CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY

ASSIGNMENT OF BACHELOR'S THESIS

Title:	Decentralized exploration of an unknown environment
Student:	Khilda Slyusar
Supervisor:	RNDr. Miroslav Kulich, Ph.D.
Study Programme:	Informatics
Study Branch:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	Until the end of summer semester 2016/17

Instructions

Assume a team of autonomous terrestrial robots equipped with ranging sensors exploring an unknown flat 2D terrain. The aim of the thesis is to extend simulation SW of the exploration framework developed by IMR, CIIRC with communication functionalities enabling decentralized coordination of a team of exploring robots.

Specifically:
1. Get acquainted with approaches to multi-robot unknown space exploration.
2. Get acquainted with message oriented middlewares (ZeroMQ, ActiveMQ Apollo).
3. Design communication and coordination functionalities for a multi-robot team performing exploration. Especially, design exchange of maps built by the particular robots, and negotiation of goals to go to in the next exploration step under limited communication range.
4. Employ a chosen message oriented middleware to implement the designed functionalities.
5. Verify the implemented communication and coordination functionalities experimentally in the provided simulation environment.

References

Will be provided by the supervisor.

L.S.

doc. Ing. Jan Janoušek, Ph.D.
Head of Department

prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague February 4, 2016

Czech Technical University in Prague

Faculty of Information Technology

Department of Computer Science

Bachelor's thesis

Decentralized exploration of an unknown environment

Khilda Slyusar

Supervisor: RNDr. Miroslav Kulich, Ph.D.

18th May 2016

Acknowledgements

I would like to truly thank my supervisor RNDr. Miroslav Kulich, Ph.D. for the time and energy that he spent on our consultations. I want to express gratitude to my parents and my younger sister Eva who were always very supportive.

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum provided under the programme "Projects of Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on 18th May 2016 . . . .

Citation of this thesis

Slyusar, Khilda. *Decentralized exploration of an unknown environment.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2016.

Abstrakt

Tato bakalářská práce se zabývá problémem decentralizovaného prohledávání neznámého prostředí homogenním týmem autonomních robotů. Jako základ navrženého systému slouží softwarový rámec pro prohledávání více roboty vyvinutý skupinou Inteligentní a mobilní robotiky Českého institutu informatiky, robotiky a kybernetiky, který byl v rámci práce rozšířen o možnost kooperace mezi roboty v týmu. Práce obsahuje porovnání různých protokolů směrování, porovnání přístupů pro výměnu informací v ad hoc sítích a porovnání přístupů koordinace více robotů. Návrh systému rovněž zahrnuje detailní analýzu možných komunikačních vzorů v distribuovaných robotických systémech. Výsledkem této analýzy je popis struktury aplikace, která byla posléze implementována. Testování vyvinutých funkcionalit bylo provedeno souborem experimentů s různými nastaveními exploračního procesu a s různým počtem robotů.

Klíčová slova multi-robotické systémy, decentralizované prohledávání, simulace, omezená komunikace, kooperace

Abstract

This bachelor's thesis deals with the problem of decentralized exploration of an unknown environment by a homogeneous multi-robot team. The exploration framework for multi-robot terrain exploration by Intelligent and Mobile Robotics Group at the Czech Institute of Informatics, Robotics and Cybernetics is used as the basis for the design and implementation of an application that has the functionality to cooperate with other team members. The thesis includes a comparative analysis of the various routing protocols, a comparison of message passing approaches in ad hoc networks and a comparison of algorithms for multi-robot coordination. The design includes a detailed analysis of the possible communication patterns for distributed multi-robot system. The result of the analysis is a structure description of the implemented application. Testing of developed functionality is based on the results of several experiments with various input conditions of the exploration process and various teams of robots.

Keywords multi-robot systems, decentralized exploration, simulation, limited communication, cooperation

Contents

List of Figures

List of Tables

Introduction

A mobile robot is a mechanism, which has ability to move, get and gather information from its sensors to produce a model of the environment, and send obtained data to other robots to accelerate the spread of knowledge about the terrain. Such a robot is usually autonomous, which means that the robot performs its task without any human intervention. Suchwise, a human participates only in problem definition. One of the fundamental problems for this kind of robots is exploration. That issue occurs when an environment model is not known in advance. The solution to this problem implies getting a full map of the terrain. At the same time, it requires minimizing the exploration time or the power consumption as an additional target.

There are several applications, where the complete coverage of a terrain during exploration is an integral part of a robotic mission. It might be extensive tasks like planetary exploration, reconnaissance, rescue, or applied tasks like mowing or cleaning.

The aim of this thesis is to create a framework with communication functionalities enabling decentralized coordination of a homogeneous team of mobile robots. General description of the terrain exploration problem, the analysis of existing approaches of the communication implementation in distributed systems and the comparison of various approaches to the coordination of the multi-robot team are placed in the Chapter 1. The Chapter 2 discusses the possible methods of implementation of communication between particular modules. A comparison of ZeroMQ and Apollo ActiveMQ is also described here. The communication establishment scheme and the general scheme of communication between the team members are proposed based on the selected library. The Chapter 3 contains a description of the provided environment framework by Intelligent and Mobile Robotics Group at the Czech Institute of Informatics, Robotics and Cybernetics. The structure of developed application is also described here. The Chapter 4 contains results of the experiments, which allow to verify the performance of the developed framework. The final evaluation of the thesis is in the Conclusion.

Chapter 1

Theory

1.1 Task of multi-robot exploration of an unknown environment

Existing approaches consider exploration in a wide range from practical solutions in the real world with physical robots to theoretical solutions with virtual robots. In general, exploration is an iterative process that, at each step performs the following operations: the robot receives information from its sensors, updates an internal model of the surrounding space on the basis of these data, selects the next goal for the navigation on the basis of current knowledge, and gradually moves to the selected destination. The process continues until there are no more available goals, that is, all areas of the environment will be explored [1]. The main source of optimization is to prepare a set of possible goals and select from them a next goal, which is the most favorable for exploration purposes. One of the most known approaches is frontier-based exploration [2], which will be considered in this thesis.

For purposes of this approach, the terrain model is represented as an *occupancy grid*, which was first described in [3]. Occupancy grids are used to represent an environment as a discrete grid. Each cell of the grid corresponds to an area of the terrain, and contains a probability that the obstacle can be in this area. By default, a cell corresponding to the unexplored area of the environment is assigned an initial probability (typically 0.5). If the occupancy probability in the cell is more than the initial probability, the corresponding area is assumed to be an obstacle. If the occupancy probability is less than the initial probability, then the corresponding area is free from obstacles.

There are several possible ways for obtaining information about an environment, such as a monocular camera, a sonar or a distance sensor with a limited range (also known as a laser rangefinder), see more details in [4]. This thesis will proceed from the fact that the robots are equipped with *laser rangefinders*. It is a device that allows to determine the distance to an object with a

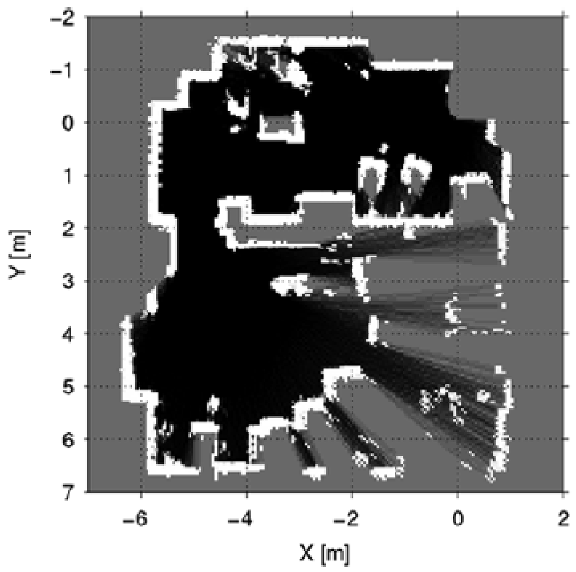


Figure 1.1: An example of the occupancy grid constructed on the basis of data obtained from the laser rangefinder [4]

laser beam. This sensor has a limited visibility, however, it can gather data around itself, launching beams sequentially in different directions. Thus, the robot receives from the sensor a set of distances. This set will be herein called a laser scan. The distances from the laser scan may be transferred to coordinates on the grid using the sensor's coordinates at the time of the scanning, the sensor's initial rotation angle and the angular resolution, which depends on the number of distance's measurements.

