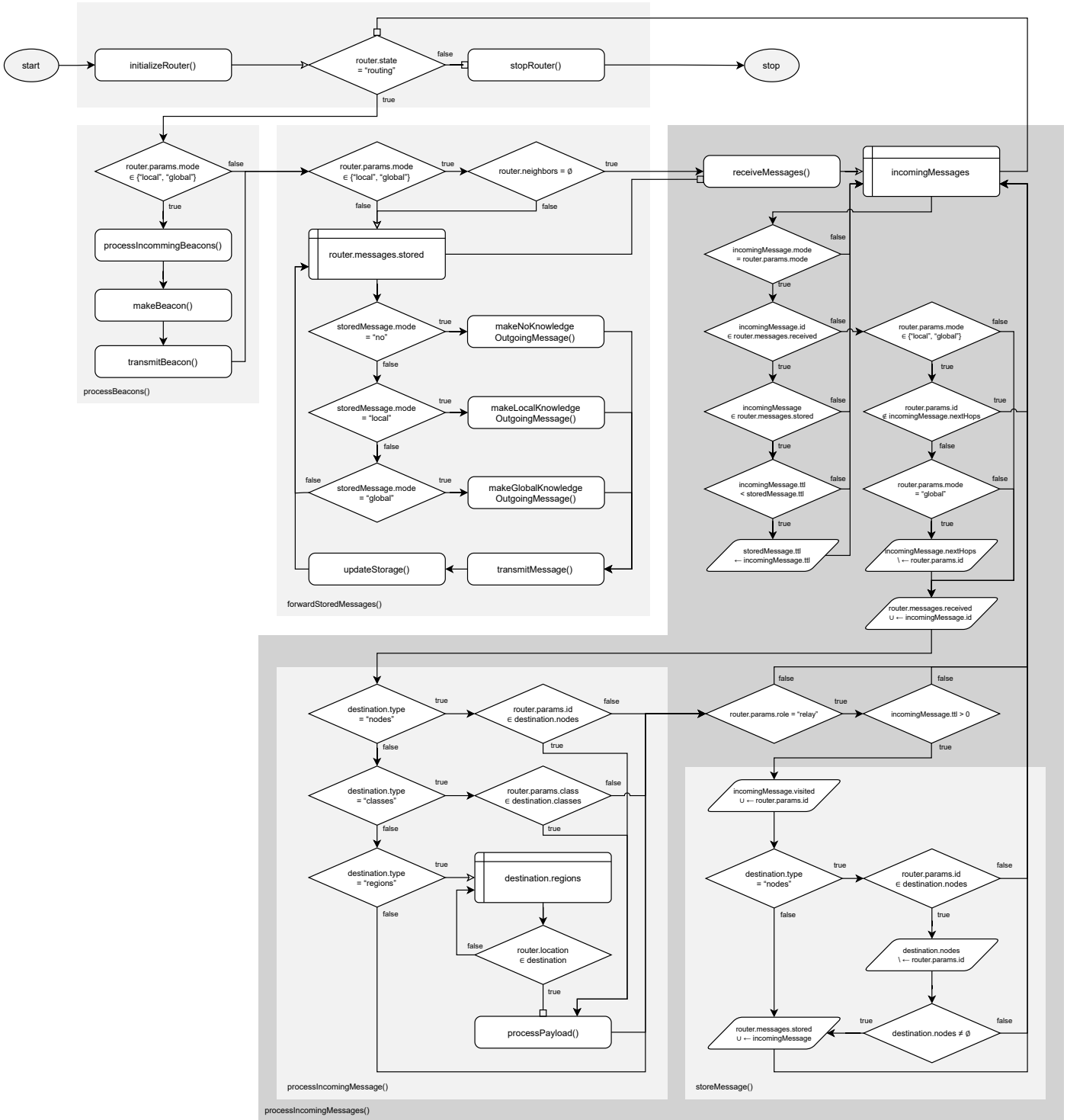


Figure 4. Delay-tolerant multicast router flowchart.

4.1. Basic Router Operations

The high-level steps of Algorithm 4, *delay-tolerant multicast router* (DTMR), are as simple as possible. First, the *router* is 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 the particular usage scenario, as well as the capabilities and network

role of the particular device. When a new routing process starts, Procedure *initializeRouter* is called. Then, an initial *router* instance is created with an empty geographic *location* of the node, lists of *received* and *stored* messages, the set of *neighbors*, and the *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"* .

Algorithm 4: Delay-tolerant multicast router

```
router ← initializeRouter()                    // initialize router (routing process) of the node
// continue routing as long as router state is set to ''routing''
while router.state = "routing" do
    router.location ← getNodeLocation()        // set current location of the node
    processBeacons()
    forwardStoredMessages()
    processIncomingMessages()
stopRouter(router)
```

Procedure initializeRouter

```
Output:
router                                                          // router instance
router ← (                                  // initialize empty (start) state of the router
    location ← (                                                // location of the node
        lat ← ∅,                                                        // latitude
        lon ← ∅                                                        // longitude
    ),
    messages ← (                                                    // message lists
        received ← ∅,                       // list of identifiers of received messages
        stored ← ∅                            // list of messages stored in the router
    ),
    neighbors ← ∅,                           // set of current neighbors of the node
    params ← (                                          // parameters of the router
        class ← getNodeClass(),                                    // set class name
        id ← getNodeId(),                                          // set identifier
        mode ← getKnowledgeMode(),                    // set network knowledge mode
        role ← getNodeRole()                                       // set network role
    ),
    state ← "routing"                          // set router state to ''routing''
)
output router
```

Next, the looped routing process starts and proceeds for as long as the *state* of the router is set to *"routing"*. There, the 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 the radio beacons transmitted by each node. The procedure starts with calling Procedure *processIncomingBeacons* is designed to receive and process the *incomingBeacons* to compile the set of current *neighbors*. As presented in Procedure *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' messages are generated by each *router* following Procedure *makeBeacon* and transmitted.

Procedure processBeacons

```
if router.params.mode ∈ {"local", "global"} then
    processIncomingBeacons()
    beacon ← makeBeacon()
    transmitBeacon(beacon)
```

Procedure processIncomingBeacons

```
incomingBeacons ← receiveBeacons()                                  // receive beacons
// determine current neighbors of the node
if incomingBeacons ≠ ∅ then
    router.neighbors ← getCurrentNeighbors(incomingBeacons)
```

Procedure getCurrentNeighbors(incomingBeacons)

Input:
incomingBeacons // incoming beacons

Output:

```
neighbors ← {neighbor_i}_{i←1}^{j} :                  // set of j neighbors of current node
  neighbor_i ← (                                                  // neighbor i
    class_i,                                                          // class
    id_i,                                                         // identifier
    location_i ← (                                                  // location
      lat_i,                                                        // latitude
      lon_i                                                         // longitude
    ),
    nodes_i ← (                          // classes and number of neighbors of neighbor i
      classes_i ← {class_k}_{k←1}^{l} :                    // set of l neighbor classes
        class_k ← (                                  // 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
foreach beacon ∈ incomingBeacons do
    neighbor ← getNeighborDetails(incomingBeacon)          // get received neighbor details
    neighbors ∪← neighbor                        // add neighbor to the set of neighbors
output neighbors
```

Procedure makeBeacon

Output:
beacon // beacon of current node

```
beacon ← (
    class ← router.params.class,                                    // set node class name
    id ← router.params.id,                                              // set node id
    location ← router.location,                                   // set node location
    nodes ← (                          // set classes and number of neighbors of current node
        // group and count neighbors by their class name
        classes ← getNeighborClassNamesAndNumbers(router.neighbors),
        number ← |router.neighbors|       // set the number (cardinality) of current neighbors
    ),
    role ← router.params.role                                       // set node role
)
output beacon
```

4.2. Message Forwarding, Processing, and Storage

In the core phase of Algorithm 4, *delay-tolerant multicast router* (DTMR), Procedure *forwardStoredMessages* is executed. If the *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 the 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*, there is an attempt to generate an *outgoingMessage*. If it is created, then it is transmitted to be received by other nodes in the radio range. Afterwards, message storage is updated, as defined in Procedure *updateStorage*.

Procedure forwardStoredMessages

```
1  if router.params.mode ∈ {"local", "global" } then
2      if router.neighbors = ∅ then
3          break        // break forwarding procedure because currently there are no neighbor nodes
   // process each stored message
4  foreach storedMessage ∈ router.messages.stored do
5      outgoingMessage ← ∅                          // 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 ← makeNoKnowledgeOutgoingMessage(storedMessage)
           // localized routing
9          case "local" do
10             outgoingMessage ← makeLocalKnowledgeOutgoingMessage(storedMessage)
           // global knowledge routing
11         case "global" do
12             outgoingMessage ← makeGlobalKnowledgeOutgoingMessage(storedMessage)
13     if outgoingMessage ≠ ∅ then
14         transmitMessage(outgoingMessage)                    // transmit message
15         updateStorage(storedMessage)                        // update storage
```

The *outgoingMessage* is prepared using one of the three introduced procedures, i.e., Procedure *makeNoKnowledgeOutgoingMessage*, Procedure *makeLocalKnowledgeOutgoingMessage*, and Procedure *makeGlobalKnowledgeOutgoingMessage*, based on the knowledge *mode* of the *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 *incomingMessage* is processed if its knowledge *mode* is the same as the knowledge *mode* the *router* operates in. Next, it is checked whether this message has already been received by the current node. This is achieved by comparing the *id* of the *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 the message storage of the current node. When it is, the *ttl* of the *storedMessage* is updated to the *ttl* value of the *incomingMessage* if the latter value is lower. This situation means that the message, after reaching the current node, has been relayed in the network more, beyond the current node, and hence, *ttl* should be updated to the lower value to reflect this message dissemination progress.

If the message has reached the current node for the first time, a different type of processing takes place. When the knowledge *mode* of the router is *"local"* or *"global"*, a check is performed to verify if the *id* of the 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 the current *incomingMessage* was not destined to this node. Otherwise, in the case of

"global" knowledge *mode*, the *id* is removed from the set of *nextHops* since it reached the desired global network knowledge next hop.

Procedure processIncomingMessages

```
    // receive internal (originated) and external (forwarded) incoming messages
 1  incomingMessages ← receiveMessages()
 2  if incomingMessages ≠ ∅ then
        // process each incoming message
 3      foreach incomingMessage ∈ 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 ∈ router.messages.received then
                    // if incoming message is present in message storage of current node
 6                  if router.messages.stored[incomingMessage.id] ≠ ∅ then
 7                      storedMessage ← 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 ← incomingMessage.ttl
                // message reached current node for the first time
10              else
11                  if router.params.mode ∈ {"local", "global" } then
12                      if router.params.id ∉ 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 \ ← router.params.id
                    // store the id of received message
16                  router.messages.received ∪ ← 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)
```

When the processing continues in the current iteration, the *id* of the *incomingMessage* is added to the set of unique identifiers of *received* messages. Subsequently, Procedure *processIncomingMessage* is executed. Based on the destination *type* of the *incomingMessage*, it performs applicable checks to asses if the message reached the current node as the member of the requested multicast group. When the target is *"nodes"* and the *id* of the *router* is one of the destination *nodes*, the payload of the *incomingMessage* is directed to be processed. Similarly for *"classes"*, when the *class* of the *router* is one of the destination *classes*. A related check is performed for *"regions"*, but in this case each *destination* region of *incomingMessage* is checked one by one to test whether the current *location* of the *router* is located within the multicast destination region. Since there may be numerous regions defined and some could be overlapping, the payload is processed and further checks are stopped for the current *incomingMessage* when the first matching *destination* region is found.

In the final stage, if the 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 *storeMessages* first adds the *id* of the *router* to the list of nodes *visited* by *incomingMessage*.

Next, if the destination type is set to *"nodes"*, the *id* of the 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.

Procedure processIncomingMessage(incomingMessage)

```
Input:
incomingMessage                                                    // incoming message
payload ← ∅
// proceed based on destination type of incoming message
switch incomingMessage.destination.type do
    // destination type set to a group of individual nodes
    case "nodes" do
        if router.params.id ∈ incomingMessage.destination.nodes then
            // payload was intended for this (current) node
            processPayload(incomingMessage.payload)
    case "classes" do
        if router.params.class ∈ incomingMessage.destination.classes then
            // payload was intended for this node class
            processPayload(incomingMessage.payload)
    // destination type set to regions
    case "regions" do
        foreach destination ∈ incomingMessage.destination.regions do
            if router.location ∈ destination then
                // payload was intended for this destination region
                processPayload(incomingMessage.payload)
                // break the loop because the node belongs to at least one destination region
                break
```

Procedure storeMessage(incomingMessage)

```
Input:
incomingMessage                                                    // incoming message to be stored
incomingMessage.visited ∪← router.params.id    // add current node to the list of visited nodes
if incomingMessage.destination.type = "nodes" then
    // if this node is one of destination nodes
    if router.params.id ∈ incomingMessage.destination.nodes then
        // remove current node from the set of destination nodes
        incomingMessage.destination.nodes \← router.params.id
        // do not store the massage if it just reached the last desired destination node
        if incomingMessage.destination.nodes = ∅ then
            incomingMessage ← ∅
if incomingMessage ≠ ∅ then
    router.messages.stored ∪← incomingMessage                      // store the message
```

5. Simulation and Analysis Methodology

The goal of the simulation study is the multi-criteria evaluation and comparison of the introduced multicast algorithms. Therefore, the simulation and analysis approach makes use of our network modeling architecture introduced in [9]. In that article, the typical characteristics of the sources of the urban nodes' location data are also discussed and multiple open-data examples are provided and analyzed. The current study uses our related modeling methodology presented and proven valid in [10]. Now, the custom-

developed urban sensor network connectivity research environment is extended to enable multidimensional studies related to modeling multicast structures.

The simulation environment is based on *Linux*, *PostgreSQL*, and *Python*. Basic graph-related data structures and multiple standard operations are provided by the *NetworkX* library [23]. The graphs are, in turn, visualized over map data provided by *OpenStreetMap* [24]. This software framework has been chosen because of its stability, flexibility, detailed documentation, and proven value for data and network scientists. The implemented modeling architecture is depicted in Figure 5. It is aimed at constructing graph network topology representations that describe the structure of the connections between the elements of the modeled urban sensor networks.

A graph is defined as a pair $G = (V, E)$, where V is the set of nodes (vertices) and E is the set of edges (links) connecting the vertices [25]. They correspond to network devices and wireless connections between them. Each node and edge can be labeled with more than one attribute (parameter), e.g., an identifier, occurrence time, or weight (cost).

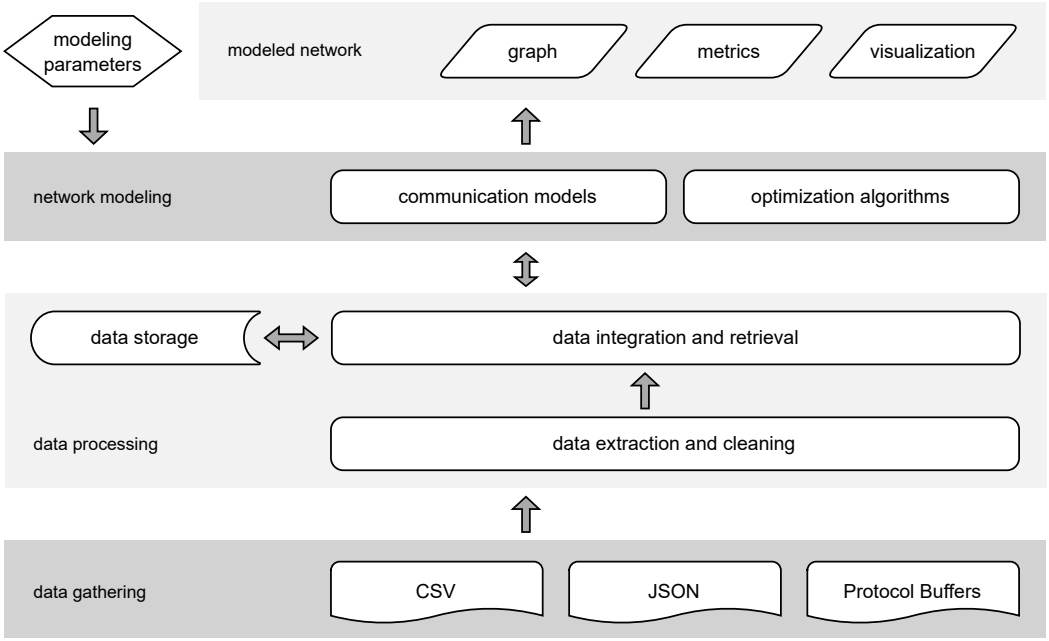


Figure 5. Open-data-based network modeling architecture [9].

The high-level steps of the simulation are depicted in Figure 6. The multicast structures are constructed and analyzed as graphs. No actual wireless propagation models, communication protocols, or power management schemes are simulated. In this way, technology and protocol agnosticism is guaranteed in every aspect. By this means, the efficacy and efficiency of the algorithms can be studied using graph theory methods. Therefore, the key aspects and metrics of the modeled networks can be scrutinized.

A single multicast modeling process starts with modeling space connectivity. The network devices' data are filtered and grouped into subsequent intervals, i.e., time *slots* of the space nodes. Then, the *slots* are turned into a time-ordered sequence of *space connectivity graphs* (SCGs) grouped as a *space connectivity list* (SCL). Next, these time-spanning time-changing network topologies are represented as directed *space–time connectivity graphs* (STCGs). Links in those graphs obey the direction (flow) of time. Finally, *space–time multicast graphs* (STMGs) are constructed in the *STCGs* using the introduced opportunistic multicast algorithms—NKOM, LKOM, and GKOM.

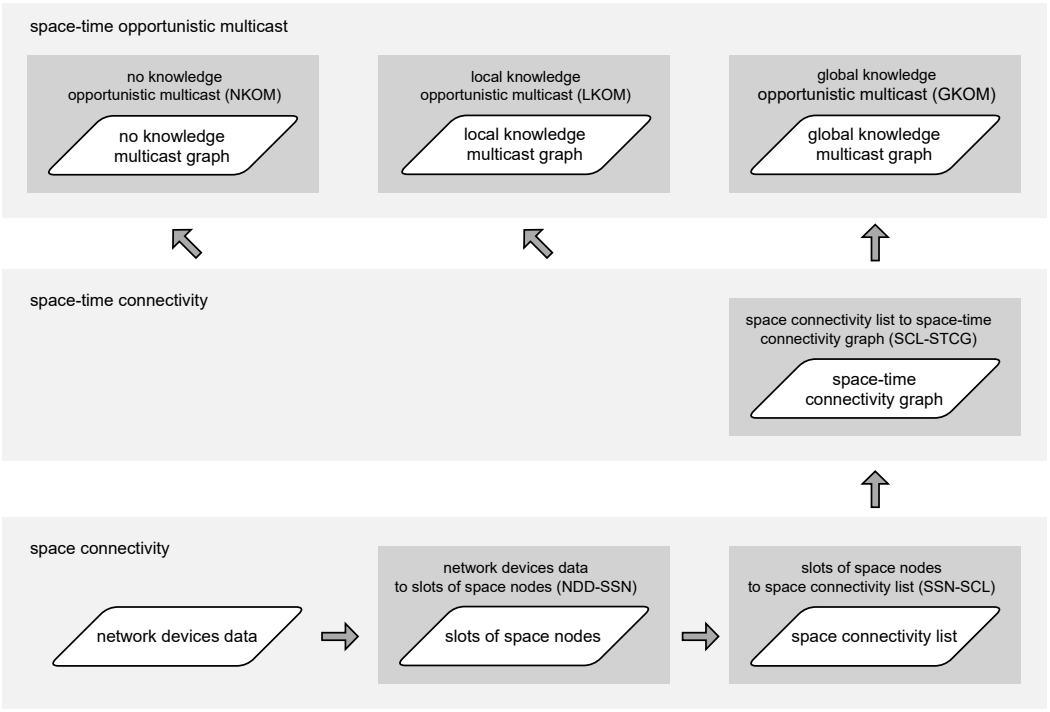


Figure 6. Flow for modeling opportunistic delay-tolerant multicast.

An example space–time connectivity graph in Figure 7 is presented in the *time-expanded* form. We have designed this form primarily as an intermediate structure which enables the usage of algorithms designed for traditional (static) graphs and network representations. Each node instance is represented as slot start- and end-instances, with an additional slot zero, which is needed as the starting point for some of the graph processing algorithms. Stationary simple and advanced, as well as mobile advanced nodes are distinguished, together with the respective space and time edges connecting them. An example expanded space–time multicast graph depicted in Figure 8 was constructed in the space–time connectivity graph presented in Figure 7.

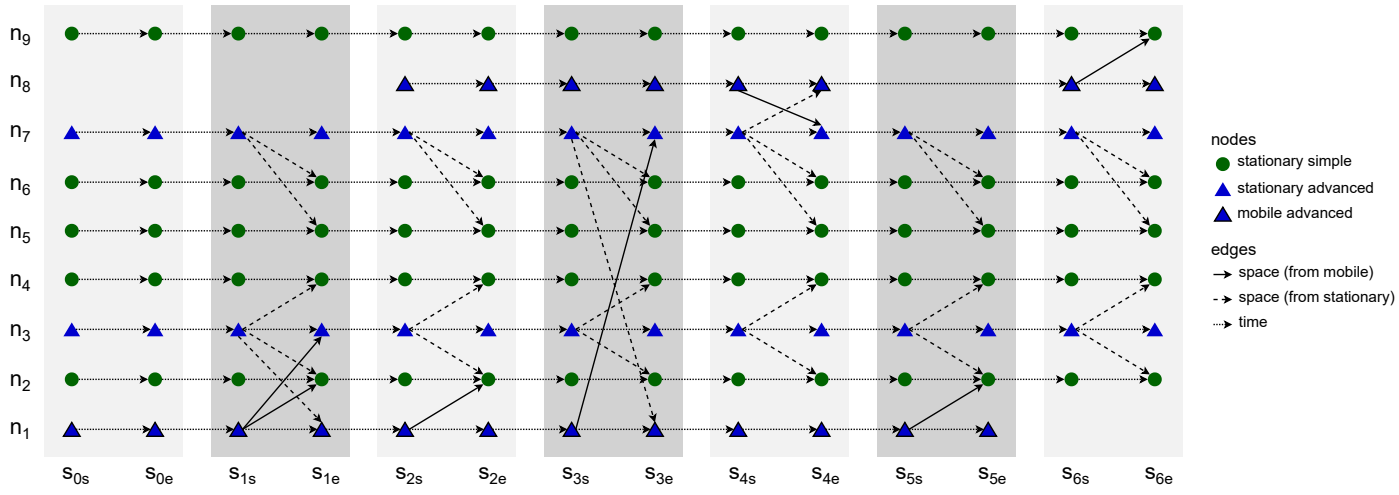


Figure 7. Example space–time connectivity graph [10].