The next step after receiving the data from the sensor is to add new data to the existing occupancy grid. It uses a Bayesian based updating technique. Based on the assumption that all occupancy grid cells are independent, the probability for each cell is updated separately. Bayes rule is used for the computation of a new cell probability based on the laser scan's data. It defines a new cell probability as the current cell probability multiplied by the probability for a given cell according to the sensor data, and multiplied by a normalization constant.

Figure 1.1 shows an example of the occupancy grid that can be built on the basis of data received from the laser rangefinder. Cells with a high occupancy probability are displayed in white. Cells which have an initial probability represented in gray, which means that these cells have not yet been explored. Black cells indicate a low occupancy probability, that is, this area of the environment is free.

After updating the occupancy grid a robot needs to select a set of goals towards which it can move. Frontier-based exploration is based on the concept of *a frontier*, which is a boundary between the already known part of the terrain

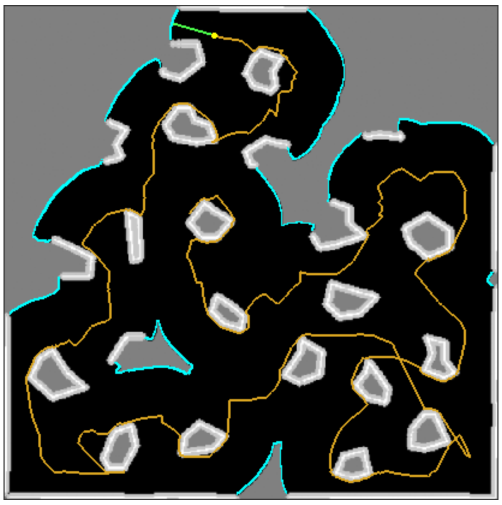


Figure 1.2: An example of frontier-based exploration carried out by a single robot

and still an unknown part. The occupancy grid system uses the definition of frontier cell, i.e. a free cell, which is an adjacent with at least one unexplored cell. Adjacent frontier cells are combined into frontier regions. Each region with a size comparable to the size of the robot is considered a frontier [2].

In the phase of choosing the best goal, the robot must choose one goal to which it will move. It is selected from the frontiers, which had been prepared in the previous step, and should provide an increase in knowledge about the environment. Moving to one of these frontiers allows to expand the known area. The list of frontiers is updated during exploration of new areas. The robot carries out the removal of the points that are in the already known territory, and adds new ones that are on the new border between the known and the unknown part of the terrain.

The stop condition of the exploration is the lack of frontiers, that is, in terms of the occupancy grid - there is no free cell that is adjacent to an unexplored cell.

This thesis implies a common coordinate grid for all robots in the team and each robot knows its initial position on this grid in advance. That is, the thesis does not consider the problem of localization of robots in the terrain, and localization of laser scans on the occupancy grid received from sensors.

In the case where more than a single robot are involved in exploration, the data obtained from other robots may be an additional source of information about the terrain. For this reason, the robots are equipped with a wireless communication device with the same distance range allowing to broadcast messages. The data are transferred between two robots, if they are in range of each other devices, regardless of the presence of obstacles in between. Such an assumption was made for the convenience of the simulation.

Figure 1.2 shows the process of the frontier-based exploration of the square terrain with obstacles by one robot. The known obstacle-free zone of the map is colored in black. The unexplored area of the environment is painted in gray. The closed polylines are obstacles. The filling of obstacles is also gray as for unknown zone of the map, because these areas are not available for exploration. The yellow point is a representation of the robot. The travelled path is shown as the orange curve. All the blue points are the frontiers, because they lie on the boundary between the known (black) and unknown (gray) area. The robot moves to one of the points on the border of zones to expand the known region. The line between the current position of the robot and this target point on the border is colored in green. The exploration will end when there is no blue point. This means that all achievable unknown zones will be explored by the robot.

1.1.1 Single robot vs. multi-robot system

Exploration of terrain can be carried out by a single robot. This implementation will be easier since there is no need to develop a system of communication and cooperation, in contrast to approaches employing a robotic team. A single robot has a bounded speed, so it needs an effective algorithm of goals selection. The majority of present articles deals with the problem of developing an optimal algorithm for the exploration of an unknown environment, which minimizes time and energy spent by a robot. However, the effectiveness of exploration will be always limited by real-world requirements, such as the maximum possible speed of the robot.

In addition to physical limitations, exploration by one robot strongly depends on the accuracy and fidelity of the data received from sensors. In the case of an optimal algorithm for exploration, information obtained from the sensor will not overlap. In other words, the robot will be received only the data from still unexplored areas of terrain. That is, there will be no possibility of eliminating sensor error with repeated application data from the same zone.

Besides the errors in the data that may be obtained from the sensor, the robot itself can break, or it may run out of energy. In this case, the launched exploration of the terrain will not be completed at all.

The solution of these problems is to use a team of robots instead of a single robot. Using a group of mobile robots enables to overcome physical restrictions. The team divides a solution of the problem between all members, so the overall exploration time will decrease. Due to the fact that group members do not transmit complete information about the known zones of the map to each other, it is possible to explore the same area of the terrain by another robot. Some amount of overlapping is desirable, because the repeated coverage decreases the mission's efficiency. Thereby, merging of overlapping information helps to avoid inaccuracies in data obtained from the sensors to create more precise map. Additionally, the team of robots is more fault-tolerant. It means that single robot failure in the team is acceptable as the rest of the robots can continue this exploration.

The use of a team allows to significantly optimize the process of exploration, however, it is necessary to consider the negative side. The robot as a part of a team must be able to create a connection and share information with others, to consider and analyze incomings from other robots' data. Development and implementation of these functionalities greatly complicates creation of such a robot and testing its working in the team. Environment research by a team also contributes to the process a certain degree of ambiguity. The robot does not always have information about the goals of other robots in the group, or other robots may force it to change its actions, for example, to select another frontier as the goal of the exploration.

The main difficulty is to coordinate actions of team members. A large number of robots allows to accelerate the process of terrain exploration, however, the establishment of an effective coordination of all team members has an important role. As described above, in this thesis the robots have a common coordinate system and there is no need to conduct localization. Thereby, each robot knows its own position relative to the others. The challenge of effective coordination in this case is to send team members to different non-overlapping areas of the terrain. Frontier-based exploration is best suited to implement such coordination, because it aims to move to the border between the known and unknown part of the environment [5].

The next challenge is integration of information from different robots into a consistent map. The solution highlighted in [6], which describes frontier-based exploration, is extended to apply to a team of robots. Each robot in the team maintains and supports the relevance of a grid-based map. During the movement to the new selected frontier, the robot receives information from its sensor and creates a local map based on the obtained data. This local map is added to its own global map and sent to the rest of the robots. Other robots receive this part of the map, add it to the collection of local maps from other robots, and update its global map with the new local map. Based on the maps received from other robots, the robot reduces its ignorance of the terrain and modifies the list of frontiers. This approach allows the exploration to be cooperative and decentralized at the same time. Local maps obtained by one robot are available to everyone else in a communication range. Thereby, the robot can determine which areas of the environment have already been explored by other robots and to choose another area to exploration.

The local maps are transmitted immediately after the data is received from sensors, i.e. a team of robots becomes robust against breakage of individual robots. If the robot is broken or has ended energy, it stops sending the local maps. However, other robots will continue to explore, regardless of the operability of the other team members. Also, such a system is asynchronous. Other robots do not wait for each other to perform some action and, therefore, failure of one will not provoke others stop.