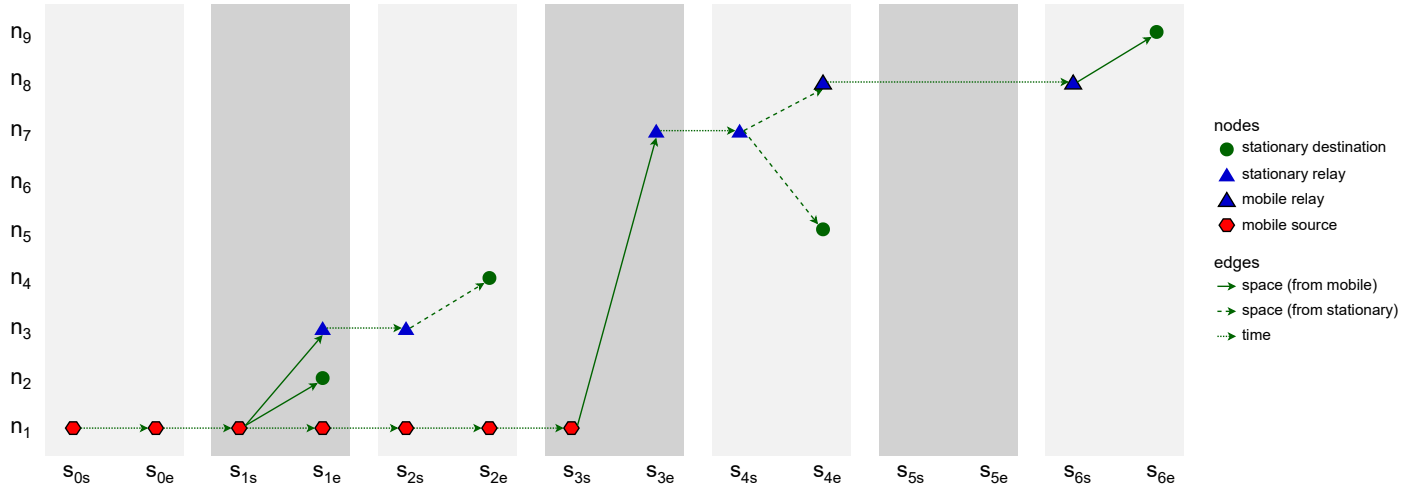
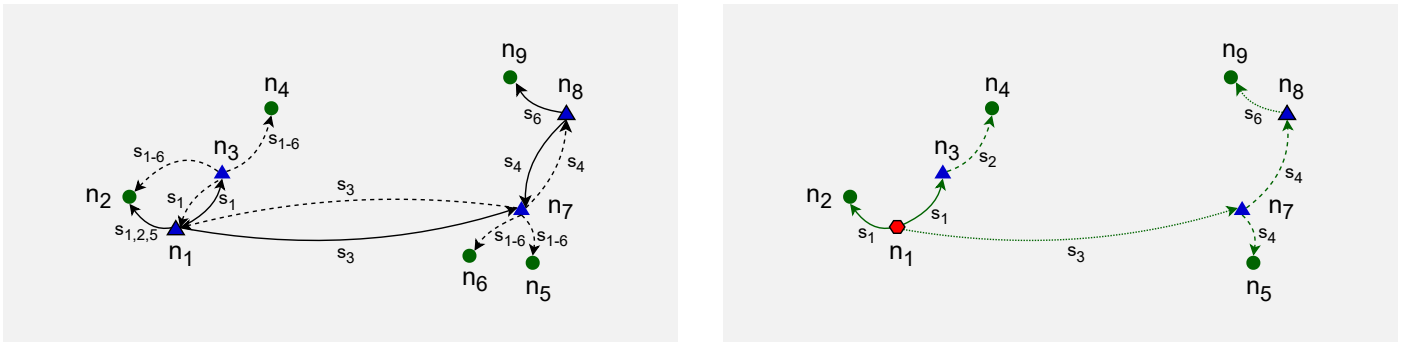


Figure 8. Example expanded space–time multicast graph [10].

The more reader-friendly *time-aggregated* equivalents of the example graphs are depicted in Figure 9a,b. They use the symbols presented in the legends of the related Figures 7 and 8, respectively. This representation is well-fitted to capture the outcomes of the algorithms that solve problems in space–time graphs. Therefore, it is used in the present research as a practical representation of the modeled multicast graphs. The depicted edge labels, for instance, s_2 and s_6, indicate in which time *slot* an edge existed between two nodes. The time-respecting multicast tree visible in Figure 9b connects, over time, mobile source node n_1, through intermediate relay nodes, with four stationary destination nodes n_2, n_4, n_5, and n_9.



(**a**) Space–time connectivity graph.

(**b**) Space–time multicast graph.

Figure 9. Example time-aggregated graphs [10].

For a comprehensive discussion of the used time-changing network connectivity modeling approach and graph representations, please refer to our original paper [10].

5.1. Comparative Study Methodology

There exist no equivalent algorithms designed for multicast in heterogeneous urban sensor networks. Therefore, the introduced algorithms are compared with one another. For general metrics, graphs constructed with no network knowledge are the baseline of the comparisons. This is dictated by the fact that no-network-knowledge approach is the simplest one. The local knowledge multicast is related but more sophisticated. In both cases, the 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 the multicast source, with the shortest-time paths, only to the desired destinations. Therefore,

for the core multicast structure metrics, the global knowledge graphs become the baseline. The spatiotemporal shortest paths are determined with Dijkstra's algorithm [26] applied to each space–time connectivity graph with the 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 the multicast tree. The space–time connectivity graphs are directed, which is caused both by the modeled arrow of time and by the different roles of different classes of the nodes. The multicast tree could be alternatively modeled as a *directed Steiner tree* (DST), which minimizes the overall cost of the tree, but there exists no known exact algorithm to solve this problem [27], and hence, to be used to model an optimal baseline solution.

5.2. Statistical Analysis and Visualization

Each step of the simulation and modeling is accompanied by the generation of related statistical data. A number of metrics are computed using custom-developed functions, while others are calculated by the methods of the NetworkX graph modeling framework. The data are processed as data structures called pandas DataFrames [28]. They enable complex multidimensional data consolidation, filtering, and statistical processing. Due to the complexity and nature of the study, individual data points and their numerical aggregates are not the center of attention. The main concern are the relationships between the parameters of the modeled network structures and the trends they display. Therefore, the investigated data are presented with the Seaborn data visualization library [29].

The analyzed data are discrete, and hence, the sets of interest are depicted as scatter plots. Different subsets of the data are represented with different point styles. To make the trends clearly visible in large and overlapping data sets, regression lines are overlaid on the plots, as visible in Figure 15. Related sets of data are grouped in named rows and columns of subplots, as the ones that pertain to different cities and radio ranges in Figure 18. They are categorical, which means that they shift and group the data horizontally around the values of interest to make the categories, such as city or knowledge mode, easily distinguishable. Additionally, small jitter, i.e., random deviations, is introduced to the horizontal distributions of the categories. This makes the otherwise overlapping points more visible. Moreover, relationships between the sets of variables are presented in pair plots, as in Figure 41. To address the multidimensional nature of the data, the figures related to the subsets of the investigated metrics are grouped as well as commented in dedicated sections, e.g., in Section 6.2, focused on the the cost metrics of the modeled multicast structures.

5.3. Simulation Data Sources and Node Classes

The conducted research and analysis presented in [9] indicate that open data sources which meet the requirements of the study exist for only four Polish cities. Other Polish urban areas do not yet provide any similar data or provide them in a limited scope. The attempt to find equivalent open data sets related to urban areas in other countries was not successful. Therefore, the sources for Gdańsk, Poznań, Warsaw, and Wrocław are used as the ones that provide comparable data scope, granularity, and update frequency, as listed in Table 1.

From the gathered data, the geographic coordinates expressed in *World Geodetic System '84* (WGS 84) notation [30] are extracted and transformed into modeled network connectivity topologies using the algorithms introduced in [10]. In all areas of interest, the locations of buses and trams are available in real time. In Gdańsk and Poznań, the coordinates of public transport stops and ticket machines are provided. In Warsaw and Wrocław, no data on the locations of ticket machines are available. In the case of Wrocław, the locations of city bike rental stations and parking lots of Vozilla (city electric car rental service) can be used instead. The most frequently used data format is currently *JavaScript Object Notation* (JSON). Only from Poznań are mobile node data provided in the *Protocol Buffers* (protobuf) encoded and minimized form. The update frequency of the data varies.

For the continuously updated ones it spans from a few seconds to 10 s for Warsaw and 20 s for Gdańsk. The locations and number of stationary nodes change much more infrequently, e.g., for ticket machines and air quality meters. Those location-related data can remain unchanged for hours or days. To allow heterogeneous networks to be modeled, the sources of data are classified into three meaningful logical classes—mobile advanced, stationary advanced, and stationary simple. Each class is assigned one of the basic network roles:

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

The advanced nodes are assumed to be of more significant computing, power, storage, and communications resources. This means that they can be used to perform complex *delay-tolerant network* (DTN)-forwarding operations. Conversely, simple nodes are merely the recipients of the communications.

Table 1. Urban open data sources used in simulation study.

City	Class	Scope	Format	Updates	Provider
Gdańsk	Mobile advanced	Buses and trams [31]	JSON	20 s	Open Gdańsk
	Stationary simple	Public transport stops [32]	JSON	24 h	Open Gdańsk
	Stationary advanced	Ticket machines [33]	JSON	24 h	Open Gdańsk
Poznań	Mobile advanced	Buses and trams [34]	protobuf	Continuous	ZTM Poznań
	Stationary simple	Public transport stops [35]	JSON	Infrequent	Poznan city hall
	Stationary advanced	Ticket machines [36]	JSON	Infrequent	Poznan city hall
Warsaw	Mobile advanced	Buses and trams [37]	JSON	10 s	City of Warsaw
	Stationary simple	Public transport stops [38]	JSON	Infrequent	City of Warsaw
Wrocław	Mobile advanced	Buses and trams [39]	JSON	Continuous	Open data Wrocław
	Stationary simple	City bike rental stations [40]	JSON	5 min	Open data Wrocław
	Stationary simple	Vozilla parking lots [41]	JSON	Continuous	Open data Wrocław
All	Stationary advanced	Air quality meters [42]	JSON	Continuous	Airly

5.4. Simulation Areas and Example Modeled Networks

The four cities of interest are among the most populated and largest ones in Poland. The key numbers related to Gdańsk, Poznań, Warsaw, and Wrocław are presented in the next sections.

The most populated city is Warsaw, which is more than two and a half times more populated than Wrocław. The population of Wrocław, in turn, is twenty three percent larger than the one of Poznań. Similarly, Poznań is thirteen percent more populated than Gdańsk. The densities of the populations vary similarly, where Gdańsk is the least densely populated area, preceded by Poznań, Wrocław, and Warsaw, which is almost two times more densely populated than Gdańsk. Surprisingly, the expected number of public transport day routes (lines) is the lowest in Poznań, and is a little larger in Gdańsk and Wrocław. In Warsaw, twice as many routes on average are to be found operating throughout the city.

In each of the cities an area of 3 by 2 km was selected for the study. They were chosen to cover partially distinct and partially alike urban regions, which include both the busy city center as well as less dense surroundings. Each city topology, as presented in the figures, indicates unique street, building, terrain, and infrastructure layouts. As a result, the modeled networks are likely to express both similarities and differences. Example time-changing multicast graphs are built in the areas of interest based on the gathered data starting on Wednesday 27 November 2019, at 3:00 p.m.

For clarity, simple omnidirectional radio coverage is assumed. The relays are marked as dark blue triangles and destination nodes are displayed as pink circles. The mobile nodes are additionally designated with black borders. Every node is depicted in the geographic

location it occurred for the first time in the period of interest. Multicast destination regions are marked with red dashed rectangles. Space connections, i.e., the ones that occur without message buffering (in the same time interval), are marked as solid links. Pink dotted lines are multicast core space–time connections, i.e., the ones that lead to the destinations and require message buffering in the relay. Dark-blue dashed space links and dash-dot space–time links do not lead to destination nodes. Label 126 (93 s, 73 m) in Figure 10 shows that the edge exists in slot (modeled time interval) number 126, the message has to be buffered for 93 slots in the relay before being forwarded, and that the space distance between the relay and the next-hop node, at the moment of message forwarding, is 73 m.

5.4.1. Gdańsk

- Population:
  - Total: 486 thousand [43] in a metropolis of around one million in northern Poland;
  - Density: 1797.00 per km^2 [44];
- Public transport day routes: around 80 [45];
- Simulation area:
  - Latitude: 54.34398–54.36191;
  - Longitude: 18.62036–18.66666;
- Example no-network-knowledge multicast graph to destination nodes in Figure 10:
  - 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: 7716.00 m, time cost: 8073.00, time span cost: 4944.00, core nodes: 22.

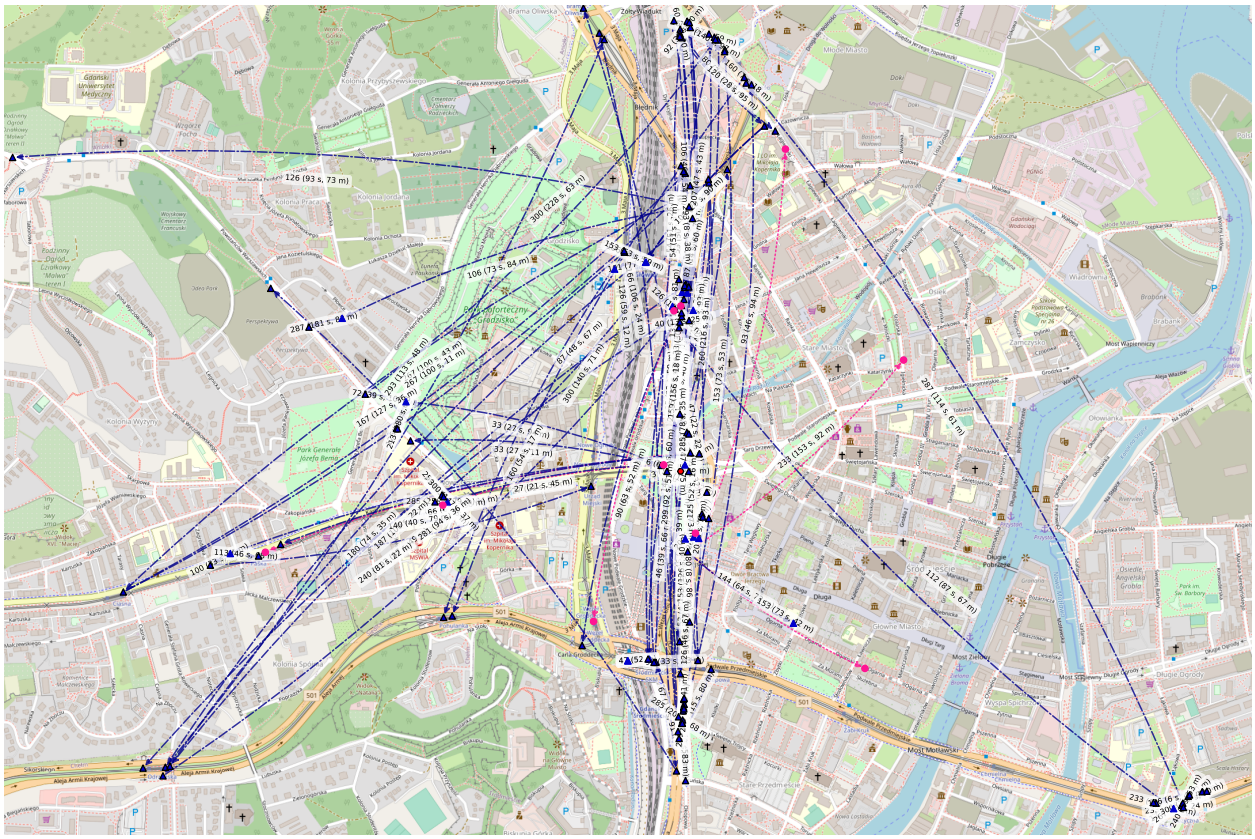


Figure 10. Example no-network-knowledge multicast to destination nodes in Gdańsk.

5.4.2. Poznań

- Population:
  - Total: 547 thousand [46] in a metropolis of almost one million in west-central Poland;
  - Density: 2031.00 per km^2 [44];
- Public transport day routes: around 70 [47];
- Simulation area:
  - Latitude: 52.39853–52.41645;
  - Longitude: 16.88965–16.93389;
- Example local knowledge multicast graph to destination classes in Figure 11:
  - 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: 19,406.00 m, time cost: 16,319.00, time span cost: 7594.00, core nodes: 150.

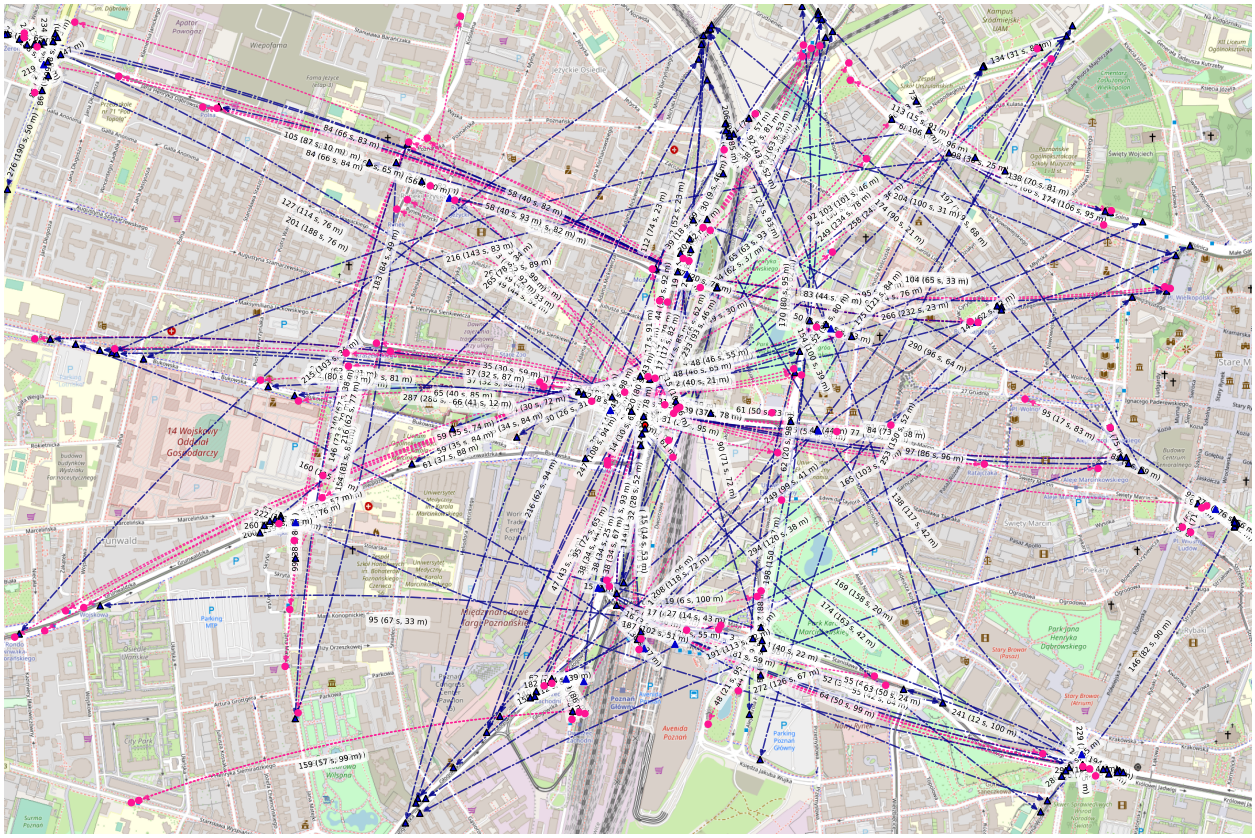


Figure 11. Example multicast to destination classes with local network knowledge in Poznań.

5.4.3. Warsaw

- Population:
  - Total: 1.794 million [48] in a metropolis of 3 million in east-central Poland;
  - Density: 3469.00 per km^2 [44];
- Public transport day routes: around 190 [49];
- Simulation area:
  - Latitude: 52.22082–52.23879;
  - Longitude: 20.97058–21.01454;

- Example local knowledge multicast graph to destination regions in Figure 12:
  - 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: 21,391.00 m, time cost: 17,580.00, time span cost: 11,599.00, core nodes: 80.

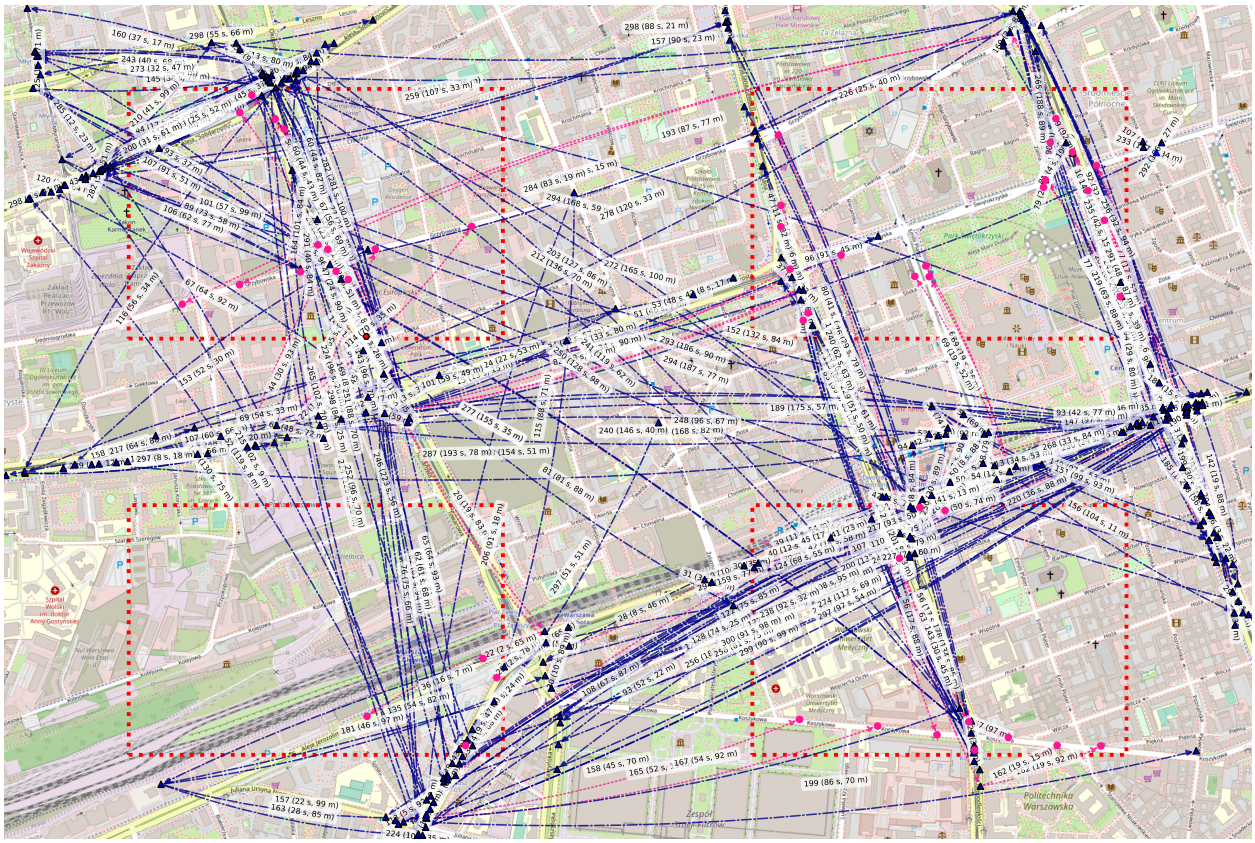


Figure 12. Example multicast to destination regions with local network knowledge in Warsaw.

5.4.4. Wrocław

- Population:
  - Total: 674 thousand [50] in a metropolis of around 1.25 million in south-western Poland;
  - Density: 2192.00 per km^2 [44];
- Public transport day routes: around 85 [51];
- Simulation area:
  - Latitude: 51.10015–51.11813;
  - Longitude: 17.01273–17.05570;
- Example global knowledge multicast graph to destination regions in Figure 13:
  - 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: 2411.00 m, time cost: 825, time span cost: 641, core nodes: 43.

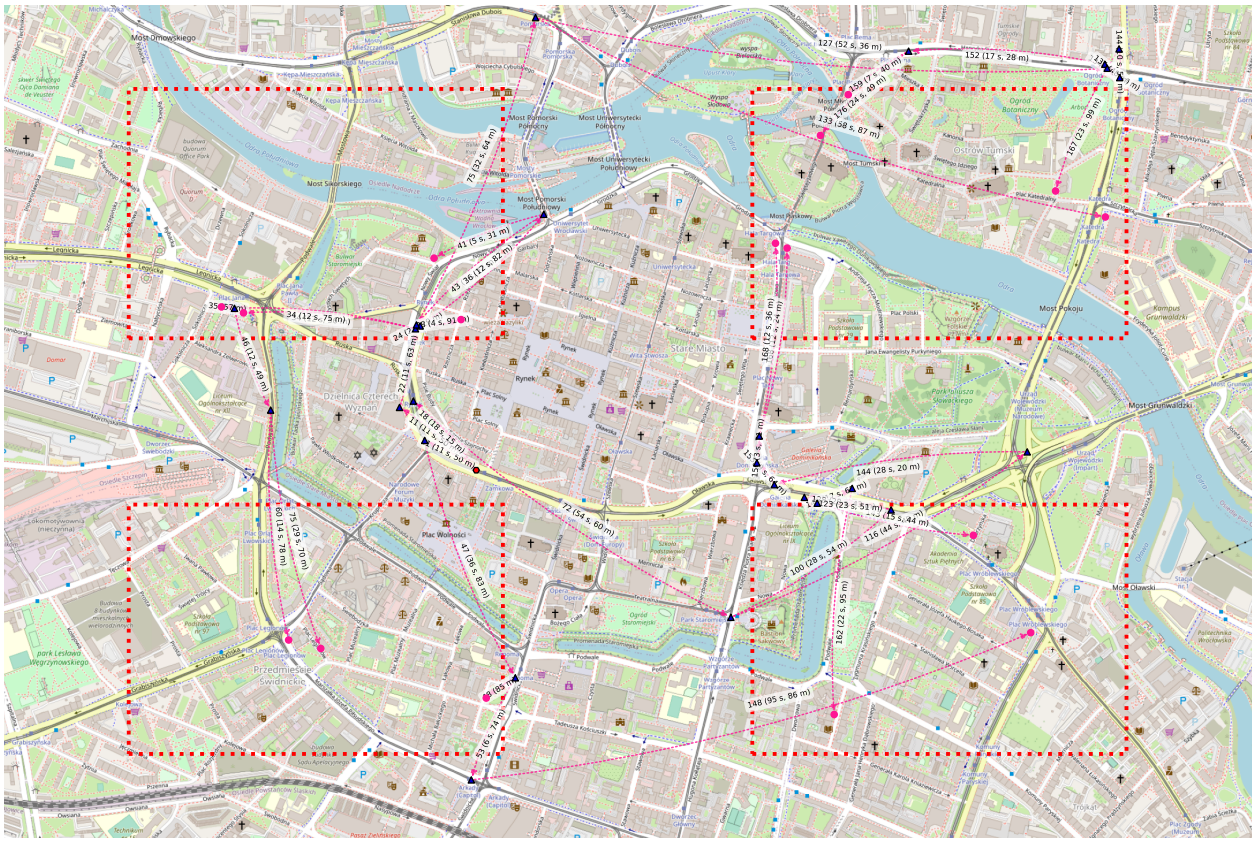


Figure 13. Example multicast to destination regions with global network knowledge in Wrocław.

5.5. Simulation Architecture and Parameters

The designed simulation architecture is based on object-oriented data structures implemented as a hierarchy of nested lists, presented in Figure 14. This approach enables multi-faceted modeling and analysis of multicast communication. The simulations are performed with the listed input parameters for the algorithms, which are the main steps in the modeling flow depicted in Figure 6. The main simulation scope characteristics (numbers), denoted with single capital letters, resulting from the architecture and parameters are also given:

- Algorithm *network device data to slots of space nodes* (NDD-SSN) [10]:
  - 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 *slots of space nodes to space connectivity list* (SSN-SCL) [10]:
  - radio coverage: omnidirectional;
    - radio ranges: (25 m, 50 m, 100 m) $\Rightarrow Q = 3$;

        * space distance: great-circle distance between two nodes.
- Algorithm *space connectivity list to space–time connectivity graph* (SCL-STCG) [10]:
  - 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 4, *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.

The simulations are performed in four urban areas of interest, defined in Section 5.4. They are based on the data gathered on Wednesday 27 November 2019 between 3:00 p.m. and 5:00 p.m. This 2 h period includes afternoon rush hours in the middle of a work week, which covers different modeling and analysis scenarios. The period is then divided in Algorithm *network device data to slots of space nodes* (NDD-SSN) into topologies the durations of which are defined by the *topologyLength* parameter. The chosen values are 75, 150, 300, 600, and 1200, which are the subset of a geometric sequence with a common ratio of 2. They are the number of space connectivity graphs in the space connectivity list of a given topology. This division enables the modeling and study of network structures that are related but have different time spans and properties. In this way, the trends related to the aspects of scalability, efficacy, efficiency, and optimization can be studied.

Since the *slotLength* is set to 6 s, a series of topologies of 16, 8, 4, 2, and 1 space–time connectivity graphs are constructed that last for 7.5, 15, 30, 60, and 120 min, respectively. The time of *slotLength* is expected to be sufficient to successfully transmit a message between two neighboring nodes. Unlimited message storage (buffer) is assumed in the relays. Three effective radio ranges, i.e., 25 m, 50 m, and 100 m, are used to model omnidirectional radio coverage. These values are based on empirical observations that current popular sensing-related short-range wireless connectivity technologies are expected to provide up to around a 100 m outdoor range at higher transmission throughputs in urban non-line-of-sight scenarios. The actual range depends on the topology of the area, obstacles, wireless medium utilization, the transmit power [52], and data transmission parameters [53]. The space distance between two devices is computed with the haversine formula, that calculates the great-circle distance between two points on a sphere [54].

The sources of open data on the location of the nodes are grouped into three classes—mobile advanced, stationary simple, and stationary advanced, as presented in Table 1. Each location of a node is timestamped. A defined data-lookup window is related to every class. Their widths (durations) are determined based on an analysis of the update frequencies of the sources. The data related to mobile nodes are updated most frequently, which can be as often as every few seconds. Hence, a window of 10 s ensures that location changes will be reflected correctly in the modeled network graphs. This enables the NDD-SSN algorithm to correct short node data or source outages. The locations of the stationary nodes change

or are updated not more frequently than once every 24 h. Therefore, this interval is used as the window for fixed nodes.

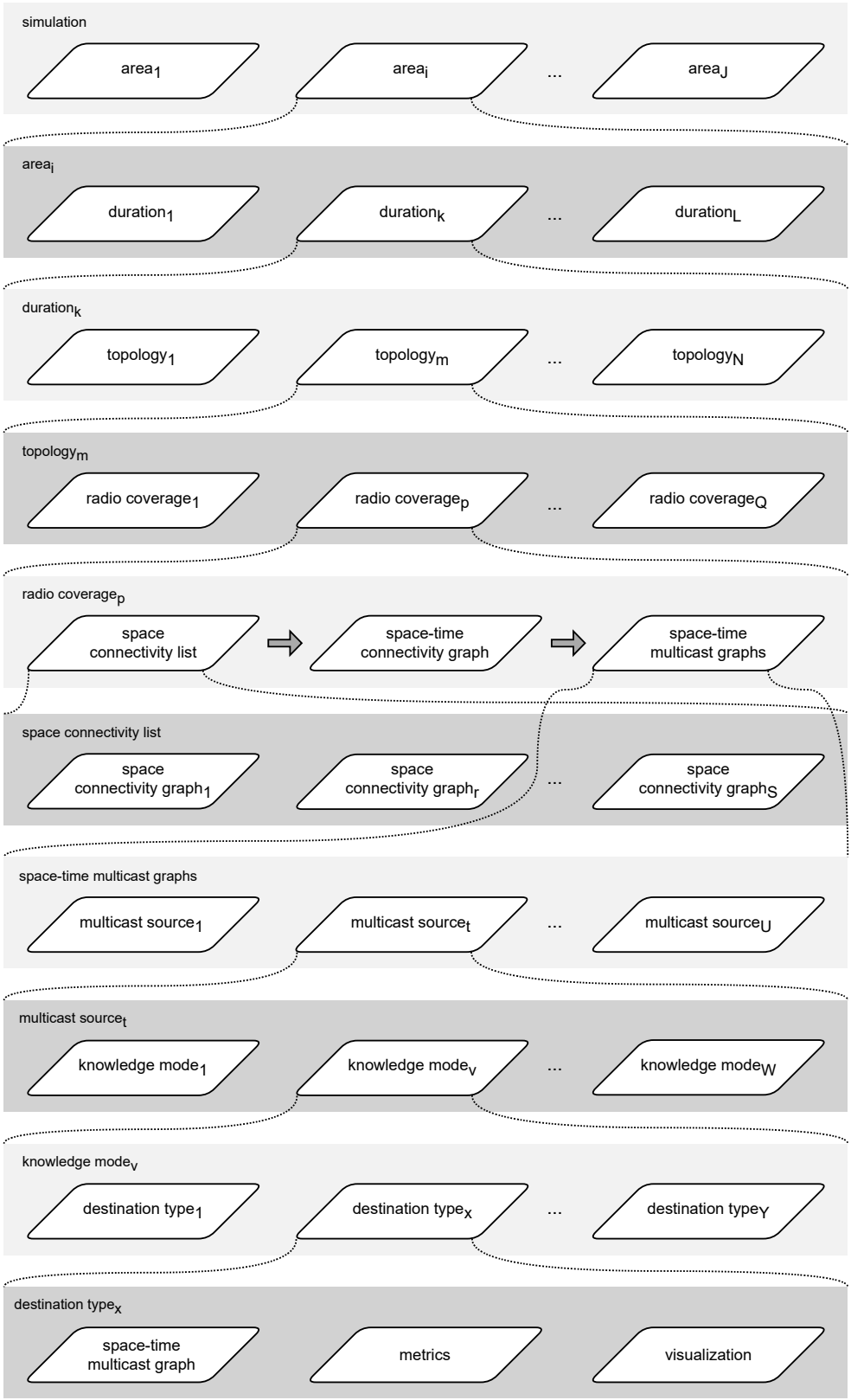


Figure 14. Simulation architecture.

The space–time connectivity graphs (networks) are modeled for each of the 31 space connectivity lists. The algorithm SCL-STCG uses the default values of the unit costs. Therefore, as explained in [10], the correct time-shortest paths can be found using Dijkstra's algorithm, based on the time-distance weight (cost) of the edges. These parameters are the means of balancing the buffering time with message forwarding, and hence, storage resource use with transmission-related power consumption of the relays. Time distance in these graphs expresses how much time, i.e., time slots, has to pass before the device (node) will be close enough to the neighboring device to establish a wireless link.

The time-spanning multicast graphs are modeled for a single message originated by a source. Their durations are limited by setting the *time-to-live* (TTL) of the message to the respective *topologyLength* value. Before constructing multicast structures in a 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 the modeled structures. This process is based on the use of a uniform random number generator and repeated to select five sources in the first graph of the 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 the 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 a 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 be 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. The time distance of an edge in a 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 five different multicast sources are selected, it results, together with nine destination type and network knowledge mode combinations, in the maximum of 45 separate multicast graphs constructed in each of 93 topologies in each of four areas of interest. Some of the graphs may not be constructed when no nodes meet the criteria. Others might consist only of the source when the source node does not establish any connections, e.g., due to insufficient radio coverage or routes. The chosen simulation scope and parameters resulted in 16362 multicast graphs being modeled. When constructed, the metrics and visualization of each space–time multicast graph were produced and stored.

5.6. Simulation Study Metrics

The metrics in this section are defined in relation to the main usage they are intended for. A number of related metrics are used as well. The parameters are divided into the ones that pertain to nodes (devices) and those that relate to edges (connections) of the graphs (networks). In the next sections, the studied metrics are first depicted in absolute values, then some are presented as ratios (percentages) related to the reference ones.

- Node metrics:
  - all nodes—total number of nodes;
  - mobile relay nodes—the number of mobile relays;
  - mobile relay nodes to all nodes ratio—the percentage of mobile relay nodes as compared to the number of all nodes;
  - potential destinations—the nodes that could be multicast destinations in the space–time connectivity graph of interest;
  - reached to potential destinations ratio—the percentage of multicast destination nodes that were reached;
  - nodes ratio to no knowledge—the percentage of nodes in graphs constructed with local and global network knowledge, as compared to the number of nodes when no network knowledge was used;
  - mobile relay nodes ratio to no knowledge—the percentage 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;
  - core nodes ratio—the percentage of core multicast nodes as compared to the number of all nodes.
- Edge metrics:
  - space cost—the sum of space distances of all edges. It can be interpreted as an indicator of the total power required to transmit the messages;
  - time cost—the sum of time distances of all edges. It can be used as an indicator of the total computing resources needed to propagate the message to the leaves of the tree;
  - time span—the number of time intervals (slots) covered by multicast graph. It is the maximum time reach of the message originated by the source node. It can be interpreted as the maximum delivery delay of a given message;
  - 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.

All of the used metrics are non-negative.

6. Space–Time Multicast Analysis

The opportunistic multicast algorithms introduced in Section 3 were simulated in *space–time connectivity graphs* (STCGs) analyzed in [10], in accordance with the simulation parameters defined in Section 5.5. These graphs were constructed based on four series of 1200 *space connectivity graphs* (SCGs) modeled for the urban areas of interest in Gdańsk, Poznań, Warsaw, and Wrocław. As presented in Figure 15, the numbers of nodes in the cities vary over time, and yet, express visible trends and expected average values. They are related to the varying-over-time numbers of mobile relays traversing the studied areas; please compare to Figure 16. A thorough discussion of numerous aspects of the modeled time-changing wireless networks can be found in [10].

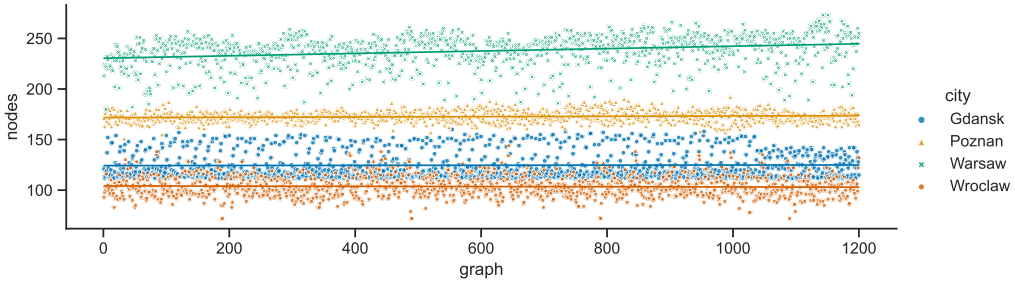


Figure 15. All nodes in space connectivity graphs [10].

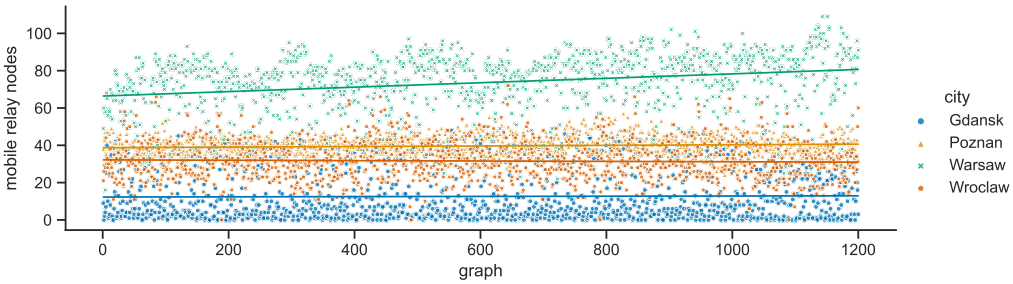


Figure 16. Mobile relay nodes in space connectivity graphs [10].

In total, 16,362 *space–time multicast graphs* (STMGs) were constructed in *time-aggregated* form 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 a given city area, destination type, and duration, as presented in Figure 17. For destination types defined as classes and regions they are the highest in Warsaw, followed by Poznań and Gdańsk, with Wrocław having the lowest number. In multicast routing directed at individual nodes, the number is the same for all cities.

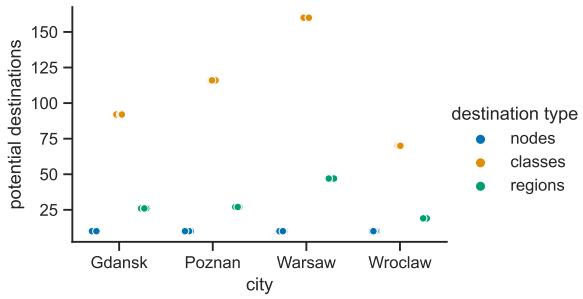


Figure 17. Potential multicast destinations.

6.1. Multicast Nodes

Due to the highly multidimensional structure of the data related to the modeled multicast graphs, different views on it have to be presented. First, the trends in the number of nodes will be analyzed in relation to Figure 18. In this figure, the numbers are compared for graphs that are the multicast structures directed at reaching the nodes within the given destination regions. This view is chosen because it presents the scenario which is between the one with the smallest number of destinations, i.e., individual-node-directed multicast, and the one with the largest number, i.e., class-directed multicast. Moreover, there exists a relationship between the number of destination nodes and the parameters of the resulting graphs. This will be further discussed and the reasoning will be proved valid.

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

Figure 18 indicates that after the initial phase of increase in the number of nodes that follows the increase in duration and radio range, the number approaches or reaches the limit 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 same characteristics. To investigate them in more detail

the networks of 30 min duration will be studied, since this period is the middle one of the periods of interest and ensures that the conclusions drawn can be accordingly extended to both shorter and longer durations.

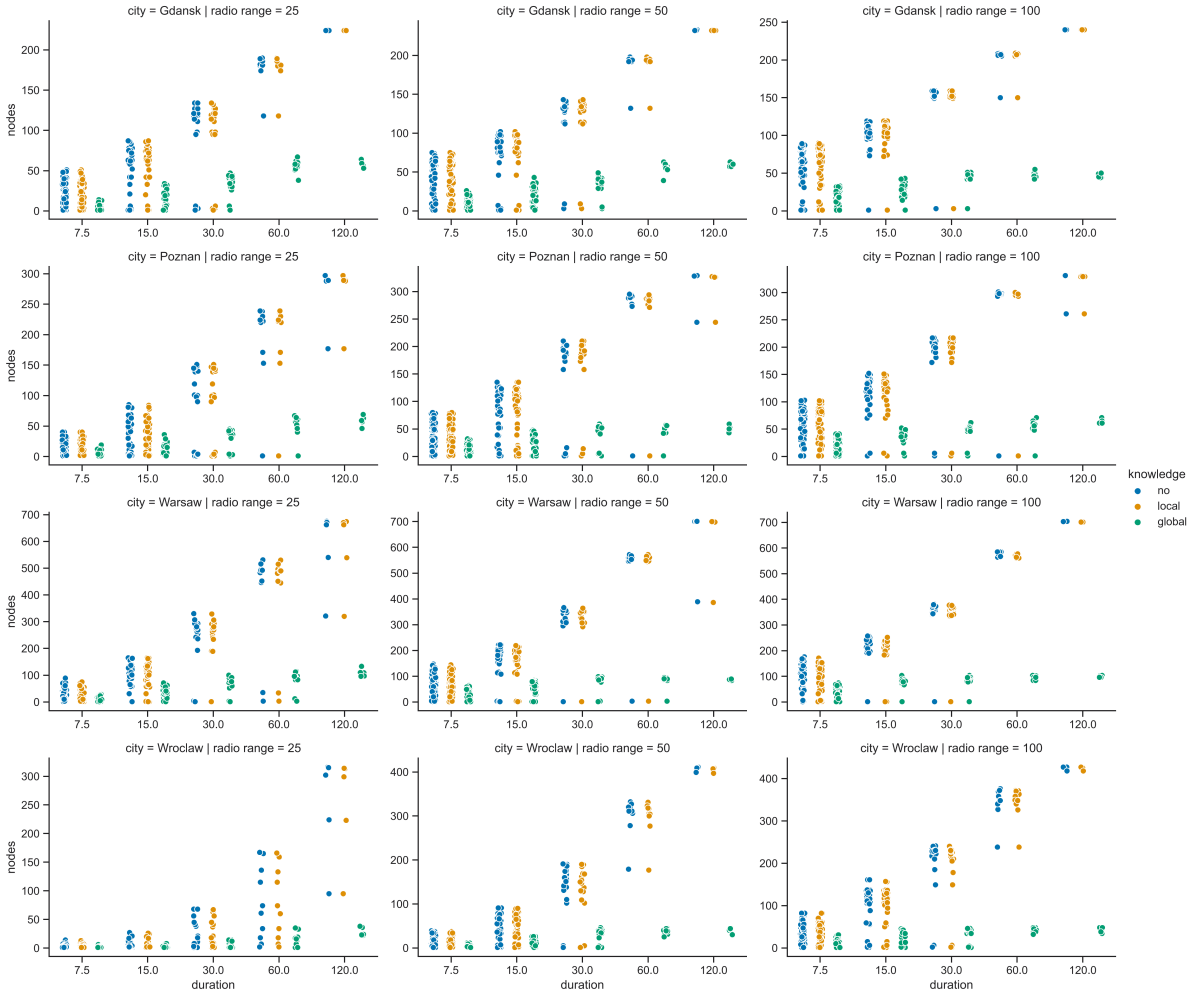


Figure 18. All nodes in multicast graphs with destination regions.