However, there is a problem caused by real-world conditions and limited bandwidth of communication channels. The presence of a large number of robots in the team and active communication between them can provoke a drop in the communication line. The solution is the ability of each robot to act independently from the other team members and to conduct the exploration on its own without the guidance of some central point of control. Thus, the team of robots needs to have a decentralized architecture of multi-robot systems to work effectively under conditions of limited communication [5].

The problem of decentralized terrain exploration by a multi-robot team requires solving the problem of communication between team members and the problem of coordination its actions. The following sections provide an analysis of existing approaches to communication in distributed systems. Possible approaches for the implementation of coordination within the team also will be discussed.

1.2 Mobile ad hoc networks

There are two types of networks [7]: with a pre-established infrastructure and without it. The second type of network is called an ad hoc network. It is a multi-hop wireless network that has no predetermined structure. Members of such a network have the ability to move independently and to establish relationships with the others of the network. The main feature of ad hoc network is a dynamically changing topology, because network nodes can continuously move. The absence of fixed structure and centralized administration is a consequence of the high mobility of the network members. However, this network has limited bandwidth capabilities of the wireless connection and it is necessary to consider that each node has a limited amount of energy.

It may be necessary in ad hoc networks to hop several nodes, until the destination receives a packet. It is therefore necessary to have a routing protocol that describes how two nodes must communicate with each other. The two main functions of a routing protocol are to choose a route to deliver messages between the source and receiver and deliver them to the correct destination.

The basic routing protocols are *link-state routing* and *distance vector*. In the first case, each node has a complete view of the network topology and knows the exact cost of the route to any other point in the network. The cost function returns a value, which allows to determine the effectiveness of one route over the other. This function for routing protocols is usually the number of intermediate nodes through which the message forwards before it reached the destination. To maintain correct costs for links each node of the network periodically sends to all other nodes the costs of its outgoing links through flooding. It is one of the known forms of broadcasting. The source sends information to all its neighbors that are in communication range. The nodes, which have received the message, forward it to their neighbors and the process continues until the message has reached all network members. When the node receives the information via flooding, it updates its view of the network topology. After updating, the node applies a shortest path algorithm, which calculates the next hop on the shortest path to each node in the network [8].

In distance vector, routing information is transmitted only between directly connected nodes. Each member of the network periodically sends information to all its neighbors. Based on the received nodes start the algorithm of the shortest path to evaluate paths to the others. Thereby the node does not have accurate knowledge of the paths to the destination, however, it evaluates and determines to all other nodes the next hop in the path message to the destination [8].

Distance vector is more efficient in routes computing, requires less memory storage and consumes less network resources for the transmission of messages compared to link-state routing. However, this routing protocol can produce short-lived and long-lived loops. The main reason for their occurrence is that each node selects the next hop independently of the others and only on the basis of information it has. It should be noted that link-state routing can also generate loops. This is because the node may have wrong knowledge about the network topology, for example, due to the long propagation time of information through the network. Knowledge of outdated topology can lead to the creation of short-lived loops, which will disappear when the message will pass the network diameter.

Despite the fact that these two routing protocols are well researched and are in common use, both are not suitable for using in the ad hoc networks. The protocols show good performance in networks with low rate of topology changes, however, ad hoc network involves very frequent topology changes because of movement of network members. The main limitation is the fact that both are highly dependent on periodically sending control messages, which are necessary to maintain the relevant costs on the paths. Increasing the number of members of the network will increase the number of incoming and outcoming messages with routing information and it will be necessary to calculate a larger number of paths to the other nodes. Maintaining the relevant state of the network using these protocols will constantly consume resources such as bandwidth, computing power or energy. Thus the link-state routing and vector distance are not suitable for use in ad hoc networks, which imply high mobility of the network members.

The following describes the requirements that must be met for the routing protocol to be suitable for ad hoc networks.

The key of them is fully distributed operations. The protocol should not depend on a central control node. Ad hoc network differs from the stationary one that allows the nodes easily join or disconnect from the network, that is, the initial network can be divided into several networks. Due to high mobility of the nodes, the protocol should easily adapt to changes in network topology caused by movement of nodes. It also needs to be localized, since global exchange of routing information will require large overheads [7].

An important requirement is the absence of loops. The protocol should not contain them, otherwise the overall efficiency is reduced, the loop will spend more bandwidth and more computing power of each node in this loop. The protocol should take care about saving resources of network members. Because the members are the robots that have limited power, memory and computing power, so it is necessary not to conduct redundant operations. To achieve this goal it is important that the protocol is reactive. That is, the protocol does not periodically send control messages with routing information, but it acts when there is a need to convey the message. This approach will reduce the network overheads [7, 9].

In addition, the protocol needs to be scalable, this means that its effectiveness should not depend on the number of members in the network. Increase or decrease of the network should not affect its performance. It also needs to be adapted to any speed of nodes and any type of their movements. In conditions of high mobility the protocol must be able to quickly recalculate cost and build the path to the destination, which will include the fewest number of other nodes. While building the path and the forwarding of messages, it is necessary to effectively avoid outdated routes [9].

The following section contains classification of routing protocols based on the above requirements, which will outline possible approaches to data routing in the network. The main target is to select the routing protocol that is most suitable for teams of mobile robots, which establish a wireless Mobile Ad hoc Network (MANET) without any centralised structure.

1.3 Classification of Ad Hoc Routing Protocols

There are several classifications based on the routing protocols can be divided into groups.

One of the possible classifications is the division of routing protocols into centralized and distributed. In the first case, all route choices are made by a central control node. In the second case, the route calculation is distributed among members of the network. Distributed routing protocols are the most suitable for the purposes of MANET with a limited communication range.

Another classification is the categorization of routing protocols, depending on whether they change routes because of increasing congestion of certain routes. Static algorithms use the generated routes from source to destination regardless of traffic conditions. The route can be adjusted only by the refusal of the link or one of the nodes contained in the route. This type of algorithm will not provide huge throughput because it does not take into account the congestion of certain routes. The adaptive algorithm means that the route can vary depending on the load on some nodes or routes. In the case of MANET

Table 1.1: The examples of different routing protocols categories

Proactive	Destination Sequenced Distance Vector (DSDV) [10], Wireless Routing Protocol (WRP) [11], Global State Routing (GSR) [12], Fisheye State Routing (FSR) [13]
Reactive	Dynamic Source Routing (DSR) [14], Temporally Ordered Routing Algorithm (TORA) [15]
Hybrid	Zone Routing Protocol (ZRP) [16], Ad hoc On-Demand Distance Vector Routing (AODV) [17]

this classification is inessential, because the routes will be updated frequently in a rapidly changing topology and the traffic situation in the network will also change.

The main and most practical for ad hoc networks is the division into proactive (table-driven), reactive (on-demand) and hybrid, which combines advantages of the first two categories. The table 1.1 contains some examples of each category of routing protocols.

Proactive protocols constantly maintain the relevance of the routes between sources and destinations. For this reason, when there is a need to forward a message, the required route is already known and the message may be immediately transferred. To maintain a correct view of the network topology the protocol responds to every change in the structure by sending changes across the entire network. The protocol of this type periodically sends route update messages to its neighbors. Moreover, route updates are sent in the case that the network topology does not change. These protocols require to store routing information in one or more tables in each node [18].

Proactive protocols are not suitable for ad hoc networks, because route tables need to be refreshed at each change in network topology. This leads to excessive consumption of resources, which reduces network performance at high loads. Also, these protocols do not scale well and control overheads are proportional to the number of nodes in the network. Furthermore, they require a large memory capacity, as each node needs to store one or more tables [9].

An example of a proactive protocol can be Destination Sequenced Distance Vector (DSDV) [10], Wireless Routing Protocol [11], Global State Routing [12] or Fisheye State Routing [13]. In this thesis will be discussed in greater detail DSDV routing protocol.

Reactive protocols perform the process of route discovery only on request. The source floods the network with route query requests when a packet needs to be routed using distance vector routing or source routing.

In the first case, each node contains information about the active routes that are maintained while there is a need or not yet expired route lifetime, that prevents stale routes. Distance vector routing uses the address of the next hop and the destination to route packets. In the second case, the nodes do not need routing tables, because the source system adds routing information into a header of the data packet.

The problem of reactive protocols is the delay of packet transmission during the route discovery. On the other side, the route is discovered only when needed, i.e. generally it is less memory demanding compared to proactive protocols and requires relatively little control traffic overhead. Also on-demand protocols use flooding, to disseminate information over a network. This method is acceptable, because it is used only during a route discovery, however, it produces network overhead and redundantly uses bandwidth [18].

Dynamic Source Routing (DSR) [14] protocol or Temporally Ordered Routing Algorithm [15] are examples of reactive protocols. The first of these will be described in detail in this thesis.

Hybrid protocols typically try to reduce delay of route discovery from reactive systems by creating some form of routing tables and reduce control traffic overhead from proactive systems [18]. Thus, they combine positive aspects of both categories. Examples of hybrid protocol are Zone Routing Protocol [16] and Ad hoc On-Demand Distance Vector Routing (AODV) [17]. The last of these will be described herein.

1.4 Multi-robot communication

In this section, DSDV, DSR and AODV will be described in detail, which are representatives of three categories of routing protocols: proactive, reactive and hybrid, respectively.

1.4.1 Destination Sequenced Distance Vector Routing

DSDV was first described in [10] and it is an improved variant of distance vector routing protocol for use in ad hoc networks. This means that the routing protocol also uses the Bellman-Ford algorithm to calculate the minimum number of hops to the destination. As a result, each node maintains a routing table that contains entries for the next hop on the shortest path to all reachable destinations. Each of these entries stores the destination address, the address of the next hop on the path, the metric, which is defined as the minimum number of hops to reach the destination, and the sequence number received from the destination.

Sequence numbers that are assigned to the route, determine its freshness. When the node receives a message with the new information, it compares the sequence numbers of the new route with those already recorded in the routing table. New information is used only if the route has a more recent sequence number. If these numbers are equal, the route that contains a lower number of hops is chosen. All routes with older sequence numbers are no longer used. If new route information is recorded in the routing table, the metric is changed in comparison with the value from the incoming broadcast message, i.e. the number of hops to the destination will be increased by one. These routing updates are immediately planned to be sent to the neighbors.

Route sequences numbers are even, if they are generated by the destination and increase with the propagation of a new route. If this number is odd, it means that it was generated by any other node of the network and indicates a broken link. That is, if it is found that the link is broken, for example, because a neighbor does not send regular routing dump, this link is assigned an infinite metric and the sequence number is increased by one, becoming an odd number. Each node transmits information to its neighbors, so information about the broken link is spread across the network. The use of such a route numbering mechanism allows to avoid the major drawback of distance vector routing protocol, i.e. DSDV does not contain loops.

DSDV uses two types of dumps with updates to be more adapted to the conditions of use in ad hoc networks and to reduce network overhead. *A full dump* contains all available information about the routing information and it is sent at predetermined time intervals to all neighbors. The second type is called *incremental* and this dump is sent at the moment when a route has been changed and it is necessary to inform the other nodes in the network about its update. These packets are sent immediately and they contain only changes that have occurred since the last full dump.

Although incremental dumps are sent more frequently than full dumps, the periodic sending of complete routing information to all neighbors causes unnecessary overhead. Moreover DSDV is not the best for use in ad hoc networks, because in the conditions of frequent changes in the network topology, the nodes will be sending a lot of packets about broken links that need to be spread over the whole network. This propagation of information and frequent recalculation of optimal routes will increase the time required to build the error-free path for routing. There by, excessive overhead arises in the case that the network contains a large number of nodes, since it is necessary to store a routing table with all destinations and propagate a packet to all reachable nodes.

1.4.2 Dynamic Source Routing

Dynamic Source Routing (DSR) [14] is a reactive routing protocol, i.e. the route discovery to the destination starts only when necessary to deliver the data to this node. This routing protocol supports two types of operations: *route discovery* and *route maintenance*.

Mechanism of route discovery starts when the node is required to transfer the data, but the path to the destination is unknown. To start this process the node sends *a Route Request* (`RREQ`) packet to all its neighbors. It consists of the originator's address, the destination's address, a unique request identifier, which is determined by the originator, and contains the sequence of nodes that has already been passed and will be required to build a route for data transmission. After receiving a packet of this type each node first checks whether it is necessary to discard the packet for the case when the message with this identifier has already been processed. If this RREQ has not been processed, the node checks in its cache the presence of the route with corresponding destination. If a required route is not found, the node forwards the packet to its neighbors after adding its own identifier to the sequence of nodes. If a route to the requested destination has been found in the cache or the node is the packet's destination, the node sends *a Route Reply* (`RREP`) packet back to the originator using the transmitted sequence of nodes in reverse order.

During a route discovery, the originator saves a copy of the message in the send buffer. It contains copies of all packets that can not be sent due to the fact that there is no route to the destination. Each packet is stamped with time, when it was added to the buffer. At the expiration of a predetermined period of time the packet is discarded if it can not be transferred to the destination. For packets that are waiting in the send buffer, it is necessary occasionally to initiate a new route discovery. The rate of launching new route discoveries to the same destination should be limited, because the requested node may currently not be available. In this case, a large number of new RREQ will reduce network bandwidth. To solve this problem the originator uses an exponential back-off for situations when a new RREQ sends to the same destination. That is, if the originator's node tries to send a packet more frequently than the predetermined limit, then the messages are stored in the send buffer, until it receives a RREP. The node also can not initiate a new route discovery, until the minimum permissible time elapses between requests to the same destination.

After the route is built and the originator has received the RREP packet, the sequence of nodes from the packet's header is stored in the cache as the route to the requested destination. Because the packet contains the whole route, this principle is called source routing. Intermediate nodes use this sequence of nodes from the packet to determine to whom it should forward this message.

All the nodes store a sequence of nodes in their cache during forwarding that are contained in the packet's header. For example, the intermediate node caches the route to the originator while transmitting a RREQ or it caches the route to the destination while transmitting a RREP. Thus, route caching can speed up the route discovery, because the intermediate nodes can know the rest of the route to the destination, and reduce the spread of route requests.

The second type of operations in DSR is route maintenance. This mechanism allows to determine that the network topology is changed, for example due to the fact that the node is outside the data transmission range. That is, if it is not possible to send a packet to the next hop or if the RREQ was forwarded more than a predetermined maximum number of hops before reaching the destination, then this node sends *a Route Error* (`RERR`) packet to the originator.

This RERR means that the link is no longer available and the packet can't be delivered by the route. In this case, the originator will remove all routes from its cache which contains this link, then it will re-launch the process of route discovery.

The main advantage of DSR is complete absence of any periodic updates and a route building occurs only between the nodes that need to communicate. However, all nodes that are involved in the route discovery are stored routes from the transmitted packets, thus they increase its knowledge of the network topology. In the future it will help to reduce the time of route discovery. Route caching also allows to keep multiple alternative routes to the same destination.

However, the size of the transmitted packets increases with the length of the route, as each hop adds its identifier in the sequence of nodes in the packet's header. In the case of some network topologies, the propagation of route requests may potentially affect all the nodes in the network. Storing and transferring stale cached routes is also a disadvantage of DSR, because during the route discovery the node can use the stale route and send it as the RREP packet to the originator. During forwarding, intermediate nodes save this stale route in its cache, that in the future will increase the time of route building to some nodes.

1.4.3 Ad hoc On-Demand Distance Vector Routing

Ad hoc On-Demand Distance Vector (AODV) [17] routing is a routing protocol that combines the main advantages of DSDV and DSR. The main component is the distance vector algorithm, which is also used in DSDV. However, unlike the latter, AODV is reactive, i.e. builds routes only between those nodes that need to exchange messages. Moreover, if the originator node has a valid route to the destination in its routing table, the routing protocol is not used and the data is immediately sent.