The overall number of nodes in the multicast structures of 30 min shown in Figure 19 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 should 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 are connected to the tree. Most of the trees are composed of significantly more nodes. It is no surprise that when global knowledge of the network topology is used, the number of nodes is 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 19 exhibit varying vertical spreads. For instance, in the classes destination type with 25 m of radio range, the metrics for Poznań are the most spread (distributed) while in Wrocław they are the most condensed. They grow and become more condensed with the increase in radio range. This conclusion follows the observation that network connectivity increases with increasing range, and hence, more nodes can be reached and become members of the tree. This proves that even at lower radio ranges multicast trees with a complex structure and space–time reach can be constructed. This is

visible in Figure 20, which compares the percentages of potential multicast destinations that were connected to the trees. When the destination type changes and there are fewer potential destinations to be reached, the overall numbers of nodes in the trees decrease proportionally. Although, the efficacy of the tree construction, i.e., the number of reached destinations, remains high in each variant and city. With the lowest radio range it mostly stays between around 50 and 90 percent in Gdańsk, Poznań, and Warsaw. It is visibly lower in Wrocław, but when the range increases it moves towards much higher values, although still falling behind the other cities. In the other cities, the efficacy grows and the algorithms reach 100 percent when the modeled space–time networks are interconnected well enough.

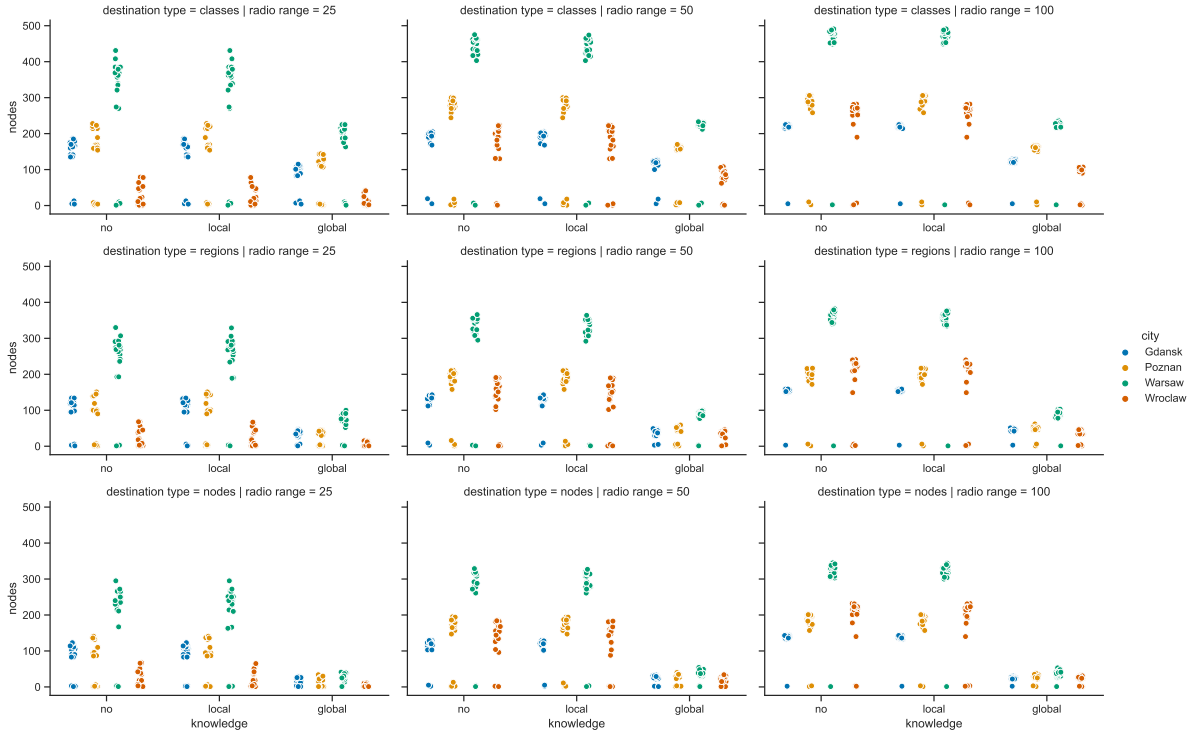


Figure 19. All nodes in multicast graphs of 30 min.

As compared in Figure 21, with global knowledge the average number of nodes can range from 60 percent of the nodes in the case of no knowledge, to less than 20 percent, depending on the destination type and radio range. This is related to the flooding-like nature of no- and local knowledge variants, which produce branches not leading to the destinations. It should be pointed out that local-knowledge-based routing tends to produce multicast structures composed of numbers of nodes similar to the 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 22 are correlated with those of the overall numbers of nodes in Figure 19 and span from a few nodes to hundreds of relays. The mobile relay node ratios, when the baseline is the number of mobile relays in graphs constructed with no knowledge, depicted in Figure 23, resemble the distributions of all nodes presented in Figure 21. The percentages of mobile relays in the cities of interest seem to decrease in most of the configurations when the radio range increases—starting with Wrocław, then Warsaw, Poznań, and Gdańsk. The global-knowledge-based approach requires on average only around 10 to 60 percent of mobile relays needed by the 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 24 as the points at 100 percent. Otherwise, when there are more nodes in the tree, the percentages of mobile relays are lower, as compared to the number of all nodes in the graph. It is visible that Wrocław, despite having the lowest absolute number 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 this results in the lowest efficacy, as expressed in Figure 20.

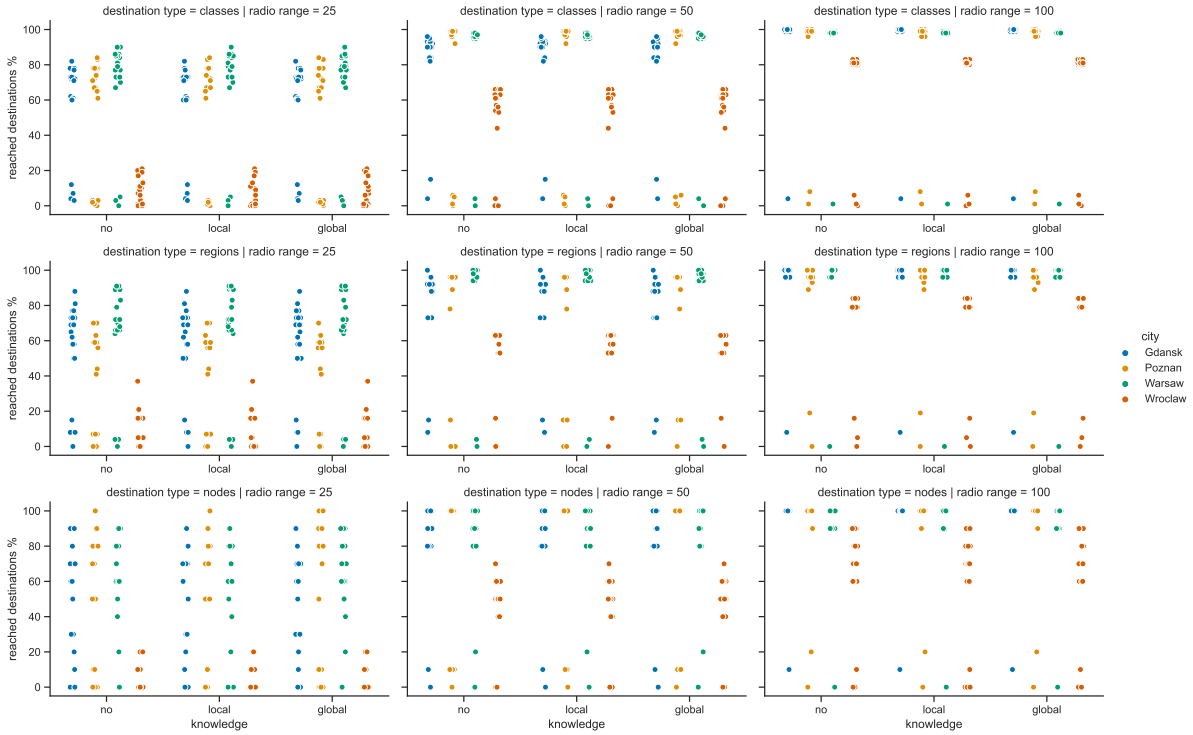


Figure 20. Reached- to-potential destinations ratio in multicast graphs of 30 min.

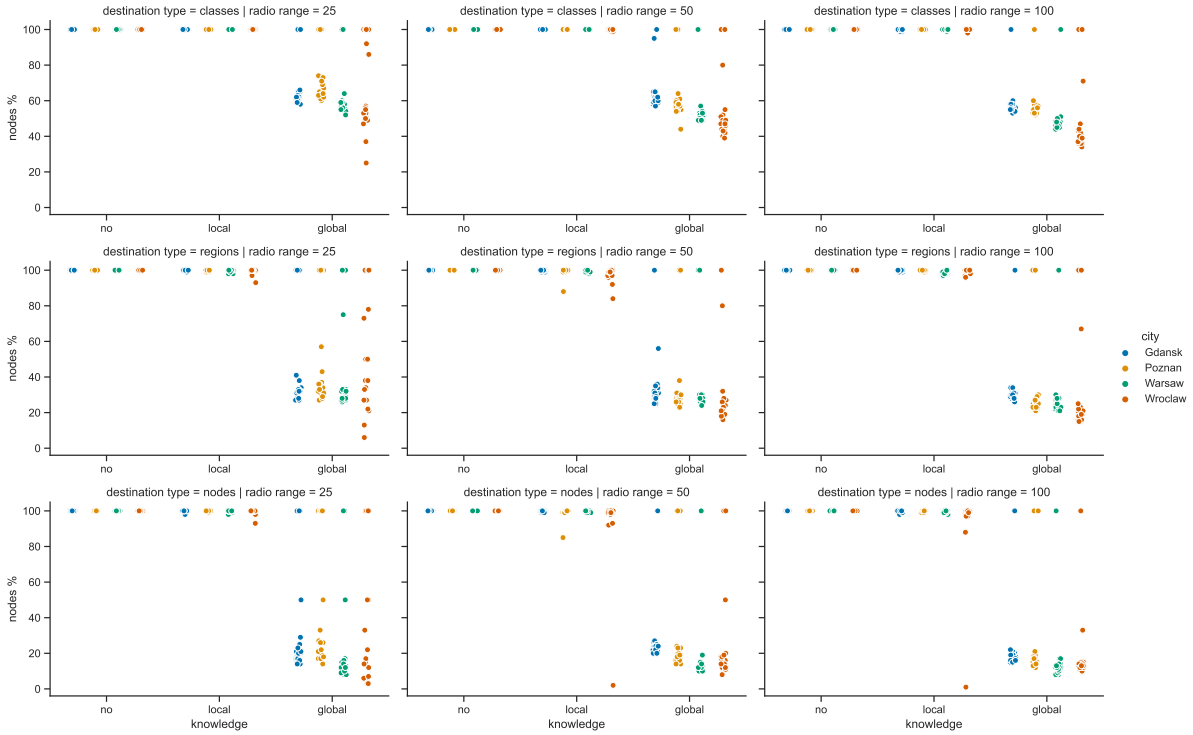


Figure 21. Nodes ratio to no-knowledge in 30 min multicast graphs.

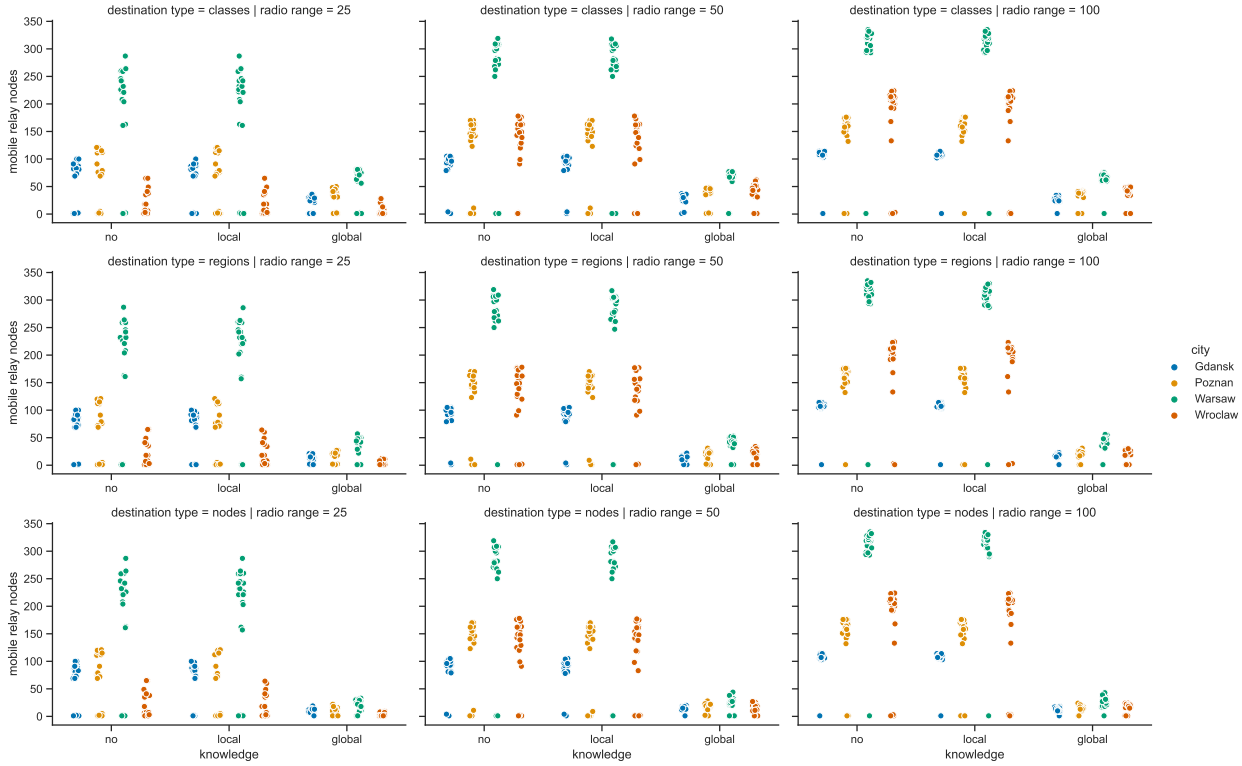


Figure 22. Mobile relay nodes number in 30 min multicast graphs.

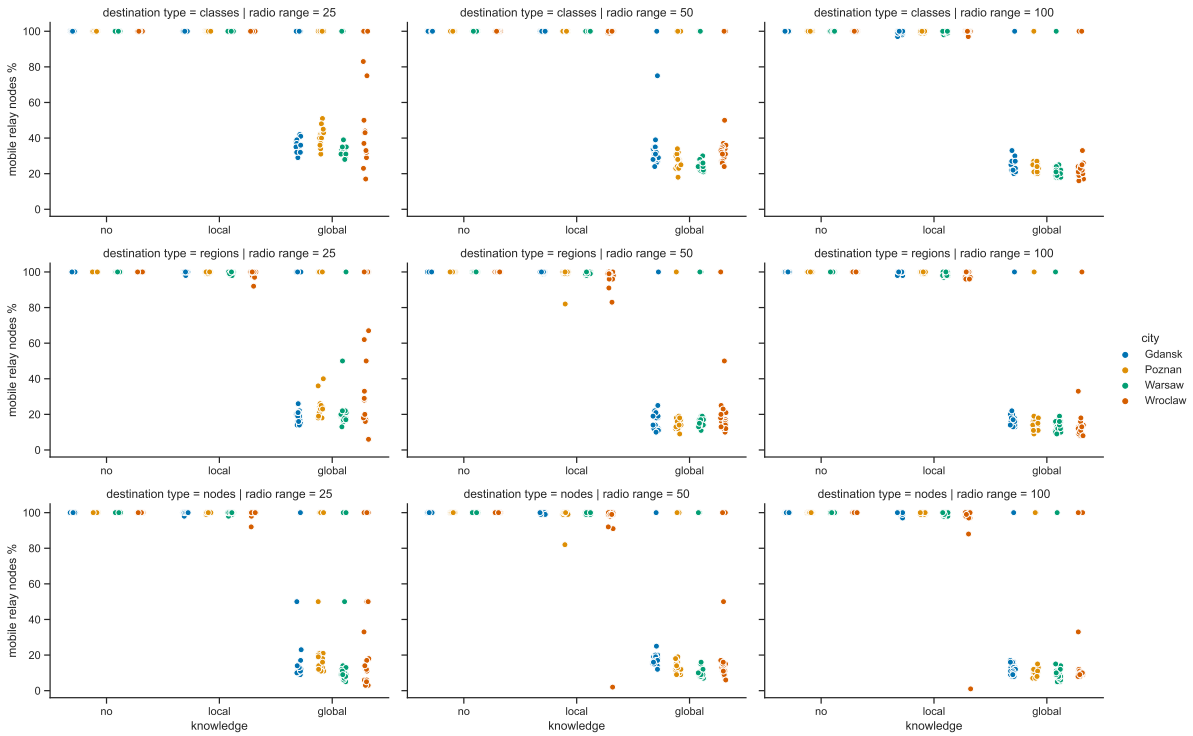


Figure 23. Mobile relay nodes ratio to no-knowledge in multicast graphs.

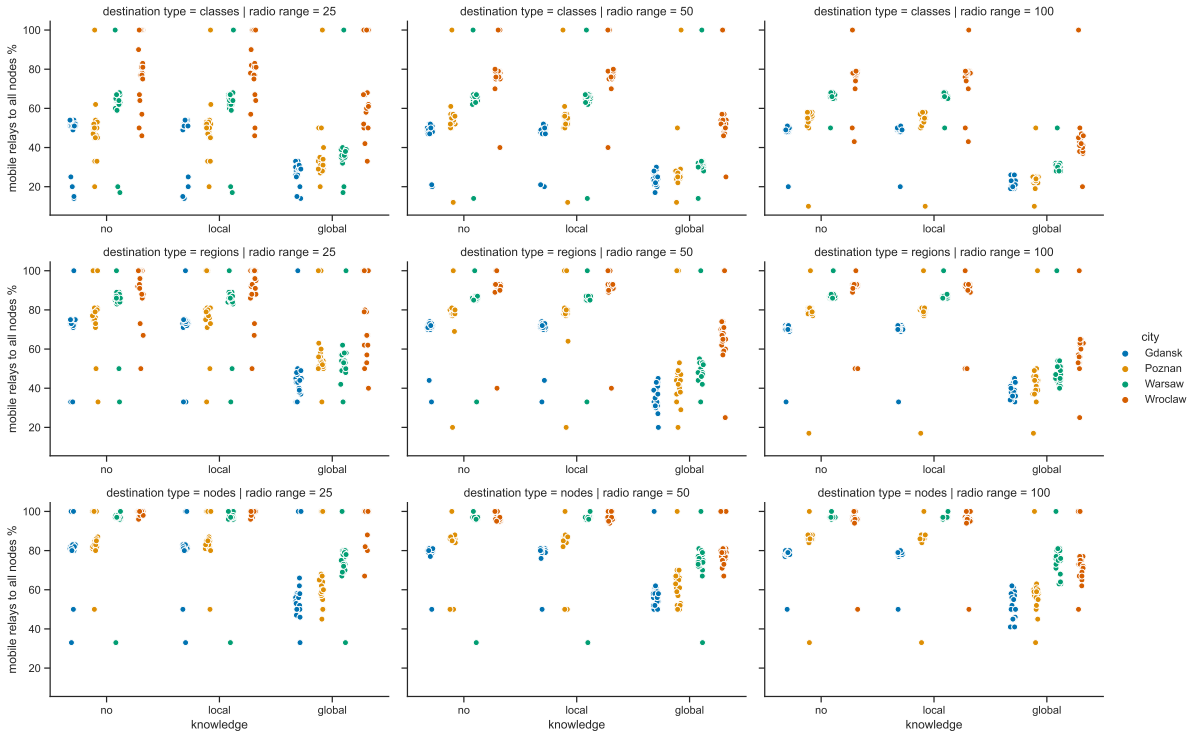


Figure 24. Mobile relay nodes to all nodes ratio in 30 min multicast graphs.

6.2. Multicast Costs

The space costs compared in Figure 25 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 the case of global-knowledge-based routing, with the ones in Warsaw being the highest among the cities.

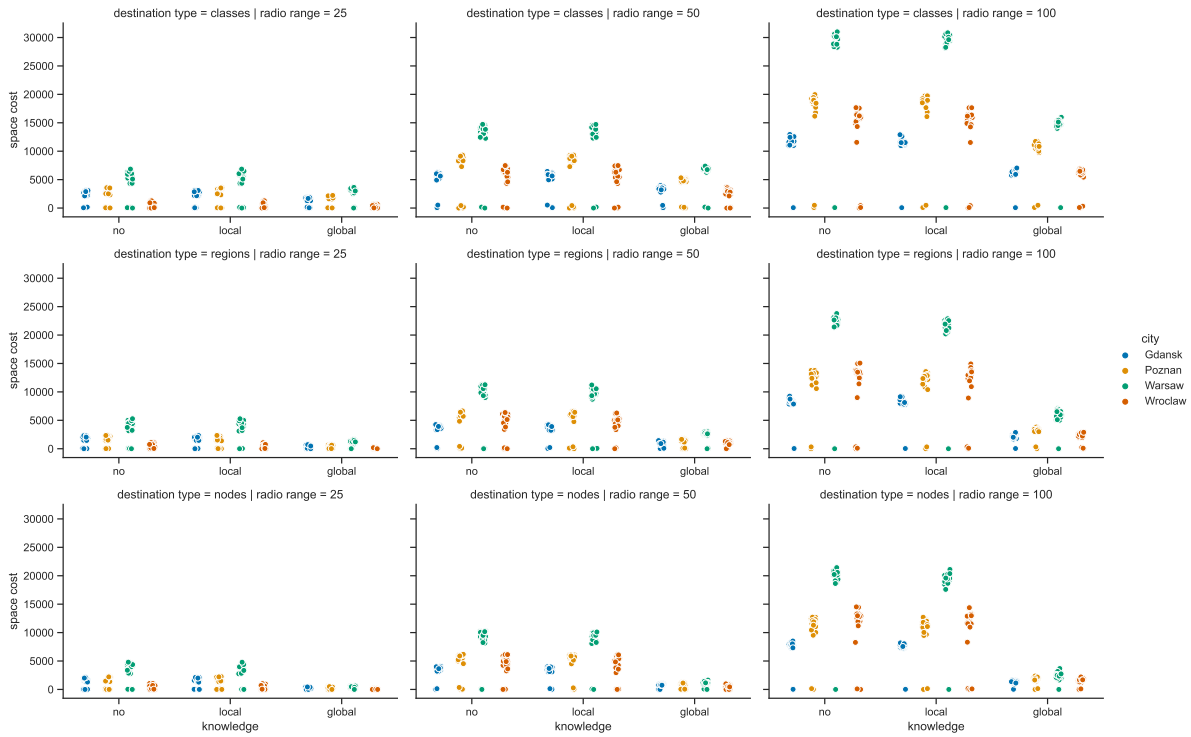


Figure 25. Space cost of 30 min multicast graphs.