AODV, like DSR, uses three types of messages: Route Request (RREQ), Route Reply (RREP) and Route Error (RERR). However, RREP packets are also used for sending periodic hello messages that broadcasts to the neighbors, within the range of data transmission. Hello messages can notify the neighbors that a node is available and all its neighbors will consider that the route to the node is valid. In the case the node has changed its position and it is no longer in the transmission area, then the neighbor will not receive hello messages from it and this node is marked as unavailable. If the node has become unavailable and the connection with it is broken, the neighbor sends a message about the link failure in the form of a RREP packet with infinite metric to a set of nodes that are taken from the route in the neighbor's routing table. Thus, AODV allows to track and quickly spread information about changes in the network topology.

In AODV, as in DSDV, each node builds its own routing table, but it does not have to contain routes to all available destinations in the network. AODV maintains only one route for each destination. The following information is stored for each route:

- *Destination IP Address*: the IP address of the destination;
- *Destination Sequence Number*: the latest sequence number, that was received in the past;
- *Route State*: current state of the route entry, for example, valid or invalid;
- *Hop Count*: number of hops needed to reach the destination;
- *Next Hop*: the neighbor, which has been designated to forward packets to destination for this route entry;
- *List of Precursors*: list of neighbors that are actively using this route entry. If there is a link failure, RREP packets with an infinite metric will be sent to these nodes;
- *Lifetime*: expiration or deletion time of the route entry.

This routing protocol supports two types of operations, like DSR. Route maintenance is provided by the use of the hello messages' mechanism and RREP packets with an infinite metric.

If there is a need to send a data, and this node does not have an active route to the destination, then the process of route discovery begins. For this, the originator sends a RREQ packet to its neighbors, which contains: destination IP address, originator IP address, destination sequence number, originator sequence number, hop count, broadcast identifier (ID).

AODV uses sequence numbers to ensure the freedom of loops, which appears in the routing protocols that use distance vector routing. The node uses its own sequence number as an originator sequence number, which is incremented before insertion in a RREQ and it is used in the destination's route table for a route entry to the originator. Also each node generates a broadcast ID that is incremented at the initiation of each new RREQ packet. Thus, each RREQ packet is uniquely identified by a pair of the originator's IP address and the broadcast ID. A RREQ packet also contains the most recent sequence number that has the node in its routing table for the destination. An intermediate node is allowed to answer to a RREQ packet only if the route to the appropriate destination from its routing table has the same or a higher sequence number than the corresponding number in the RREQ packet.

When a node receives a RREQ packet, it carried out a series of tests that determine whether the packet should be processed. First, the node verifies that the RREQ has not yet processed by it in the past, that is, the packet with the same pair of broadcast ID and originator IP address is not among the list of processed RREQs. If the packet has already been processed, then it is ignored. Otherwise, the node checks whether its own routing table has a valid and fresh route to the desired destination. That is, the route must be marked as active and the sequence number of the destination from the route entry should be not less than the destination sequence number of the RREQ packet. If the node has the required active route, it prepares and sends a RREP packet. If a suitable route is not found, it forwards the packet to its neighbors.

During packet transmission, each node stores the information about the neighbor in its routing table from which a RREQ was received. This entry later is used to build the reverse path. These route entries have a limited time of life slightly greater than the maximum time required for a packet to pass through the entire network. That is, during this time, the RREQ must reach the destination or the intermediate node with a required route and this node sends a RREP packet on the reverse path. Also each intermediate node increases the value of the hop count in the packet during forwarding process.

AODV uses together the principles of proactive and reactive approaches. Route between nodes is established only if it is necessary to send the data and there is no active route to the destination, i.e., the routing protocol reduces network overhead. At the same time, each node sends hello messages to its neighbors, that is, information about changes in the network topology quickly spreads, and after that, the nodes can mark the corrupted routes as invalid. RREQ packets in AODV are quite short, compared to a full or incremental dump, which are produced by DSDV, that is, this approach reduces the load on the network. A RREQ packet does not contain the complete sequence of nodes in the packet's header, that is an advantage compared to DSR. As a result, AODV is a routing protocol that compensates for shortcomings of DSDV and DSR, and maintains a balance between the size of the routing table in memory and the network overhead.

1.5 Multi-robot coordination

During each iteration of frontier-based exploration algorithm, the robot selects one of the frontiers as a goal to which it will move. After goal selection a problem of planning arises, i.e. the robot needs to make an obstacle-free path from its current position to the chosen goal. In the case when a team of robots is involved in exploration, each robot of the team needs to determine a path that does not only avoid obstacles, but also to avoid collisions with other robots. In the general case, any other robot is considered as a moving obstacle that has a predetermined size. Thus, multi-robot path planning problem is a problem of determining path without collisions from current position to selected goal in the environment for each robot of the team [19]. However, a general problem of optimal planning of multiple simultaneously moving objects is computational intractable, so the majority of the approaches achieves local optimization and it is less demanding on the overall optimization.

Multi-robot path planning techniques can be divided into *coupled (centralized)* and *decoupled.* The coupled technique considers a team of robots in the form of a composite robot system. Then a classical path-planning algorithm for a single robot is applied to the robot system. Using one of these algorithms in stationary environments is guaranteed to return optimal paths in polynomial time, but this problem in dynamic environments has an exponential computation time. That is, the computation time of the problem solution exponentially depends on a team size. This technique can only be used for teams with a small number of robots.

Decoupled approaches are divided into two categories: prioritized planning and path coordination. Prioritized planning scheme [20] assigns to each robot a unique priority value, which can be obtained either randomly or using some heuristics. Thereafter, paths towards goals are calculated in order of priority value of the robot. During path calculating all robots, which have the highest priority than the current node, are treated as moving obstacles. If one of the robots can not find a path to the goal, which is conflict-free with respect to robots with higher priority, then the whole algorithm fails, because this scheme does not use backtracking. Thus, a major role in the prioritized planning efficiency plays an initial prioritization of the robots in the team. The main advantage of the scheme is the fast calculation of routes in a changing environment.

On the other hand, path coordination scheme produces an independent calculation of the robots' paths, then conflicts are eliminated by adjusting a robot velocity in certain parts of the path.

1.5.1 Prioritized planning

There are several possible implementations of the scheme in practice [21]: *centralized prioritized planning* (CPP), *synchronized decentralized prioritized planning* (SDPP) and *asynchronous decentralized prioritized planning* (ADPP).

The first variant of implementation implies the existence of a centralized planner, to which the rest of the robot team reports its current location and the selected goal. The centralized planner prioritizes robots and calculates a solution that contains the path to each robot. Paths are computed by the general algorithm for prioritized planning, i.e. for each robot builds a path taking into account the paths of robots with the highest priority, which are represented as moving obstacles. Then the planner sends the paths to the robots, which are obliged to follow the chosen path.

The Synchronized Decentralized Prioritized Planning [22] algorithm is decentralized, that is, it does not use a centralized planner. Before running the algorithm, a unique priority is assigned to each robot in team. This implementation works in synchronized iterations. Each iteration of the algorithm begins with replanning. Each node itself calculates its own path from its current location to the destination and sends the generated path of the others. Then each node waits for the moment when all other robots will construct their own paths and send them to all other robots. When the node has generated its path and has received reports from all team members about their paths, then verification of the paths compatibility starts. If, during the check, a node has detected a conflict with a path of the node, which has a higher priority, then it conducts replanning of its own path and informs the others about a new path. The algorithm finishes when the robots could not find a solution without collisions or all nodes have ceased to communicate.

The advantage of SDPP is that the computation of the path is distributed between several robots. However, this implementation needs to know the number of nodes which are involved in planning to synchronize the iteration. This is not always realizable in terms of mobility nodes with a limited transmission range. In addition, while the robots do not find the collision-free solution for all, the robots idle. Such behavior inhibits the process of the environment exploration and causes the overhead of network resources.