The time costs visible in Figure 26 show a correlation with the number of mobile relay nodes in Figure 22. The distributions here though are, in some cases, more vertically spread and shifted, due to the influence of other attributes of the infrastructure.

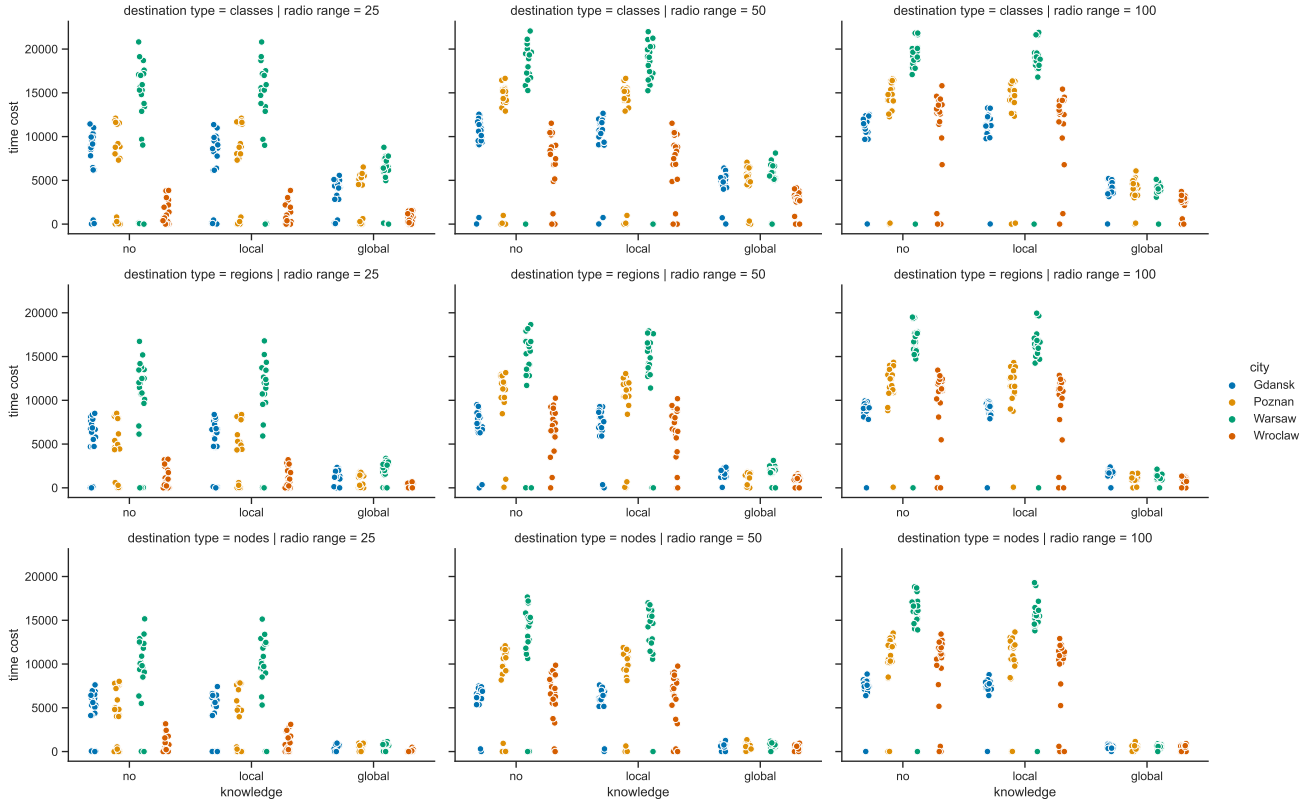


Figure 26. Time cost of 30 min multicast graphs.

The time spans in Figure 27 are the same for the no- and local-knowledge-based routing methods since they both tend to reach as far in time as possible. This causes the overhead over the global-knowledge-related routing, which grows when the radio range increases and the number of destination nodes decreases. It can also be observed in Figure 28 how much less resource consuming is the global knowledge approach. There, the sum of all the maximum time spans of each node is considered to be the overall time span cost of the structure. This metric changes the most in Wrocław when the radio range increases. This means that when the range increases, more nodes are connected to the relays, which increases the time span cost.

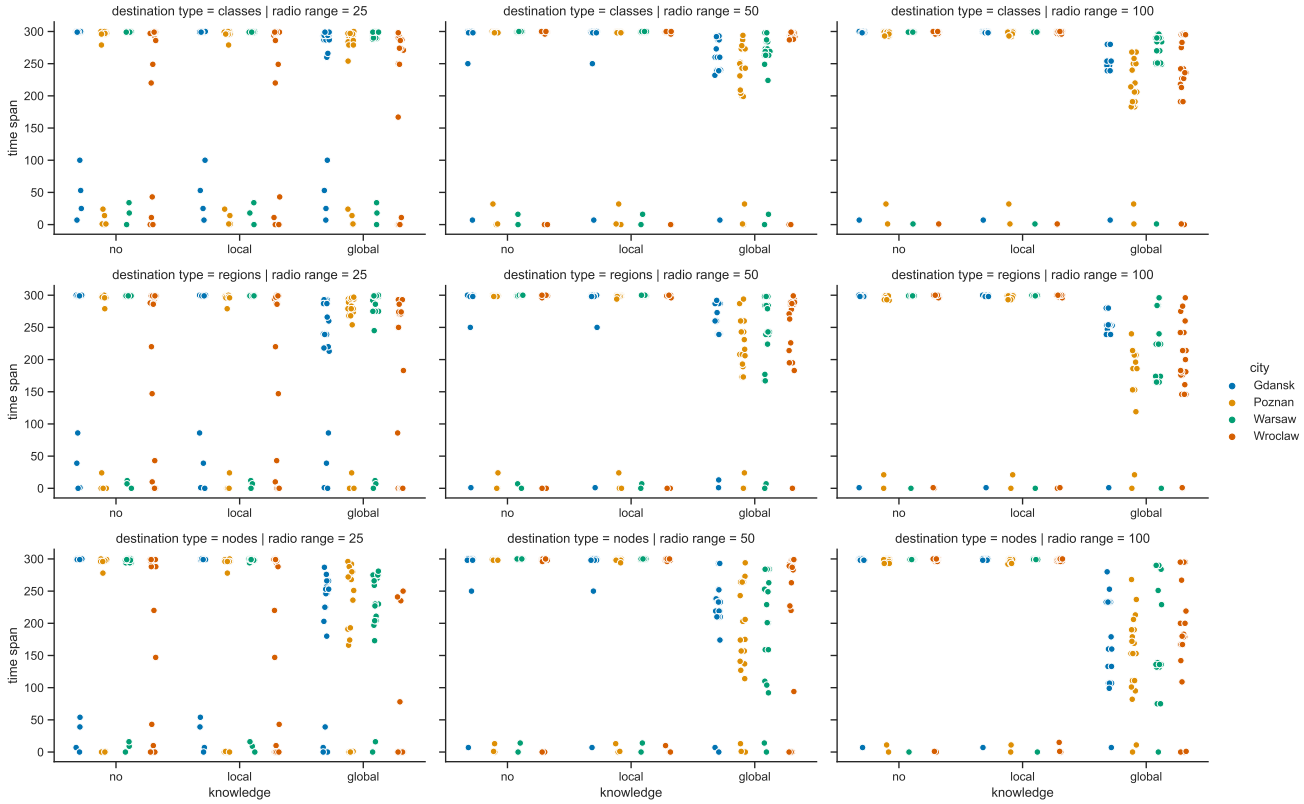


Figure 27. Time spans of 30 min multicast graphs.

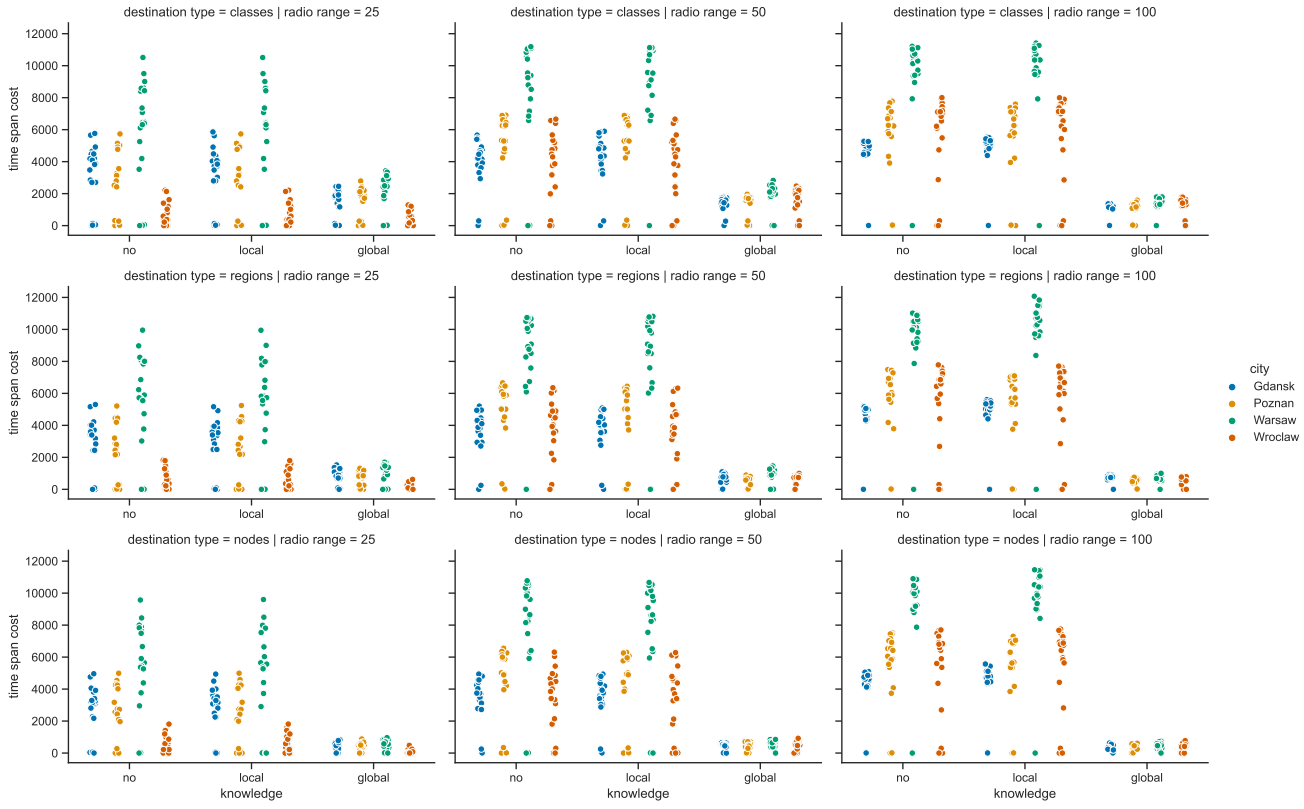


Figure 28. Time span cost of 30 min multicast graphs.

6.3. Efficacy and Efficiency of Multicast Algorithms

The relative comparisons in Figure 29 may seem like the ones related to nodes in Figure 23. However, 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. The space cost is the lowest with global knowledge in general, as could be expected. When it is zero percent, it means that no actual multicast tree was 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.

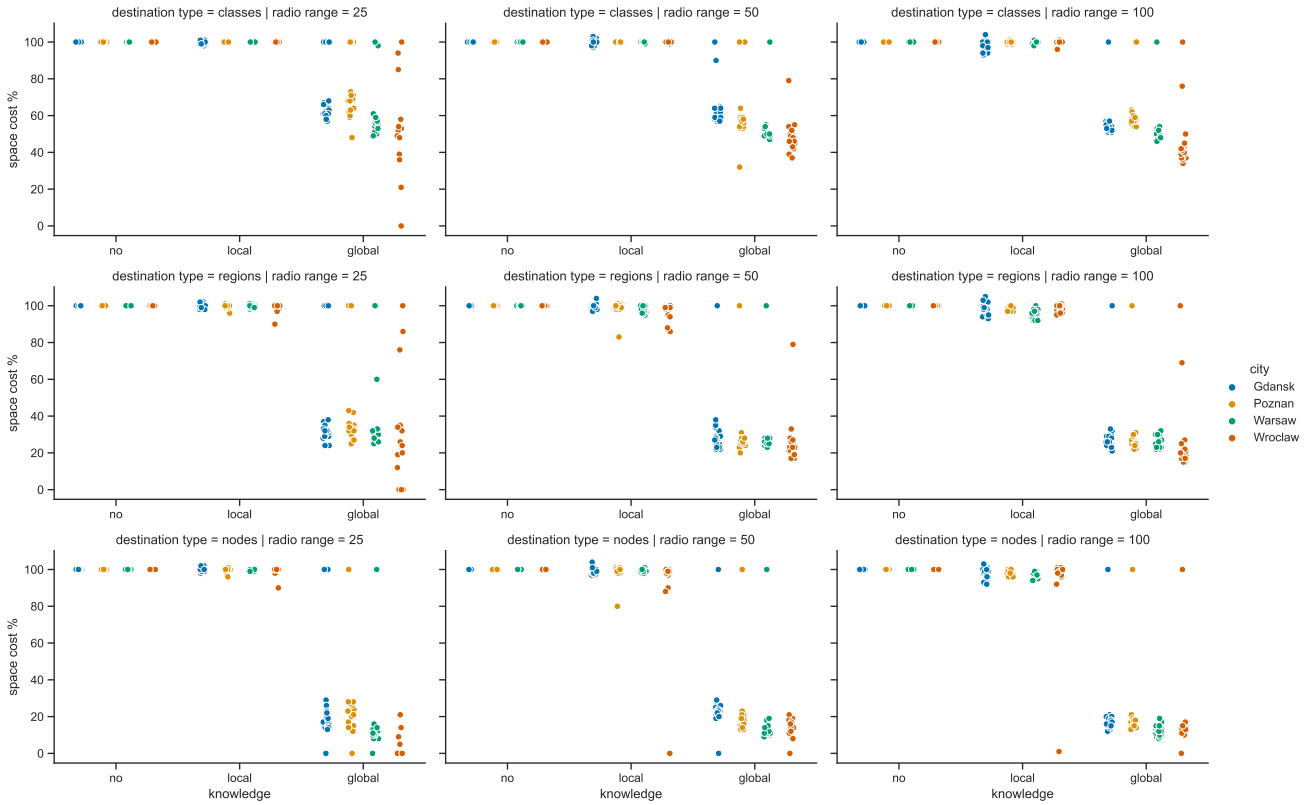


Figure 29. Space cost ratio to no-knowledge in 30 min multicast graphs.

Although the influence of the radio range on the time costs in Figure 30 is much smaller than on space costs (compare with Figure 25), the distributions of the relative time cost ratios mirror the ones of the relative space costs in Figure 29. The groups are shifted towards lower percentages. This means that global knowledge results in multicast structures that are even more efficient in the time domain than they are in the space domain. This is further visible in Figure 31 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 both use significantly less resources in general and require them to be utilized in shorter time periods, on average.

Based on the already-analyzed parameters, it could be predicted that the time span cost ratios will be the lowest for global knowledge, as presented in Figure 32. When the radio range increases they tend to become lower, as well as when the number 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.

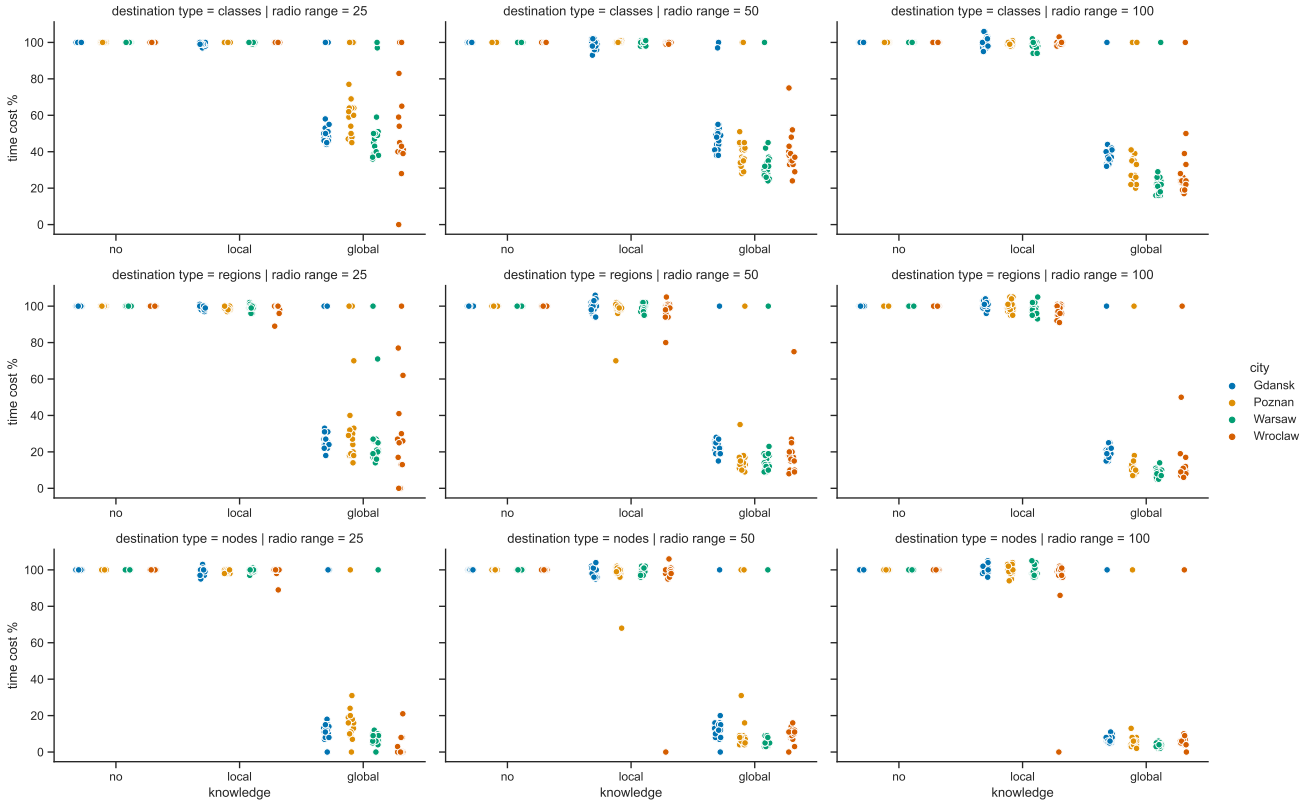


Figure 30. Time cost ratio to no-knowledge in 30 min multicast graphs.

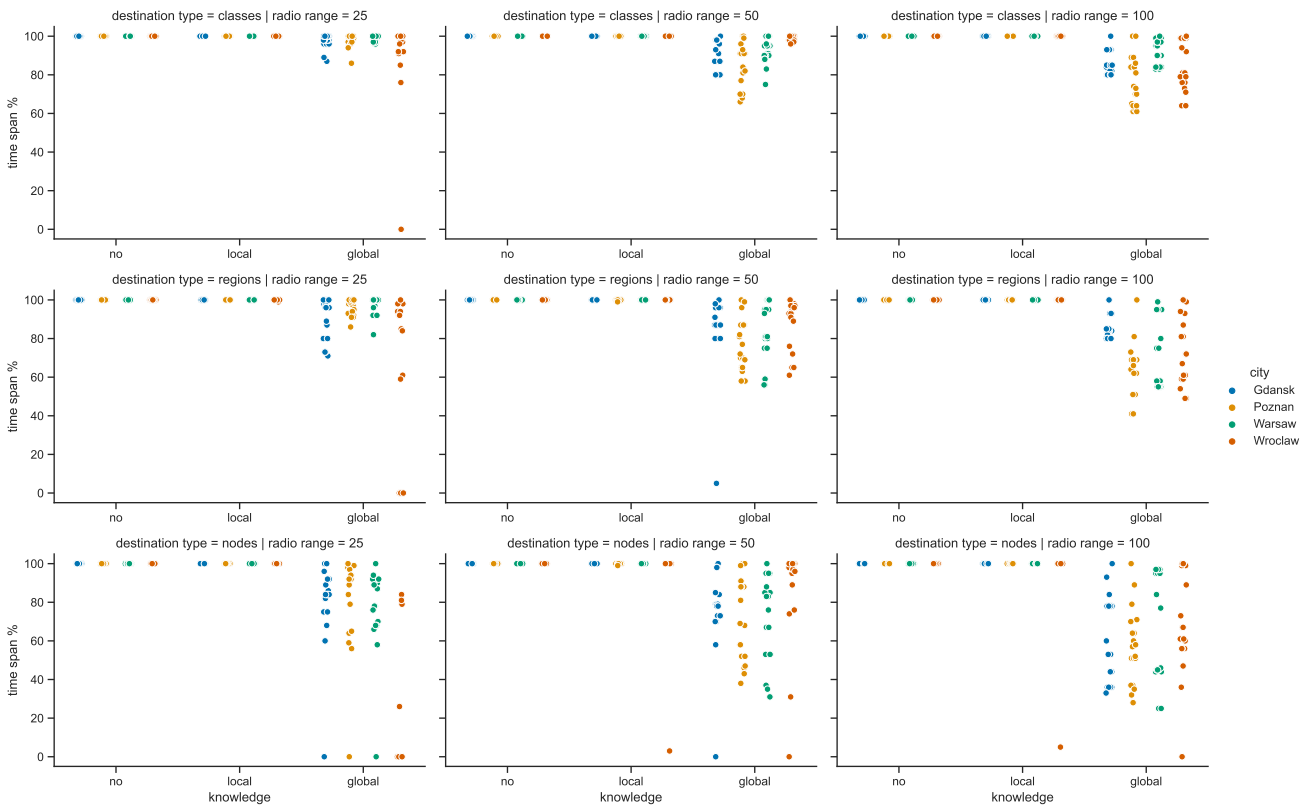


Figure 31. Time span ratio to no-knowledge in 30 min multicast graphs.

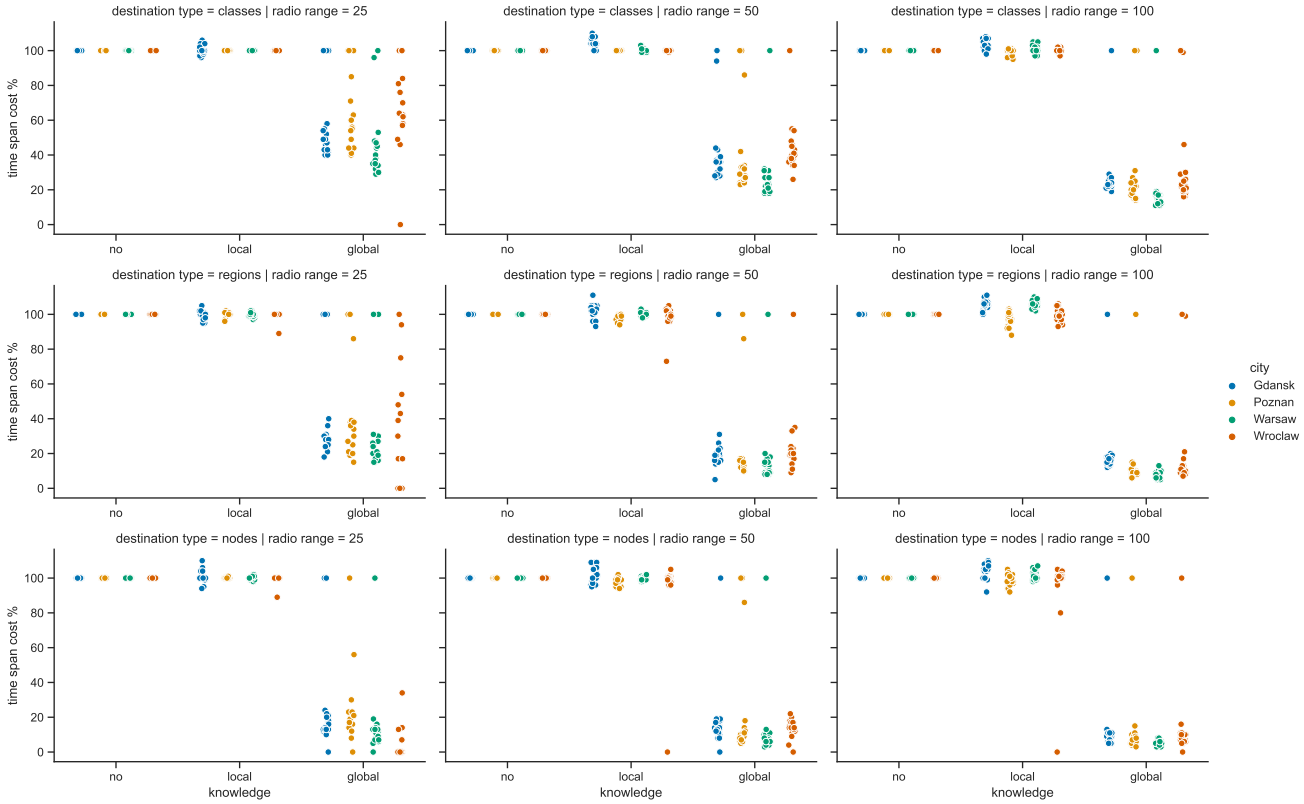


Figure 32. Time span cost ratio to no-knowledge in 30 min multicast graphs.

Core Multicast Structure

The relative distributions of core node numbers in Figure 33 demonstrate the influence of the area the algorithms operate in, and even more, of the number of destination nodes and the 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 the 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 the core space cost ratios in Figure 34 and core time cost ratios in Figure 35 closely resemble the one of the core node ratios in Figure 33, they are presented to show that they are highly correlated. The core time cost ratios appear to be somewhat lower, on average, than the core space cost ratios. This is similar to the time cost ratios in Figure 30, where the more centrally located groups of values for global knowledge represent, on average, lower percentages than their counterparts related to space cost ratios.

The time span of the core multicast structure (see Figure 36) in the 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 the core time span cost ratios in Figure 37 are most similar to the ones of the core time cost ratios in Figure 35, 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 in radio range, the average core time cost ratio drops. It decreases even more when the number of destinations goes down when changing the destination type from classes through regions to individual nodes. This means that when the radio range is the highest and the lowest number of destinations are

to be reached, the largest number of furthest-reaching time edges are redundant since they do not lead to the destinations.

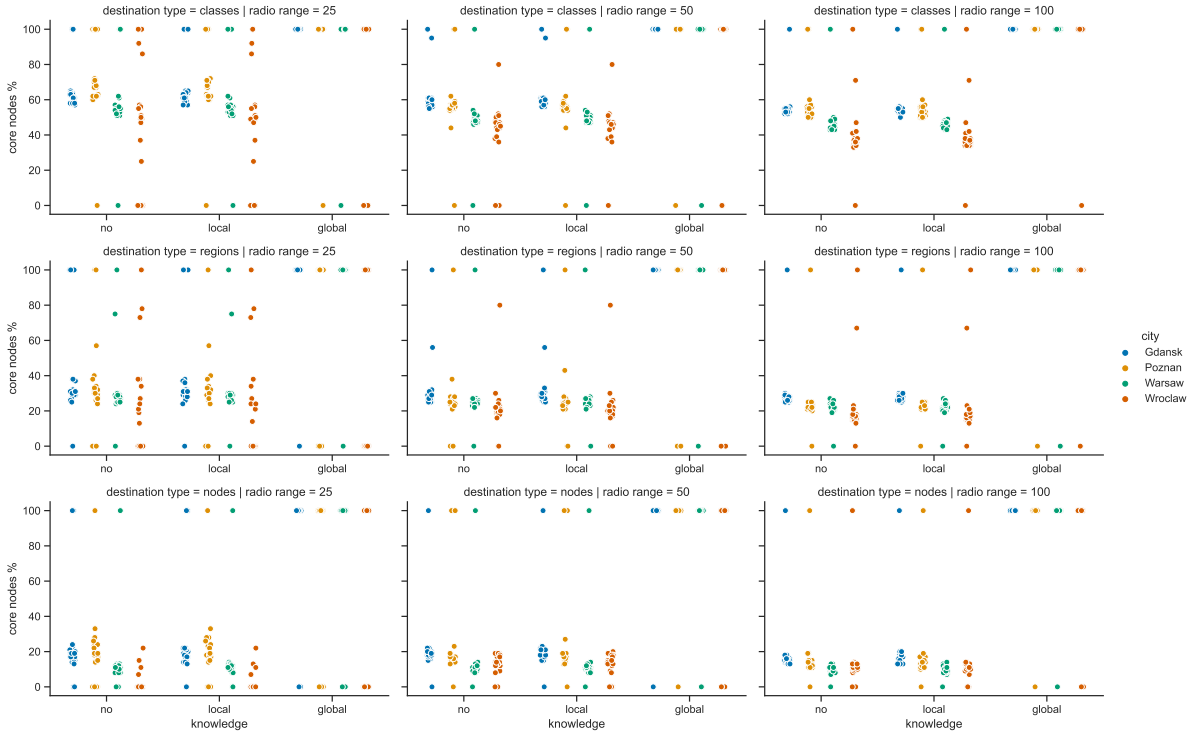


Figure 33. Core nodes ratio in 30 min multicast graphs.

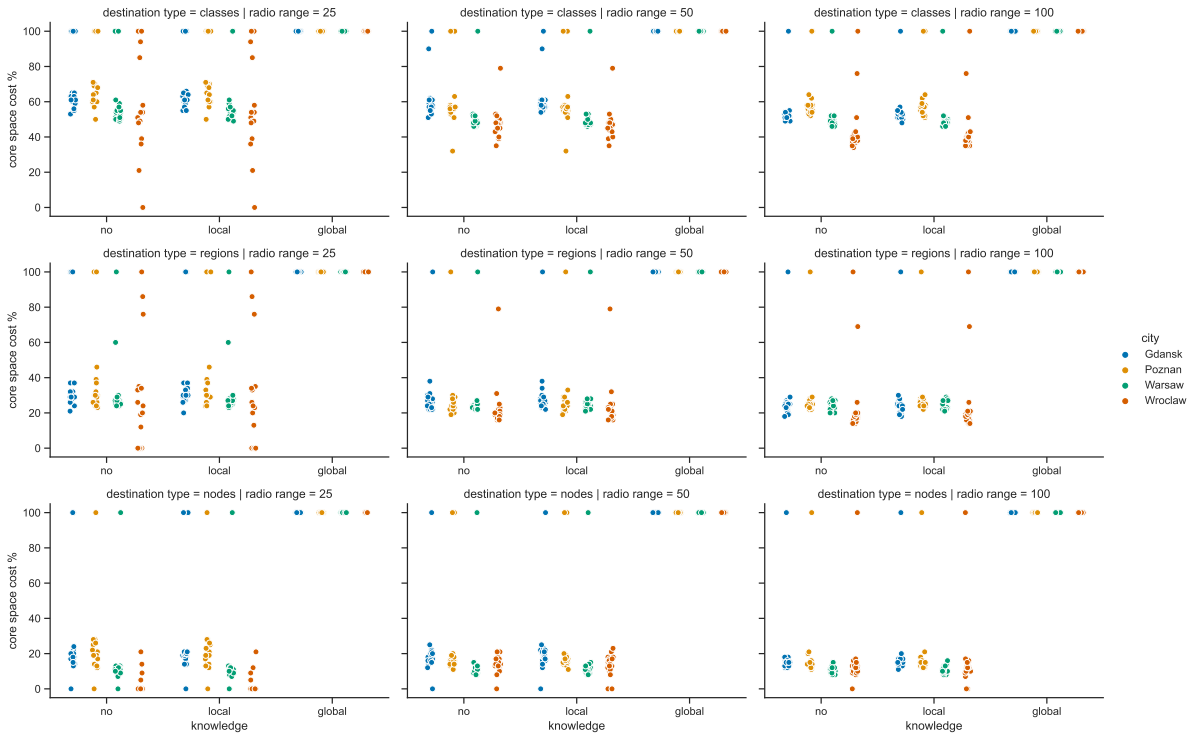


Figure 34. Core space cost ratio in 30 min multicast graphs.

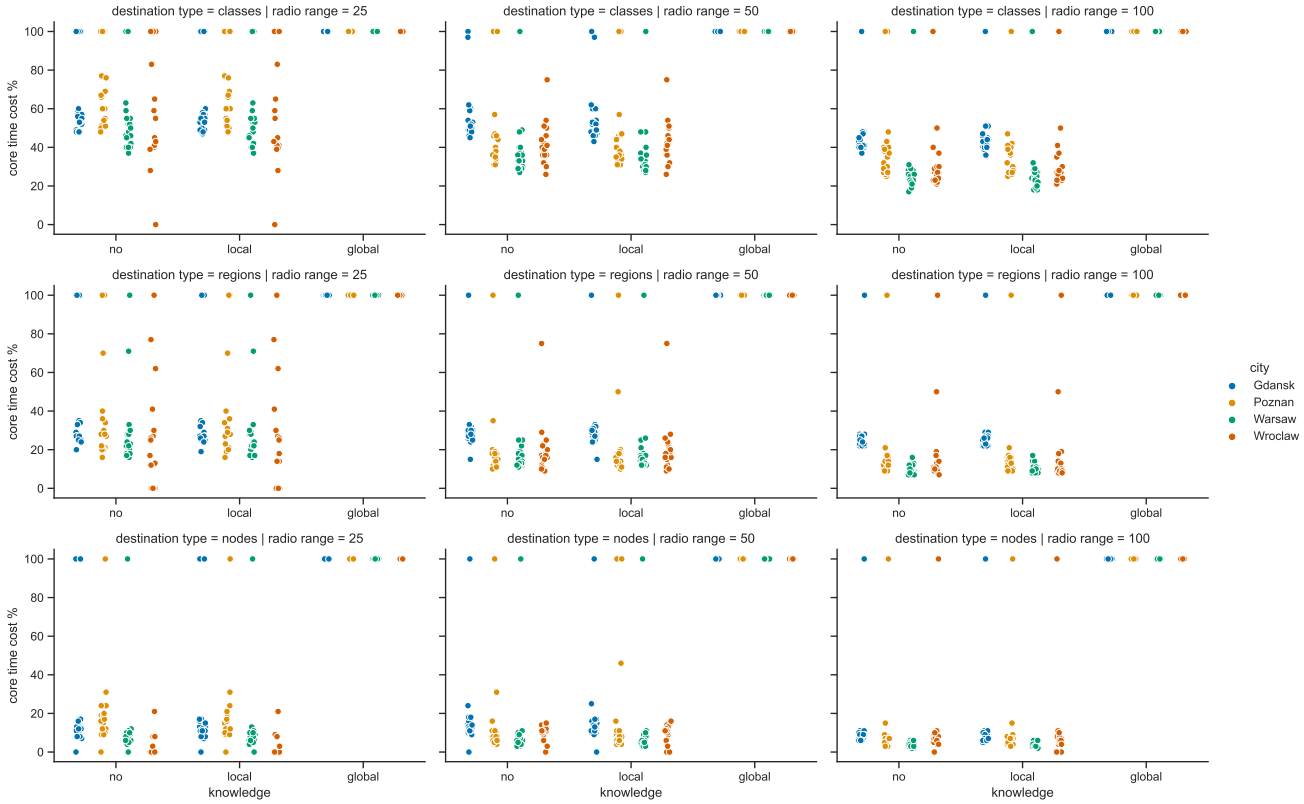


Figure 35. Core time cost ratio in 30 min multicast graphs.

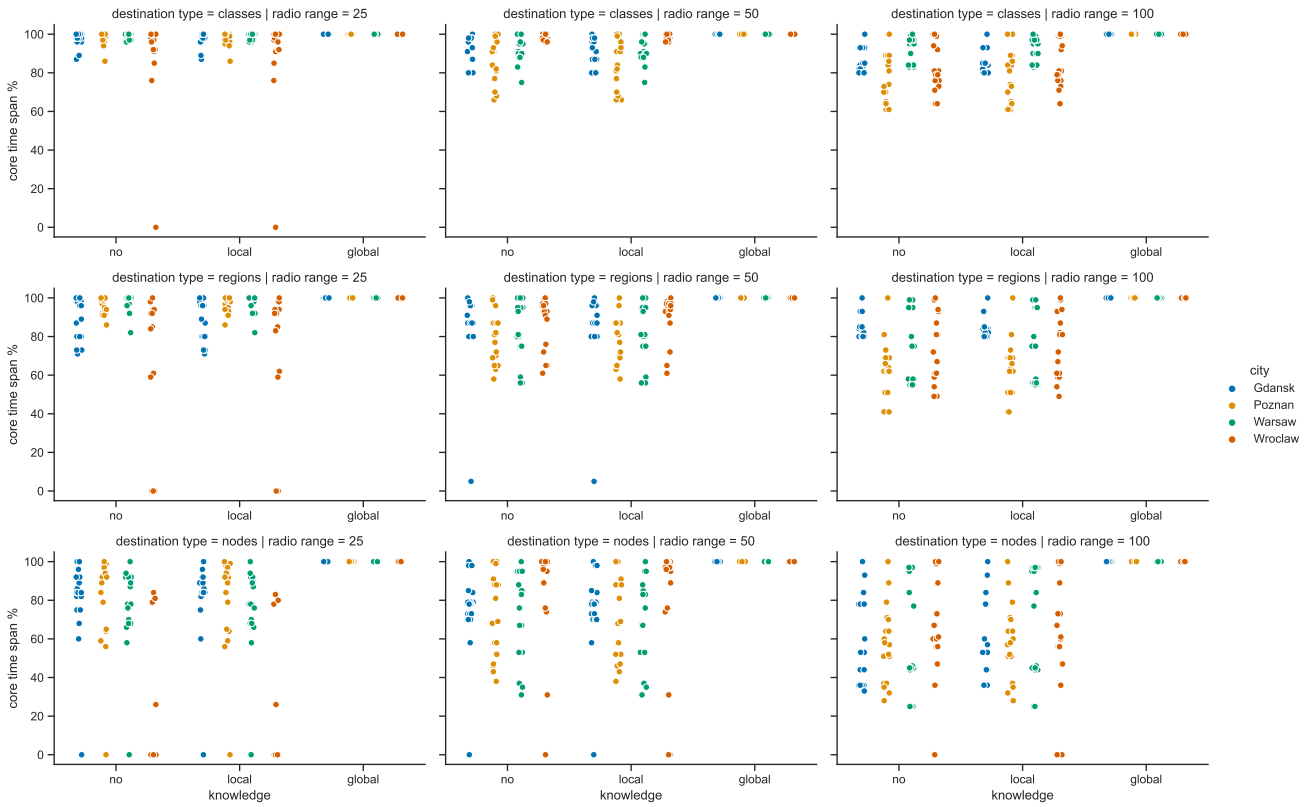


Figure 36. Core time span ratio in 30 min multicast graphs.

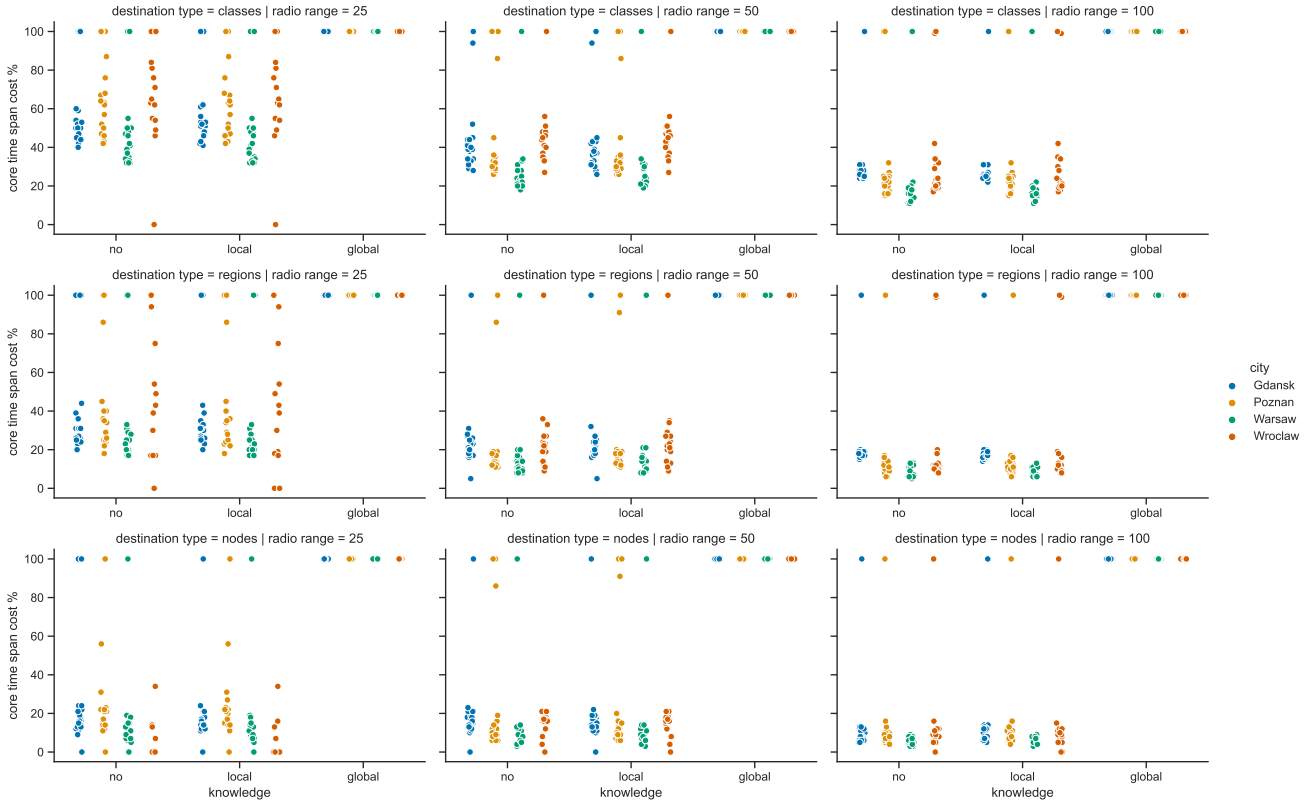


Figure 37. Core time span cost ratio in 30 min multicast graphs.

6.4. 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 23 and 24, to construct multicast structures with lower numbers of mobile relay nodes involved.

The space costs diagrams in Figure 38 show an almost linear increase trend in relation to the number of mobile relay nodes included in the multicast graphs. The numbers also increase when the radio range and number of destination nodes increase, which is accompanied by more visible separate grouping of the space costs of different cities. Warsaw is the city with 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 lowest for the lowest radio range and the lowest number of mobile relays. When the radio range is the highest and the lowest number of destination nodes are to be reached, the cost and number of the involved mobile relays tend to be second largest after Warsaw. The space costs of the Gdańsk graphs are the lowest and the most grouped when the radio range is the largest.

The time costs also exhibit a linearly increasing trend related to the number of mobile relay nodes, as visible in Figure 39. Here, the groups of values related to each city are more easily noticeable. This shows that the increase in radio range causes the time costs to be less dispersed for a given area and infrastructure. The distributions of the time span costs in Figure 40 are similar to the 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 is the absolute values, which are almost two times higher in the case of time cost.

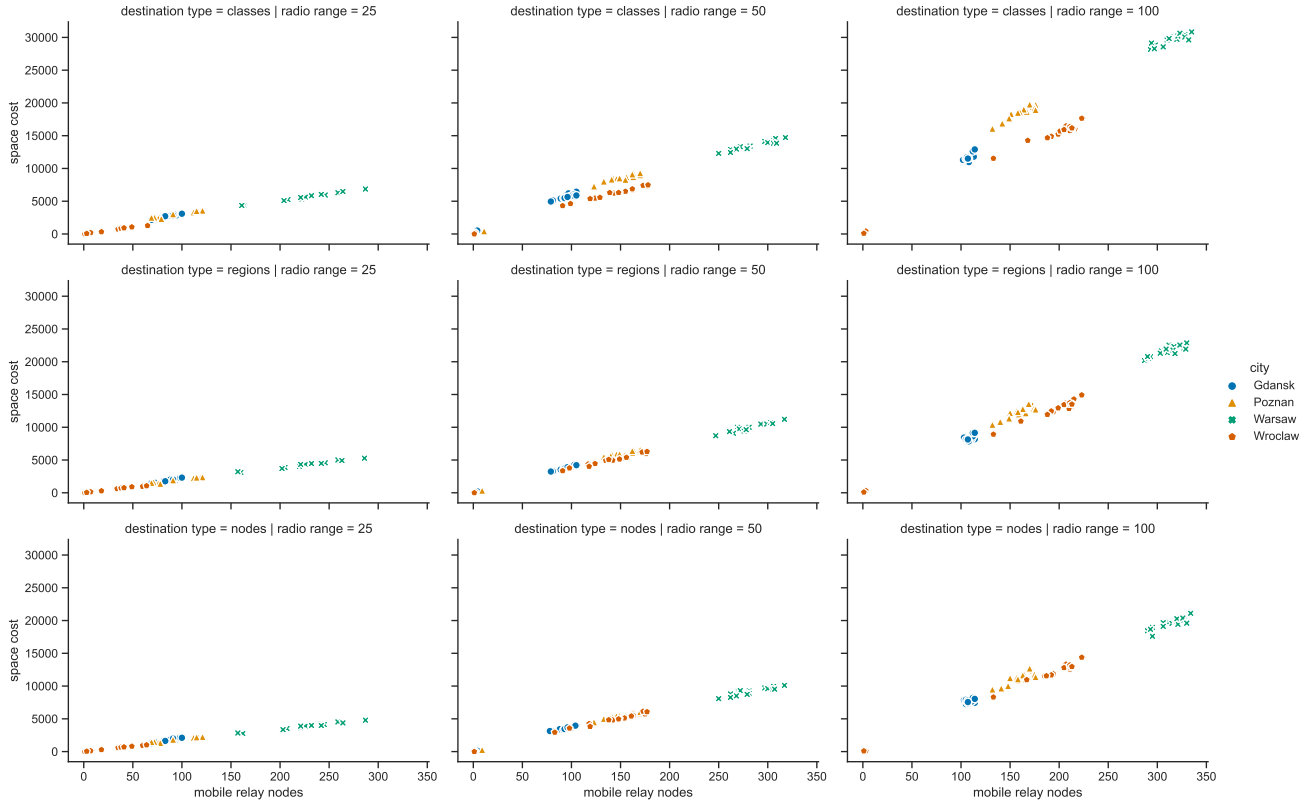


Figure 38. Mobile relays' influence on space cost of 30 min graphs with local knowledge.