Asynchronous Decentralized Prioritized Planning algorithm was suggested in [21]. This algorithm does not use iterations synchronization, which is needed in SDPP. Instead, ADPP uses a reactive approach, which forces a node to react only at the moments when it receives a special message called `INFORM` message. An INFORM message contains information about a generated path. Each node that receives an INFORM message and has a lower priority than the sender, takes into account the received path and checks that there are no conflicts between its current plan and the plans of the other team members. If a node can not generate a path to the goal, which will not cause collisions with the others, the node can sleep for a while. In addition, the process of replanning and the solution consistency checking are executed asynchronously. This means that if at the time of path re-planning a new INFORM message arrives, the planning process will be interrupted. Then the robot updates the knowledge about data paths of other nodes and the planning process starts anew.

ADPP algorithm saves computational resources, because the paths computation is distributed among all nodes. A robot does not wait for synchronization with other robots, thereby it reduces the overall time of terrain exploration. ADPP also reduces the load on the network, compared with SDPP, because robots only reports about its generated path and do not seek a common collision-free solution.

Chapter 2

Analysis and design

This chapter describes the requirements for communication between robots, a comparative analysis of various routing protocols and the analysis of software that will be established a messaging system in a distributed system. On the basis of the selected software the thesis will propose an algorithm to establish a connection and a pattern of message transmission between members of the network.

2.1 Design of multi-robot system communication

2.1.1 General requirements

As described in the section 1.1.1, a team of robots gets the benefit, if the robots transmit information between themselves, because exploration time decreases and efficiency increases. The application must be developed for future use in the team of real robots. The total number of robots in a team will not exceed ten pieces. However, as a simulator it should theoretically provide the possibility of collaboration of a few dozen of robots.

An important requirement is the ability to increase or decrease the number of robots in the a team during the already started exploration. This means that the robot will not know in advance the total number of other robots in the team and it needs to create connections during terrain exploration. This approach will allow to add new robots to the already running exploration to increase the speed of exploration, or to remove robots who, for example, ran out of energy.

One of the requirements is to minimize the time between sending a message and receiving it by the destination node. The robots are autonomous and can carry out a terrain exploration without the participation of other robots, but teamwork involves the exchange of information. Based on the obtained information, the robot can update its knowledge of the environment, reduce an unknown part of the terrain and to select new, more profitable goal for exploration. The sooner the robot gets this information, the faster it will process the data and reduce the unknown area.

In addition to minimizing delivery time of messages, it is important that messages reach the destination. That is one of the requirements is a high ratio of delivered messages. Message loss is not a problem, because the robots can carry out exploration independently, however, it may reduce the overall efficiency of the system.

The application must take into account the real-world conditions and meet the technical capabilities of real devices, because in the future, this application will be used as a basis for an application for the real robot team. It is assumed that each robot will be equipped with a wireless access point with predetermined limited transmission range. Therefore, the application should not excessively waste network resources. Moreover, it should not cause unreasonable data transfer and processing, because it requires a further computing means and it causes a higher energy consumption.

2.1.2 Message-Oriented Middleware

The application will use message passing. It is a technique for invoking behavior, which uses incoming messages from other processes to run a code. There are two types of message passing: synchronous and asynchronous.

Synchronous approach implies that message exchange takes place at a time when applications are running simultaneously. This approach is simple to implement, but has a significant disadvantage when used in distributed systems. A message sender is blocked until it receives a response from a message recipient. Therefore, the application that sent a message remains infinitely blocked if the recipient application is broken or it is outside the data transmission range. That is, synchronous approaches are not appropriate for use in distributed systems.

Asynchronous approach does not require simultaneous operation of a sender and a recipient. The sender sends a desired message, which is stored by an intermediate level of software. The sender is not blocked and it does not expect a response from the recipient, i.e. the application can continue its work. When the recipient is ready to receive data, it will get a message from the intermediate level of software. The most known type of such software is Message-Oriented Middleware (MOM).

2.1.3 ActiveMQ Apollo vs. ZeroMQ

This study examines two well-known Message-oriented middlewares: ActiveMQ Apollo and ZeroMQ. The table 2.1 provides a brief comparison of main parameters of these two asynchronous messaging libraries.

As can be seen from the table, both libraries are open-source and well-documented. The projects are working, both libraries shared a new stable

Table 2.1: A brief comparison between ActiveMQ Apollo and ZeroMQ

	ActiveMQ Apollo	ZeroMQ
Open-source	Yes	Yes
Has detailed documentation	Yes	Yes
Has new updates	Yes	Yes
Point-to-Point message passing	Yes	Yes
Publish/Subscribe message passing	Yes	Yes
Language binding for C/C++	Yes	Yes
Transport protocol	TCP, UDP	TCP, in-proc
Support brokerless model	No	Yes
Complexity of use	Requires broker configuration	Without pre-configuration

version for the past six months. Each project has a large active community of users.

MOMs usually can bind with many different programming languages. Since the application is developed using the framework, which is implemented in C++, then it is important that the application itself was written in C++. ActiveMQ Apollo and ZeroMQ have binding for C++.

There are two main functionalities of MOMs: message queuing and publish/subscribe. The first approach is a point-to-point messaging model, that is, a message is sent from a sender to a recipient. Publish/subscribe approach is a many-to-many messaging model. Thus, the message can be successfully distributed over a large distributed system. This approach works on the principle of radio, i.e. there is a publisher that creates a message and sends to a MOM and there are recipients that receive the messages from the MOM depending on their interests. Both libraries include point-to-point and publish/subscribe message passing.

Both libraries support a wide range of transport protocols, but for the purposes of the application the thesis will focus on Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and *in-proc.*

Both ActiveMQ Apollo and ZeroMQ support transfer of data over TCP, because it is the basic unicast transport protocol. It provides reliable and ordered delivery of a byte stream through applications running on hosts connected via Internet Protocol network. The protocol also provides verification that the message has been completely delivered to the recipient. Otherwise, it will be held a retry request for a missing part of the message.

ActiveMQ Apollo supports UDP, while ZeroMQ does not support it. UDP is not unlike TCP a reliable transport protocol, does not guarantee receiving messages in the right order, does not guarantee dublicate protection and does not control receipt of a message by a receiver. However, this protocol is lightweight and allows broadcasting. This means that a message can be sent to all accessible devices in a subnet with the use of UDP. Broadcasting is necessary for implementation of the hello message mechanism, which is used in Asynchronous Decentralized Prioritized Planning, described in the section 1.5.1. However, connection using UDP can be easily implemented using the standard sockets, i.e. without the use of the library.

The last transport is in-proc. It is an inter-thread transport, which is much faster than TCP and can transfer data between threads of a single application. It is a special-purpose unicast protocol created by the ZeroMQ library, which can be used to transfer data between threads of the application. An example of communication is shown in the Figure 2.1(d). This pattern can be used to organize communication between the different parts within the application itself.

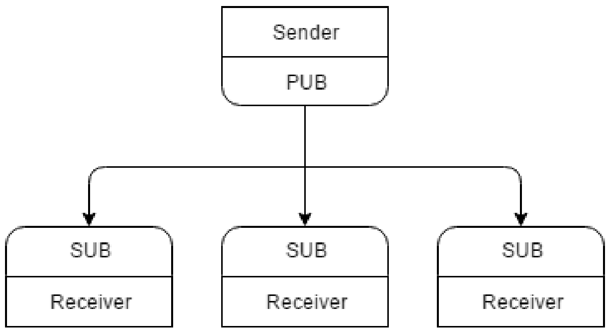
However, the most important difference is that the basic principle of ActiveMQ Apollo is to create and configure a broker. That is, it is necessary to create a centralized broker to transmit messages between the sender and the receiver. This concept is not suitable to implement a distributed system of robots, where everyone can contact each other. A distributed multi-robot system has no central node, which could be a broker. A possible solution is creation of a broker at each node of the network. However, this solution is cumbersome and requires excessive pre-configuration of brokers.