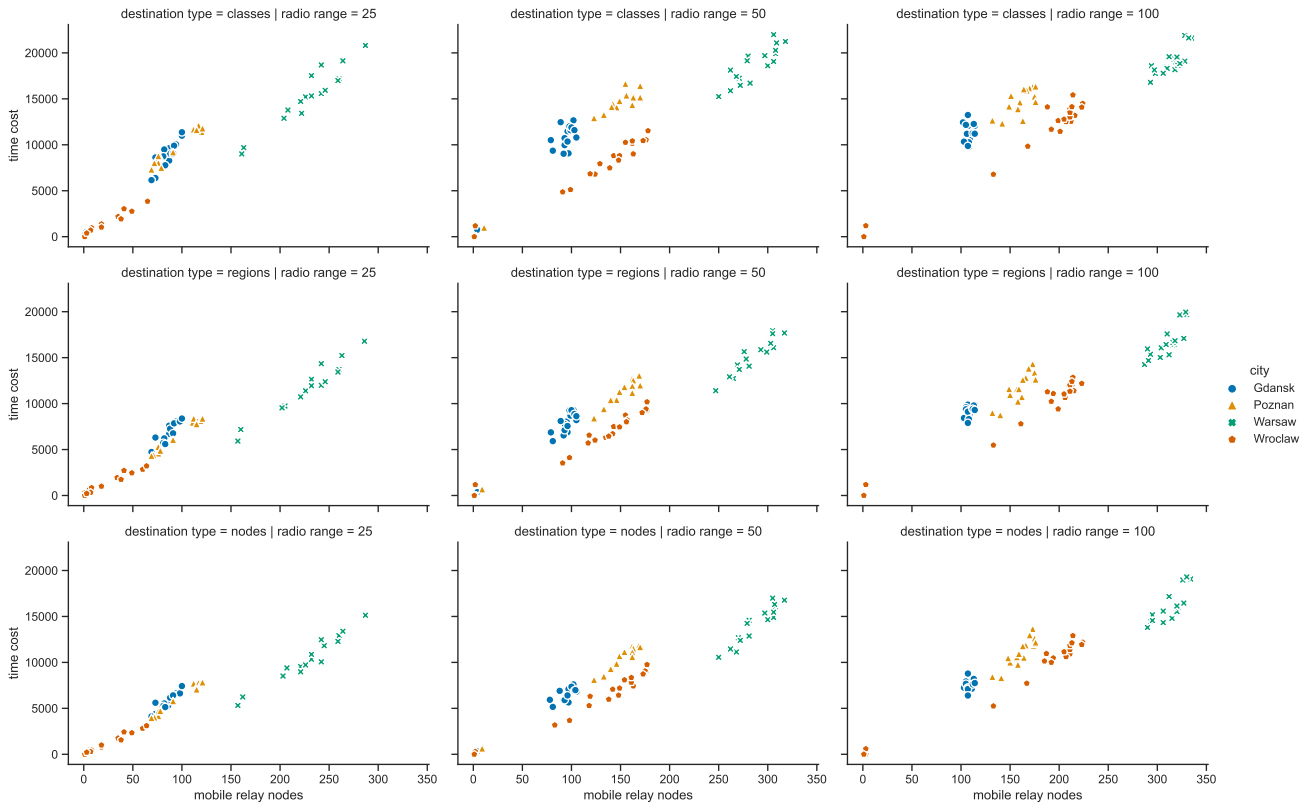


Figure 39. Mobile relays' influence on time cost of 30 min graphs with local knowledge.

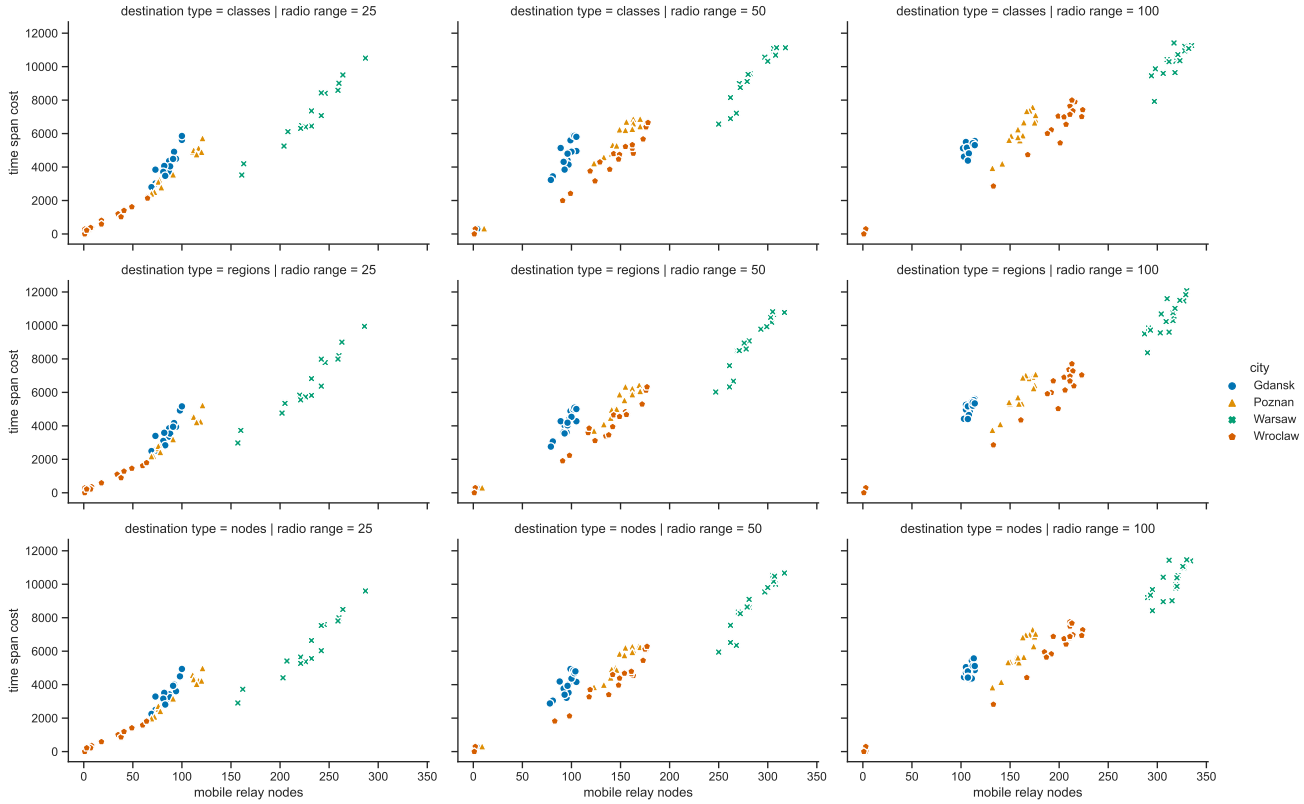
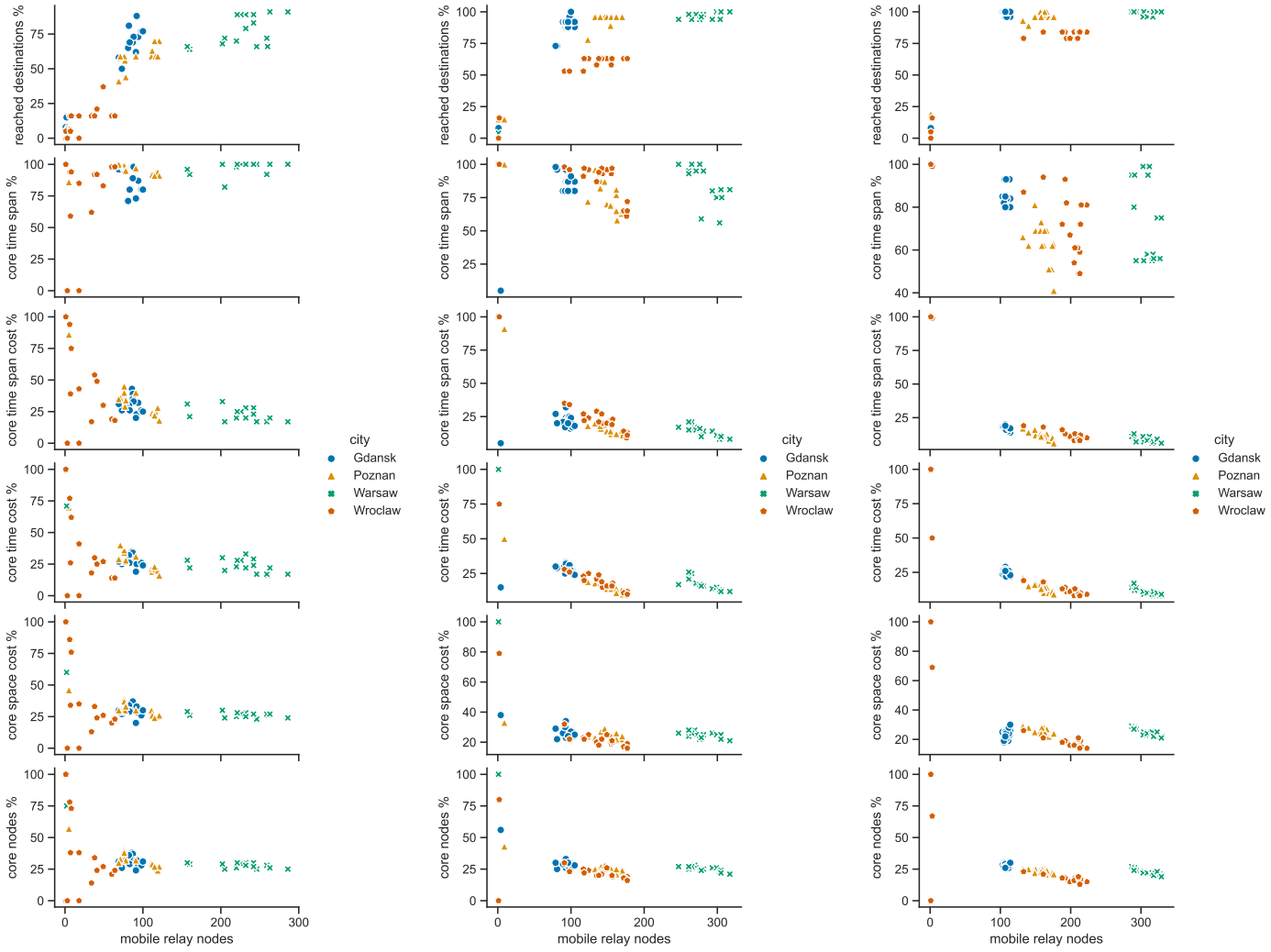


Figure 40. Mobile relays' influence on time span cost of 30 min graphs with local knowledge.

The relationships between mobile relay nodes and the ratios related to the core part of the multicast structures with destination regions and local knowledge are presented in Figure 41 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 the multicast graphs. For larger numbers the ratios are more grouped in the case of the 50 and 100 m radio ranges and do not exceed 50 percent. More mobile relays are included in the structures as well. The cost ratios decrease with the increase in mobile relay nodes. The core time span extends 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 relays grows from zero to one hundred nodes with a 25 m radio range. Then, it reaches around 75 percent regardless of further increases in the number of mobile relays in Gdańsk, Poznań, and Warsaw. When the 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.

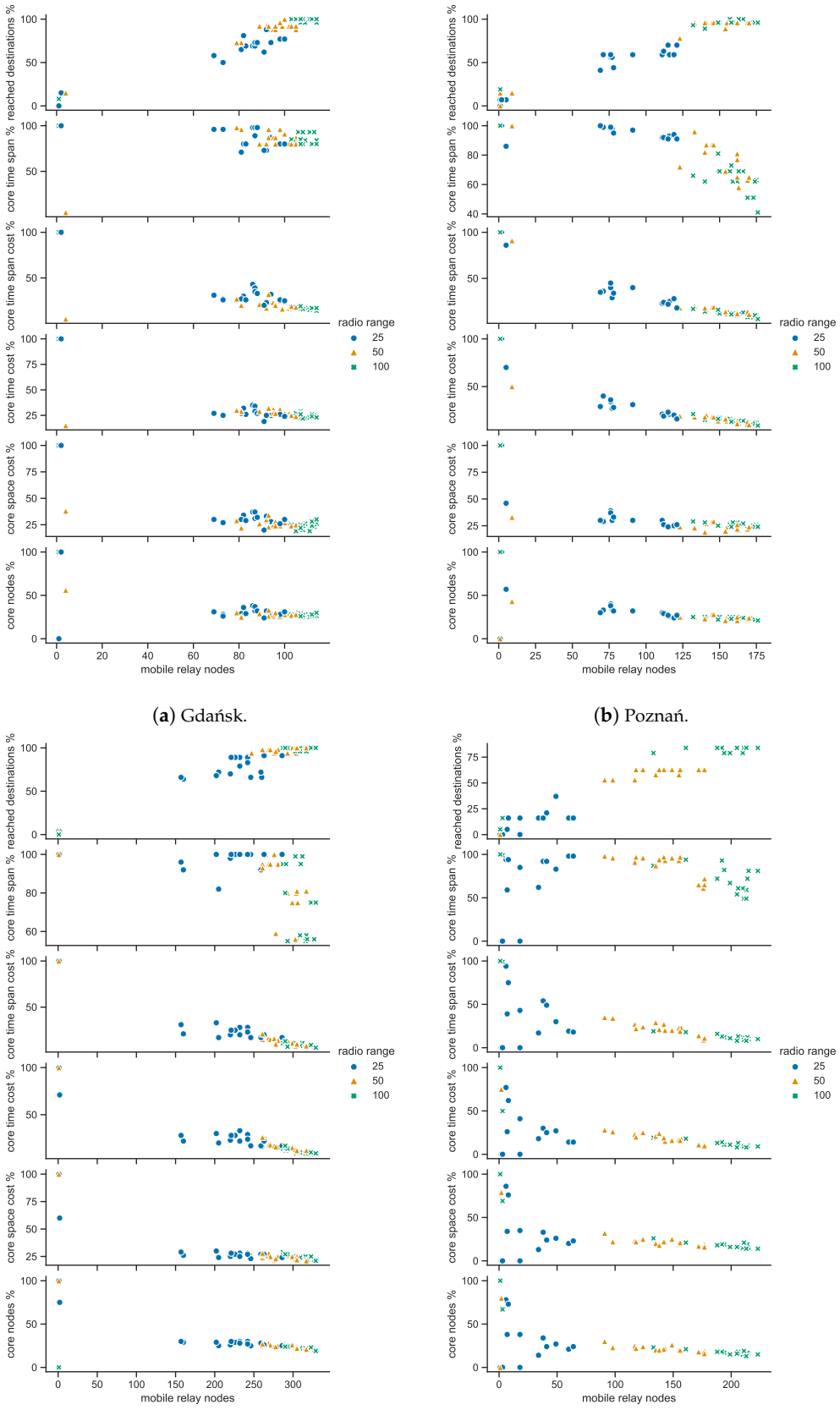
To highlight the influence of mobile relays in different city areas, the data presented in Figure 41 are shown in Figure 42 grouped in city-related sets. In Gdańsk, the number of mobile relays slightly exceeds 100 nodes. The distribution of ratios seems only a little more compressed than in Warsaw, but in fact, it is much more compressed, since in Warsaw the number of nodes spans over a 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 start to follow a decreasing linear trend, becoming denser with an increase in the number of mobile relays. It is visible that an increase in the radio range in all cities causes an increase in the mobile relay nodes included into the core multicast structure. Moreover, the increase in the radio range from 50 to 100 m brings only a few percent improvement in Gdańsk, Poznań, and Warsaw. The same increase in Wrocław results in a much higher growth in the percentage of reached destinations, starting at around 60 and reaching 80 percent. The core time span ratio appears to be more

related to the radio range than to the number of mobile relays, decreasing slightly in some cases with the increase in the number of nodes.



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

Figure 41. Mobile relay and radio range influence on multicast core of 30 min graphs with destination regions and local knowledge.



(**a**) Gdańsk.

(**b**) Poznań.

(**c**) Warsaw.

(**d**) Wrocław.

Figure 42. Mobile relay and city area influence on multicast core of 30 min graphs.

6.5. Duration Influence

Apart from the choice of the algorithm that will be used, the key controllable factor that influences the efficacy of the introduced multicasting methods is the duration of the network, i.e., the maximum life-span of the message. While the 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 the duration 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 43 shows that not only does the number of nodes increase with the increase in 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 number increases when the radio range grows. In the 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 in duration from 60 to 120 min 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 to a steady level already in 30 min networks.

As has been shown in more detail in Figure 20, each multicasting knowledge mode results in virtually the same delivery ratios, i.e., the percentages of reached destinations. In Figure 44, 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 the network duration is at least 60 min. When the radio range increases, the duration of only 30 min might also be considered satisfactory in some scenarios. Lower durations result in more distributed, and hence, less predictable, delivery ratios. In general, Warsaw is the area that approaches the highest reached destination ratios. Wrocław remains behind in all variants. The constructed multicast structures do not even reach half of the destinations when the radio range is the lowest. When the radio range increases to 50 m, a little over 60 percent of destinations can be reached when the duration is at least 60 min. Further radio range growth increases the numbers of reached nodes even at lower durations.

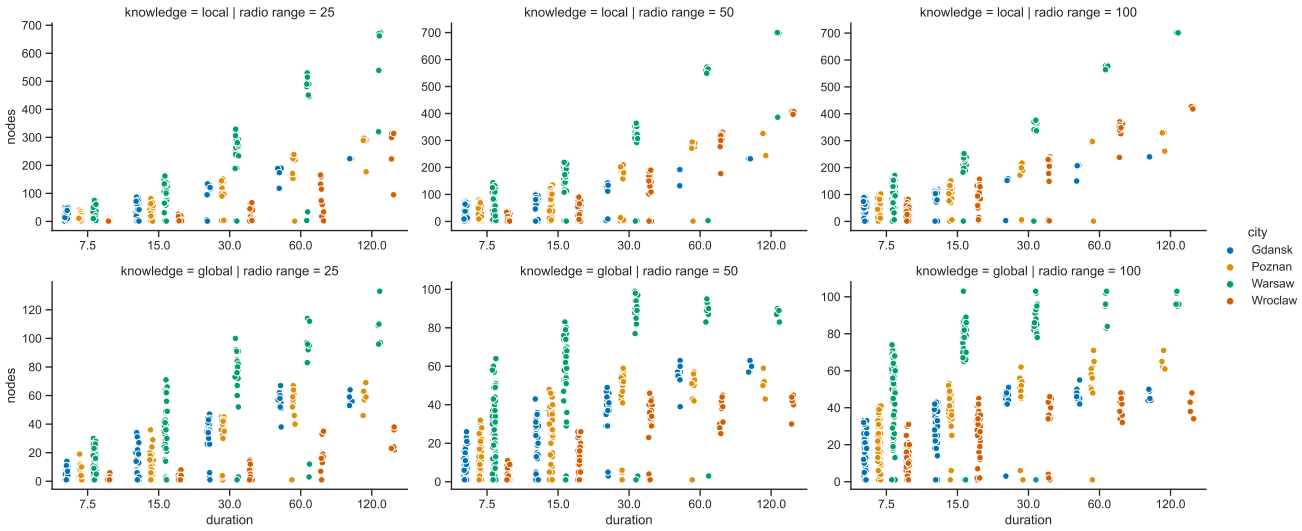


Figure 43. All nodes in multicast graphs with destination regions and local or global knowledge.

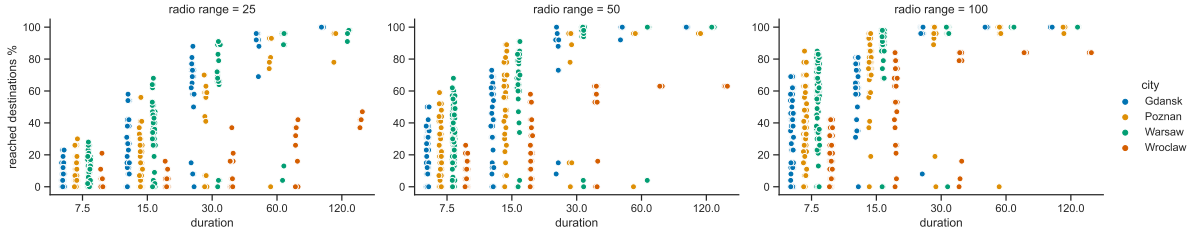


Figure 44. Reached-to-potential destinations ratio in multicast graphs with destination regions and local knowledge.

With the shortest network duration in Figure 45, the highest and the most distributed numbers of core nodes ratios in local-knowledge-based graphs can be observed. This means that when the network lasts for the shortest time, and therefore, consists of a lower number of nodes, the structures are of the highest similarity to the solutions constructed with global knowledge. When the duration increases, i.e., the messages are allowed to be distributed in the network over a longer period, the overhead also increases in the no- and local knowledge variants. This is caused by the flooding-based nature of these approaches. If the duration is set to 30 min, the number of core nodes drops to around 20–30 percent and decline a bit further with a duration increase. The ratio is zero percent when the graph has no nodes that lead to at least one destination. One hundred percent means that the local-knowledge-based multicast structure consists only of nodes leading to the destinations. This can happen either when the graph is small and reaches only a number of potential destinations or the algorithm manages to reach all of these.

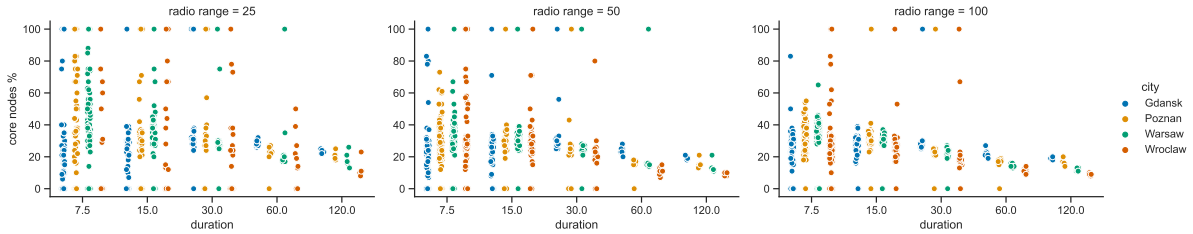


Figure 45. Core nodes ratio in multicast graphs with destination regions and local knowledge.