The ZeroMQ library was chosen for use in the application on the basis of the above comparison of the characteristics of both libraries. Because the library doesn't provide an implementation of UDP, the transport protocol was implemented using standard sockets.

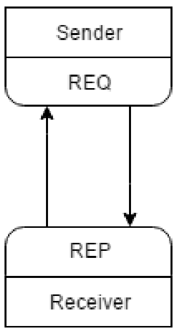
2.1.4 Comparison of ZeroMQ patterns

The ZeroMQ library offers several basic communication patterns [23] based on which it is possible to build the necessary interaction between members of the network. For the purposes of distributed terrain exploration it is necessary to create a system of autonomous robots. Each of them should be able to send messages to the robots, which are in its data transmission zone, and receive messages from them. Thus, each application must be able to send messages to multiple recipients and to receive messages from multiple senders. The sent messages may be of two types: *flooding*, which is sent to all accessible nodes in a network and *usual messages*, which is sent to a specific node. Messages of the latter type can be used by robots, e.g., for better goals selection for exploration.

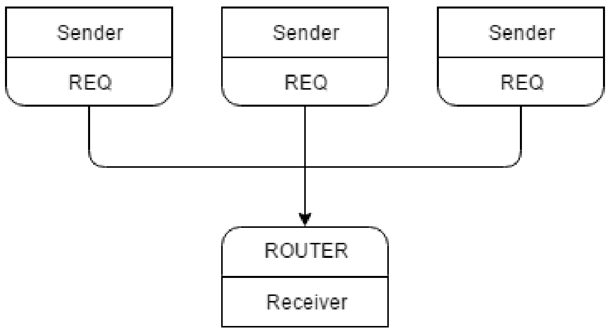
One of the basic patterns of multicast communication in ZeroMQ is *Publish-Subscribe* (`PUB-SUB`), which is represented in the Figure 2.1(a). In this pattern there is a publisher that works on the principle of radio. Subscribers connect to the publisher and set up a filter that selects only messages related to a given



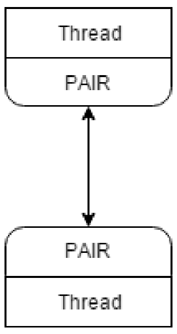
(a) PUB-SUB



(b) REQ-REP



(c) REQ-ROUTER



(d) Exclusive pair

Figure 2.1: Basic patterns of ZeroMQ library [23]

robot. So, in the projected application, each robot in a team can have its publisher, which sends messages to the others. In addition, each node creates multiple subscriptions to information from other publishers.

This pattern has significant drawbacks. Firstly, the publisher sends information to all subscribers, so there is a waste of network resources. Secondly, the robot as a subscriber can receive messages from a few tens of robots. In this case, the library fairly interleave data coming from different publishers. None of the publishers drowns out the others, however, the creation of a common queue of messages from different publishers will increase the time of receiving a response to a possible request from the message.

TCP and PUB-SUB pattern can cause problems with messages that will be collected in a queue at the publisher. This problem occurs because of the asynchronous work of PUB-SUB pair and the slow work of the subscriber who does not receive its messages. A possible solution is to use the *high-water mark*. By using this mechanism, messages will not be added to the queue, if it have reached the maximum allowable size. In this case, the message, that was ready to be sent, should be skipped or the publisher has to wait for the release of items in the queue.

Furthermore, a publisher does not track the actual receipt of a message by the recipient, the message is just given to the library for distribution across a network. In addition, PUB-SUB scheme has a symptom that is called *slow joiner* problem and causes a guaranteed loss of the first messages that were sent by the publisher during process of subscriber connecting. The problem is caused by the fact that making a TCP connection involves multi-step handshake process that takes several milliseconds depending on the network. During this time, the library manages to send a large number of messages that will never be received by the connecting subscriber.

Moreover, a publisher has no way of knowing about a fall of a subscriber or a cause of its fall. Also, a publisher has no information about whether the subscribers successfully connect to it. That is, the publisher doesn't know about their subscribers both during first connection and after restart due to a fall.

The next possible solution is to use the *Request-Reply* (`REQ-REP`) pattern or its extended version to receive messages from multiple senders - *Request-Router* (`REQ-ROUTER`). A pair of REQ-REP is shown in the Figure 2.1(b). It works in lockstep and resembles the work of procedure calls. It means that a sender sends a message when it is necessary to obtain certain information from a receiver. The recipient processes the message and returns a response to the sender. Then, the recipient continues to expect another call from the sender.

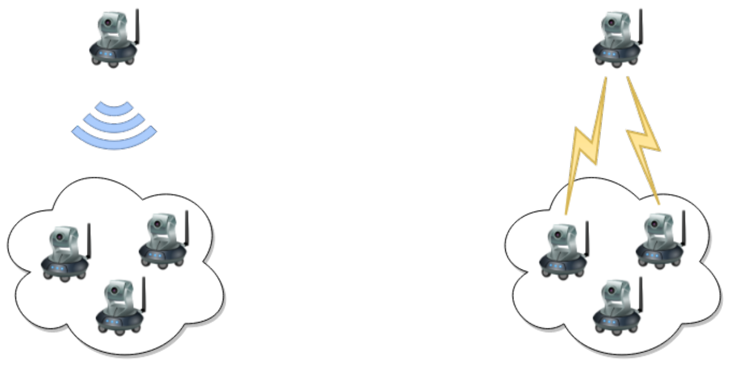
REQ-ROUTER pattern, which is represented in the Figure 2.1(c), is an extension of REQ-REP scheme. It creates an asynchronous server that can receive messages from multiple REQ senders. Unlike PUB-SUB, the scheme is reliable, because a server is obliged to respond to an incoming message. Also, there is the ability to send messages directly to required recipient.

For the designed multi-robot system, the most appropriate solution is to use each robot as a server which processes messages and sends responses on them. That is, the senders receive a current state of the server in addition to the requested information. According to the requests received from senders, a ROUTER also can receive information about the network topology.

2.1.5 Design of building connection among robots

REQ-ROUTER pattern involves the launch of a server that will accept requests for a specific socket and the launch of senders that will send requests to the selected socket. However, one of the requirements was that robots do not know in advance a number of other nodes in a team. Initially, a robot does not have any information about the other robots in the network, that is, it does not have the information necessary to create a connection. Also, other robots do not know about the existence of the robot, because it does not send any information.

Obtaining information about other robots occurs using the principle of *heartbeating.* A heartbeat is a periodic message that is sent to all available



(a) Heartbeating using UDP (b) Connection using ZeroMQ library

Figure 2.2: Connection establishment

devices. The principle allows to synchronize the different parts of distributed systems, and allows devices to tell the others about its presence. The implementation of the heartbeat uses UDP.

Figure 2.2 shows the principle of establishing a connection between autonomous robots. At first, a robot sends heartbeats to all reachable devices in current subnet via UDP. If there are robots in the data transmission range and they do not have connection to this robot, they use information obtained from heartbeat to connect to it using the ZeroMQ library. The reverse connection can be formed by the heartbeat, which is sent by the robot that has already installed one-way communication. The second possibility of creating reverse connection is to process information received by the router, which still has no connection with the message sender.

The robot can begin to transfer information to other nodes that are in the data transmission range after the connection. The routing protocol is used to transfer data to robots, which are outside this range. That is, the team of robots becomes a *mesh network*. It means that each node is a router and relays data between the network members. The next chapter contains the comparison and selection of routing protocol that is most suitable for terrain exploration by a multi-robot team.