Both the core space cost ratio in Figure 46 and the core time cost ratio in Figure 47 become less vertically distributed when the duration increases. Radio range growth entails a decrease in the part of these costs that are related to the core multicast structures built with local knowledge. This trend is visible more in the case of the core time cost ratio. This is because when the network is excessively expanded over time, the time cost becomes 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.

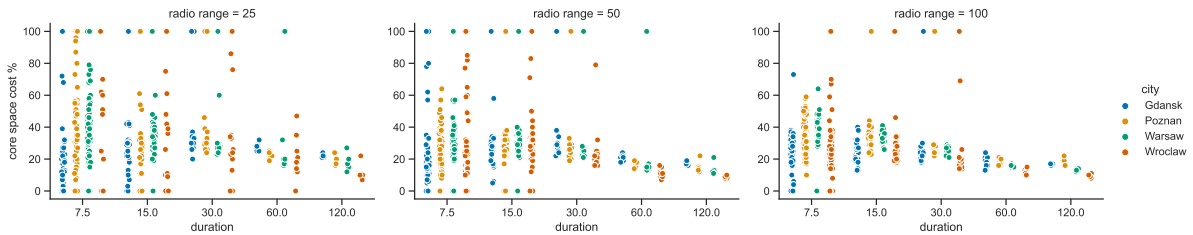


Figure 46. Core space cost ratio in multicast graphs with destination regions and local knowledge.

The core time span ratios in Figure 48 show that with durations of 30 min 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 becomes more and more apparent that the spacial and temporal span of the network becomes excessive. The core time span cost ratios presented in Figure 49 express the distributions and trends correlated with the ones of the core time cost ratios in Figure 47.

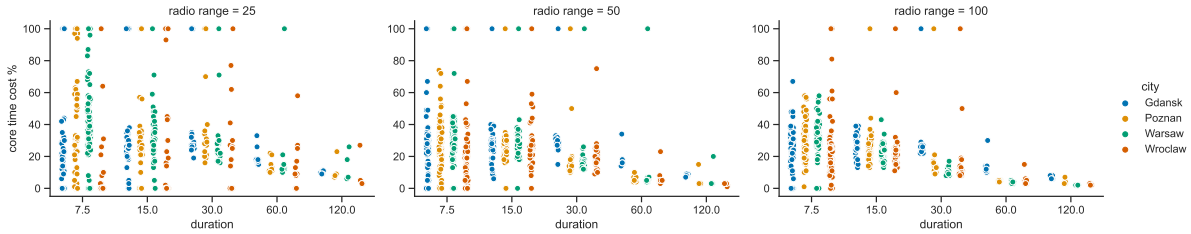


Figure 47. Core time cost ratio in multicast graphs with destination regions and local knowledge.

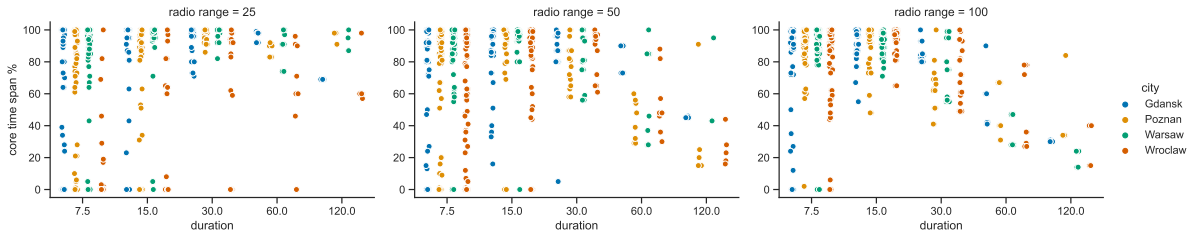


Figure 48. Core time span ratio in multicast graphs with destination regions and local knowledge.

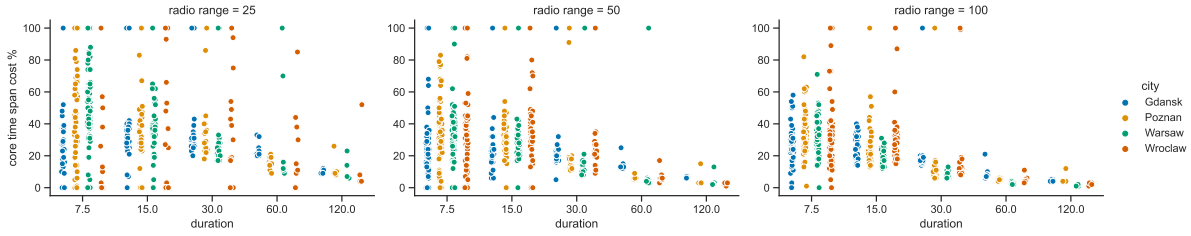


Figure 49. Core time span cost ratio in multicast graphs with destination regions and local knowledge.

6.6. Observations

The introduced and analyzed opportunistic multicast routing algorithms are the first ones designed for heterogeneous urban sensor networks using uncontrolled mobile relays. Moreover, no previous study has analyzed dynamic time-spanning time-changing multicast topologies in urban sensor networks modeled as graphs based on real node location data. Key observations stemming from the presented space–time multicast analysis are as follows:

- In general:
  - Algorithms designed for *delay-tolerant multicast routers* (DTMRs) can be used in multicast communication in an 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, most of the modeled graphs consist of significantly more nodes;
  - An increase in the 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 be constructed;

  - The area in Wrocław is the one with the most significant relative node number growth when the radio range increases.

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

- In terms of costs:
  - Space costs are the lowest in the case of global-knowledge-based routing. The ones in Warsaw are the highest in the areas of interest. These costs increase almost linearly with the increase in the number of mobile relays. Also, the larger the radio range, the higher the space cost of the multicast graphs. This indicates the related increasing power use caused by radio transmissions;
  - Time costs are positively correlated with the number of mobile relays. Therefore, the more mobile relays involved, the more computing resources of the network will be used;
  - The time spans are equal for the 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 the radio range increases and the number of destination nodes decreases. The global knowledge approach can be significantly less resource consuming and minimize the overall use of the storage capacity of the relays;
  - The time span cost grows when the radio range increases because more nodes are connected to single relays. When this cost becomes higher, the message buffers of the relays are occupied for longer;
  - The time cost and time span cost become mostly the overhead when the 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 the 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 the greedy opportunistic network-flooding-related nature of the algorithms;
  - Global knowledge multicasting involves, in general, significantly fewer relay nodes and uses them for shorter durations. Transmission medium usage is also expected to be lower. Therefore, the metrics are 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 number of reached multicast destinations;
  - A radio range of 50 m and maximum message distribution time (duration) set to 30 min, by the means of using the *time-to-live* (TTL) parameter, can be considered as the probable optimal settings for the investigated urban areas. When a higher

delivery ratio is required or the area of interest gorws, the duration has to be extended or the range (coverage) increased, when feasible.

The heterogeneous structures were constructed based on open data on the location of infrastructure and public transportation vehicles 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.

7. Summary

This paper introduces the first multicast algorithms aimed at heterogeneous opportunistic urban sensor networks using uncontrolled mobile relays. They are designed with practicality in mind, are protocol-agnostic, modular, and presented 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 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.

The novel unique multidimensional analysis of the introduced algorithms is conducted based on publicly available node location data for four Polish cities—Gdańsk, Poznań, Warsaw, and Wrocław. Time-spanning heterogeneous multicast communication structures, modeled as time-changing graphs, use both stationary and uncontrolled mobile relay nodes. They are studied in custom-developed simulation research environment related to the open-data network connectivity modeling architecture defined by Musznicki et al. in [10]. 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 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 the involved nodes, as compared to no-network-knowledge scenarios.

Main contributions of the presented work:

- Urban delay-tolerant multicast algorithms using uncontrolled mobile relays:
  - No-knowledge opportunistic multicast;
  - Local knowledge opportunistic multicast;
  - Global knowledge opportunistic multicast.
- Graph-based study of introduced multicast algorithms:
  - Methodology for multidimensional graph-based modeling and analysis;
  - Simulation architecture and custom-developed research environment;
  - Comparative investigation and observations for four Polish cities.

Suggested directions for further research:

- Urban delay-tolerant multicast algorithms using uncontrolled mobile relays:
  - 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;
  - Modeling the usage of acknowledgments and retransmissions;
  - Implementation of advanced buffer and duty-cycle management, as well as radio spectrum and power-preserving schemes;

  - Making routing decision in relation to node movement direction and speed, as well as the history of previous and predicted contacts;
  - Introduction of machine-learning-based optimizations.
- Graph-based study of introduced multicast algorithms:
  - Investigation of extended scale and scope:
    * More data sources, including the closed ones, when available;
    * More areas of different locations and sizes, e.g., suburbs, small cities, countryside, etc.;
    * Different periods of interest;
    * Various shapes of destination regions.
  - Advanced radio connectivity modeling:
    * Use of actual connectivity data, when available;
    * Use of complex radio coverage models (e.g., for LoRa-based networks [55]).
  - Different sets of multicast parameters and destinations:
    * Solution-related node classes and roles;
    * Region-based multicast to nodes of desired classes.
- Network-traffic-flow-based study of presented multicast algorithms:
  - Analysis based on the usage of discrete-event simulation:
    * Data routing in a network simulator with radio propagation, communication, and power consumption models (e.g., OMNeT++ [56] or ns-3 [57]);
    * Node movement determined by a road traffic simulator (e.g., SUMO [58]) exposed by a vehicular network simulation framework (e.g., Veins [59]);
    * Use of an opportunistic DTN simulator (e.g., The ONE [60]).
  - 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 wireless sensor networks in which uncontrolled mobile relay nodes are used. Moreover, a novel opportunistic multicast modeling and investigation approach was developed and proven valid in the implemented research environment. More than 16,000 multicast graphs were constructed and examined with numerous metrics. Various parameters, trends, and interrelationships were investigated and discussed, confirming the applicability of the introduced urban delay-tolerant multicast algorithms. Directions for further research were suggested as well.

Author Contributions: B.M. and P.Z.: conceptualization, validation, writing—review and editing; B.M.: data curation, formal analysis, investigation, methodology, resources, software, visualization, writing—original draft; P.Z.: funding acquisition, project administration, supervision. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Polish Ministry of Science and Higher Education (No. 0313/SBAD/1310).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Musznicki, B.; Zwierzykowski, P. Survey of Simulators for Wireless Sensor Networks. *Int. J. Grid Distrib. Comput.* **2012**, *5*, 23–50.
2. Musznicki, B. Empirical Approach in Topology Control of Sensor Networks for Urban Environment. *J. Telecommun. Inf. Technol.* **2019**, *1*, 47–57. [CrossRef]

3. Shah, R.C.; Roy, S.; Jain, S.; Brunette, W. Data MULEs: Modeling and Analysis of a Three-tier Architecture for Sparse Sensor Networks. *Ad Hoc Netw.* **2003**, *1*, 215–233. [CrossRef]
4. Benhamida, F.Z.; Bouabdellah, A.; Challal, Y. Using delay tolerant network for the Internet of Things: Opportunities and challenges. In Proceedings of the 2017 8th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 4–6 April 2017; pp. 252–257. [CrossRef]
5. Iqbal, M.; Wang, X.; Wertheim, D.; Zhou, X. SwanMesh: A multicast enabled dual-radio wireless mesh network for emergency and disaster recovery services. *J. Commun.* **2009**, *4*, 298–306. [CrossRef]
6. Roy, A.; Bose, S.; Acharya, T.; DasBit, S. Social-based energy-aware multicasting in delay tolerant networks. *J. Netw. Comput. Appl.* **2017**, *87*, 169–184. [CrossRef]
7. Zhou, X.; Durrani, S.; Guo, J. Drone-Initiated D2D-Aided Multihop Multicast Networks for Emergency Information Dissemination. *IEEE Access* **2020**, *8*, 3566–3578. [CrossRef]
8. Wong, K.S.; Wan, T.C. Current State of Multicast Routing Protocols for Disruption Tolerant Networks: Survey and Open Issues. *Electronics* **2019**, *8*, 162. [CrossRef]
9. Musznicki, B.; Piechowiak, M.; Zwierzykowski, P. Modeling Real-Life Urban Sensor Networks Based on Open Data. *Sensors* **2022**, *22*, 9264. [CrossRef] [PubMed]
10. Musznicki, B.; Piechowiak, M.; Zwierzykowski, P. Modeling and Analyzing Urban Sensor Network Connectivity Based on Open Data. *Sensors* **2023**, *23*, 9559. [CrossRef] [PubMed]
11. Piechowiak, M.; Zwierzykowski, P.; Musznicki, B. LoRaWAN Metering Infrastructure Planning in Smart Cities. *Appl. Sci.* **2023**, *13*, 8431. [CrossRef]
12. Kliks, A.; Musznicki, B.; Kowalik, K.; Kryszkiewicz, P. Perspectives for Resource Sharing in 5G Networks. *Telecommun. Syst.* **2017**, *68*, 605–619. [CrossRef]
13. Crockford, D. The Application/Json Media Type for JavaScript Object Notation (JSON). RFC 4627, IETF. 2006. Available online: https://www.rfc-editor.org/rfc/rfc4627.html (accessed on 20 October 2023).
14. Burleigh, S.; Fall, K.; Birrane, E.J. Bundle Protocol Version 7. RFC 9171. 2022. Available online: https://doi.org/10.17487/RFC9171 (accessed on 20 October 2023).
15. Vahdat, A.; Becker, D. Epidemic Routing for Partially Connected Ad Hoc Networks. In *Technical Report cs-2000-06*; Duke University: Durham, NC, USA, 2000.
16. Braginsky, D.; Estrin, D. Rumor Routing Algorithm for Sensor Networks. In Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, New York, NY, USA, 28 September 2002; WSNA '02, pp. 22–31.
17. Sanchez, J.A.; Ruiz, P.M.; Liu, J.; Stojmenovic, I. Bandwidth-Efficient Geographic Multicast Routing Protocol for Wireless Sensor Networks. *IEEE Sens. J.* **2007**, *7*, 627–636. [CrossRef]
18. Basagni, S.; Carosi, A.; Petrioli, C. Controlled vs. Uncontrolled Mobility in Wireless Sensor Networks: Some performance insights. In Proceedings of the 2007 IEEE 66th Vehicular Technology Conference, Baltimore, MD, USA, 30 September–3 October 2007; pp. 269–273.
19. Musznicki, B.; Zwierzykowski, P. Performance Evaluation of Flooding Algorithms for Wireless Sensor Networks Based on EffiSen: The Custom-Made Simulator. In *Simulation Technologies in Networking and Communications: Selecting the Best Tool for the Test*; Pathan, A.S.K., Monowar, M.M., Khan, S., Eds.; CRC Press, Taylor & Francis Group: Boca Raton, FL, USA, 2015; pp. 433–458.
20. Leach, P.J.; Salz, R.; Mealling, M.H. A Universally Unique IDentifier (UUID) URN Namespace. RFC 4122. 2005. Available online: https://doi.org/10.17487/RFC4122 (accessed on 20 October 2023).
21. Musznicki, B.; Tomczak, M.; Zwierzykowski, P. Geographic Dijkstra-based Multicast Algorithm for Wireless Sensor Networks. *Int. J. Image Process. Commun. Spec. Issue Algorithms Protoc. Pack. Netw.* **2012**, *17*, 33–46. [CrossRef]
22. Zhao, W.; Ammar, M.; Zegura, E. Multicasting in Delay Tolerant Networks: Semantic Models and Routing Algorithms. In Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-Tolerant Networking, New York, NY, USA, 22–26 August 2005; WDTN '05, pp. 268–275. [CrossRef]
23. NetworkX–Network Analysis in Python. Available online: https://networkx.org (accessed on 20 October 2023).
24. OpenStreetMap. Available online: https://www.openstreetmap.org/copyright (accessed on 20 October 2023).
25. Głąbowski, M.; Musznicki, B.; Nowak, P.; Zwierzykowski, P. Shortest Path Problem Solving Based on Ant Colony Optimization Metaheuristic. *Int. J. Image Process. Commun. Spec. Issue Algorithms Protoc. Pack. Netw.* **2012**, *17*, 7–17. [CrossRef]
26. Dijkstra, E. A Note on Two Problems in Connexion with Graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
27. Grigorescu, E.; Lin, Y.S.; Quanrud, K. Online Directed Spanners and Steiner Forests. In Proceedings of the Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques—APPROX/RANDOM 2021, Seattle, WA, USA, 16–18 August 2021.
28. McKinney, W. Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference, Austin, TX, USA, 28 June–3 July 2010; pp. 56–61.
29. Waskom, M.L. seaborn: Statistical data visualization. *J. Open Source Softw.* **2021**, *6*, 3021. [CrossRef]
30. Department of Defense World Geodetic System 1984: Its Definition and Relationships with Local Geodetic Systems. Second Edition. In *Technical Report*; Defense Mapping Agency: Washington, DC, USA, 1991.

31. Open Gdańsk–GPS Positions of the Vehicles. Available online: https://ckan.multimediagdansk.pl/dataset/tristar/resource/0683c92f-7241-4698-bbcc-e348ee355076 (accessed on 20 October 2023).
32. Open Gdańsk–List of Bus Stops. Available online: https://ckan.multimediagdansk.pl/dataset/tristar/resource/4c4025f0-01bf-41f7-a39f-d156d201b82b (accessed on 20 October 2023).
33. Open Gdańsk–Positions of Ticket Machines. Available online: https://ckan.multimediagdansk.pl/dataset/tristar/resource/af7bf4a9-e62e-4af2-906a-fa27c2532dfd (accessed on 20 October 2023).
34. ZTM Poznań–For Developers–GTFS-RT. Available online: https://www.ztm.poznan.pl/pl/dla-deweloperow/gtfsRtFiles (accessed on 20 October 2023).
35. Poznań–Positions of Public Transport Stops. Available online: http://www.poznan.pl/mim/plan/map_service.html?mtype=pub_transport&co=cluster (accessed on 20 October 2023).
36. Poznań–Positions of Ticket Machines. Available online: http://www.poznan.pl/mim/plan/map_service.html?mtype=pub_transport&co=class_objects&class_id=4000 (accessed on 20 October 2023).
37. Warsaw Open Data–Public Vehicle Positions–API Documentation. Available online: https://api.um.warszawa.pl/files/9fae6f84-4c81-476e-8450-6755c8451ccf.pdf (accessed on 20 October 2023).
38. Warsaw Open Data. Available online: https://api.um.warszawa.pl (accessed on 20 October 2023).
39. Wrocław Open Data–Positions of Public Transporation Vehicles. Available online: https://www.wroclaw.pl/open-data/dataset/lokalizacjapojazdowkomunikacjimiejskiejnatrasie_data (accessed on 20 October 2023).
40. Wrocław Open Data–Wrocław City Bike Stations. Available online: https://www.wroclaw.pl/open-data/dataset/nextbikesoap_data/resource/42eea6ec-43c3-4d13-aa77-a93394d6165a (accessed on 20 October 2023).
41. Wrocław Open Data–Vozilla–City Electric Car Rental–Parking Lots. Available online: https://www.wroclaw.pl/open-data/dataset/wykaz-miejsc-parkingowych-miejskiej-wypozyczalni-samochodow-elektrycznych-vozillaa (accessed on 20 October 2023).
42. Airly Developer–Documentation. Available online: https://developer.airly.org/en/docs (accessed on 20 October 2023).
43. Gdańsk w Liczbach–Liczba Mieszkańców Gdańska. Available online: https://www.gdansk.pl/gdansk-w-liczbach/mieszkancy,a,108046 (accessed on 20 October 2023).
44. Geoportal Krajowy Na Mapie. Available online: https://geoportal-krajowy.pl (accessed on 20 October 2023).
45. Gdańsk Municipal Transport Authority–Timetables. Available online: https://ztm.gda.pl/rozklady (accessed on 20 October 2023).
46. Poznan.pl–Znamy Liczbę Mieszkańców Poznania. Available online: https://www.poznan.pl/mim/info/news/znamy-liczbe-mieszkancow-poznania,188075.html (accessed on 20 October 2023).
47. Poznań Municipal Transport Company–Timetable. Available online: https://www.mpk.poznan.pl/en/timetable/ (accessed on 20 October 2023).
48. Statystyka Warszawy–Miasto Warszawa. Available online: https://um.warszawa.pl/statystyka-warszawy-2022 (accessed on 20 October 2023).
49. Warsaw Public Transport–Timetables. Available online: https://www.wtp.waw.pl/en/timetables/ (accessed on 20 October 2023).
50. Statistical Office in Wroclaw–Population. Available online: https://wroclaw.stat.gov.pl/en/zakladka2/ (accessed on 20 October 2023).
51. Wrocław Municipal Transport Company–Timetable. Available online: https://www.wroclaw.pl/komunikacja/rozklady-jazdy (accessed on 20 October 2023).
52. Karvonen, H.; Pomalaza-Ráez, C.; Mikhaylov, K.; Hämäläinen, M.; Iinatti, J. Experimental Performance Evaluation of BLE 4 Versus BLE 5 in Indoors and Outdoors Scenarios. In *Proceedings of the Advances in Body Area Networks I*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 235–251.
53. Ferreira, A.E.; Ortiz, F.M.; Costa, L.H.M.; Foubert, B.; Amadou, I.; Mitton, N. A study of the LoRa signal propagation in forest, urban, and suburban environments. *Ann. Telecommun.* **2020**, *75*, 333–351. [CrossRef]
54. Robusto, C.C. The cosine-haversine formula. *Am. Math. Mon.* **1957**, *64*, 38–40. [CrossRef]
55. Griva, A.I.; Boursianis, A.D.; Wan, S.; Sarigiannidis, P.; Psannis, K.E.; Karagiannidis, G.; Goudos, S.K. LoRa-Based IoT Network Assessment in Rural and Urban Scenarios. *Sensors* **2023**, *23*, 1695. [CrossRef] [PubMed]
56. Varga, A. A Practical Introduction to the OMNeT++ Simulation Framework. In *Recent Advances in Network Simulation: The OMNeT++ Environment and its Ecosystem*; Virdis, A., Kirsche, M., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 3–51.
57. Riley, G.F.; Henderson, T.R. The ns-3 Network Simulator. In *Modeling and Tools for Network Simulation*; Wehrle, K., Güneş, M., Gross, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 15–34.
58. Lopez, P.A.; Behrisch, M.; Bieker-Walz, L.; Erdmann, J.; Flötteröd, Y.P.; Hilbrich, R.; Lücken, L.; Rummel, J.; Wagner, P.; Wießner, E. Microscopic Traffic Simulation using SUMO. In Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems, Maui, HI, USA, 4–7 November 2018.

59. Sommer, C.; German, R.; Dressler, F. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Trans. Mob. Comput.* **2011**, *10*, 3–15. [CrossRef]
60. Keränen, A.; Ott, J.; Kärkkäinen, T. The ONE Simulator for DTN Protocol Evaluation. In Proceedings of the SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, Rome, Italy, 2–6 March 2009.