2.1.6 Comparison of Routing Protocols

There are several criteria [24] for evaluating the performance of a routing protocol:

- *Packet delivery fraction* is calculated as the ratio between the number of received messages by the destination to the number of messages sent by the sender;

- *End-to-end delay* is the average time that elapses between the first packet which was transmitted by the sender before the first data packet received by the destination. The metric includes the time of route discovery, transmission delay, queuing delay and propagation delay;
- *Routing overhead* is calculated as the ratio between the routing packets to the total number of packets transmitted by the sender;
- *Throughput* is the amount of successfully delivered data via a communication link per time. It is usually measured in bits per second.

The article [24] discusses three routing protocol - DSDV, AODV and DSR. Testing is carried out on teams of robots, whose size ranges from 10 to 80 pieces. Robots perform a task similar to the task of terrain exploration. Each node chooses some point in the terrain and moves toward it. The article considers all four criteria.

DSR and AODV show better value of packet delivery fraction than DSDV. The value is about 95% for a team of 10 robots and it tends to 100% with an increase in the team size. AODV shows the smallest value of end-to-end delay. DSDV is a proactive routing protocol, so it shows better results than DSR. The value of routing overhead for AODV is better than DSDV. However, DSR has the smallest routing overhead in comparison with the other two. Throughput has the lowest value for proactive DSDV. Both DSR and AODV show approximately the same results, but this metric for DSR is slightly higher.

A performance comparison of AODV, OLSR and ZRP is described in [25]. The simulation is conducted for teams of robots with sizes of 25, 50, 75, 100, and provides measurement of all metrics, except the routing overhead. The packet delivery fraction for AODV is approximately 90%, while OLSR and ZRP have the values of 20-40%. Moreover, AODV has the highest throughput, as compared to two other routing protocols. However, AODV shows a significantly larger end-to-end delay than reactive OLSR and hybrid ZRP.

The publication [26] compares the performance of DSDV, TORA, DSR and AODV. The maximum number of robots in the team in the simulation is 50. The article considers several metrics, which include packet delivery fraction. Both DSR and AODV shows excellent results for the packet delivery ratio, which ranged from 95 to 100% depending on the network load.

AODV is selected for implementation on the basis of a comparison between DSDV, AODV, DSR, OLSR, ZRP and TORA. Simulations use robot teams of various sizes and display the measurement results that are close to the conditions of the real world. The main advantage of AODV is a high percentage of packet delivery to the destination. The routing protocol has a good throughput compared to the rest. The designed application does not send many unicast messages, so the inflated end-to-end delay and routing overhead are not important. That is, these values are acceptable for the terrain exploration problem.

Chapter 3

Realization

This chapter describes the general structure of the developed application and provides a detailed description of its components. Moreover, it contains description of the provided simulation software, which is extended with communication functionalities enabling decentralized coordination of a team of exploring robots. In addition, it highlights the problem encountered during the use of the ZeroMQ library and presents the solution to this problem.

3.1 Provided exploration framework

Application development is carried out on the basis of the framework for multi-robot terrain exploration provided by Intelligent and Mobile Robotics (IMR) Group from the Czech Institute of Informatics, Robotics and Cybernetics. Figure 3.1 shows the internal structure of the framework. It is a loop, which continues until the exploration is not finished. The framework works with all robots in the team at once.

Simulation of robot sensors uses a file that describes an unknown environment. The application generates a sequence of measurements, based on the current robot position and the map's file. This sequence is a laser scan.

The framework performs all operations for one iteration of the exploration process. Occupancy grid is used to find a set of possible frontiers for current iteration. Then, the framework uses an exploration strategy. It is the process that determines the next goal for the motion. The framework provides several possible strategies. Greedy strategy is chosen for the developed multi-robot application. This strategy has been proposed for use by the author of frontier-based exploration [2].

Then the framework plans a path from the current robot position in the environment to the selected goal. Path planning starts with inflating the obstacles in the grid by a radius of the robot, which prevents collisions of the robot with an obstacle. After that, the framework uses the grid to build a

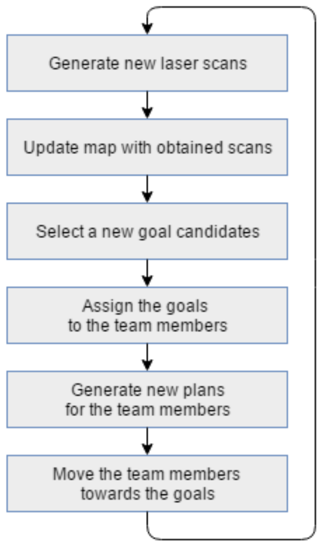

Generate new laser scans
Update map with obtained scans
Select a new goal candidates
Assign the goals to the team members
Generate new plans for the team members
Move the team members towards the goals

Figure 3.1: Framework structure

graph and runs a search algorithm on this graph to find the path. At the end, the generated path is smoothed for use in Euclidean space.

The framework has a visual component that allows to track the status of ongoing exploration. The visualization is performed using the Visualization ToolKit (VTK) library [27].

3.2 Application roles

The application has two possible roles: *Robot Application* and *Display Application*. Multi-robot terrain exploration is held by a homogeneous group of autonomous robots. Robot Application by using the provided framework performs all functions, which produces a real robot. That is, the application is able to obtain information about its surrounding area, to choose a new goal, to prepare a path to this goal and to move along the chosen path. Moreover, it contains functionalities providing cooperation with other team members.

Display Application allows to obtain information about the actual state of ongoing terrain exploration in the form of a map and to display it (see the Figure 1.2). This map is created from the data received by the application as a participant in the mesh network. That is, the Display Application is the router, but it does not carry out the exploration process. The map shows the part of the environment, which has become known on the basis of the scans from the Robot Applications. The application also shows its own position on the map, the position of the robots, their current goals and the paths to these goals. This information is retrieved from the transmitted messages. The application of this type in the real world can be used, for example, to obtain

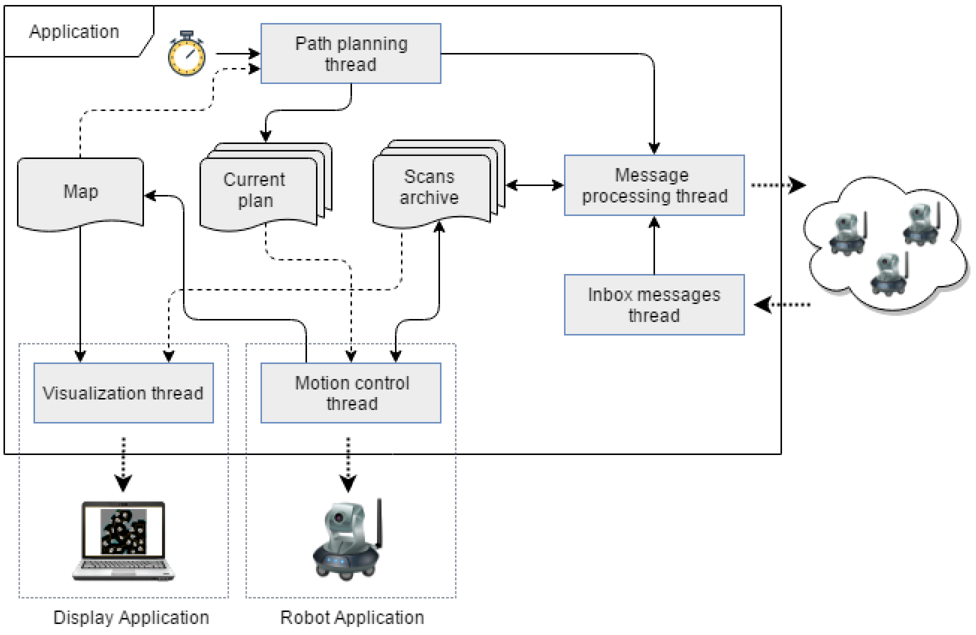


Figure 3.2: Application structure

information about the ongoing exploration taking place in a confined space that can be accessed only from the outside.

3.3 Realization of application logic

3.3.1 General structure

Figure 3.2 shows the general structure of the application, and describes the relationship between it and the external environment. Application operations have different frequency of launching and some of them can be called asynchronously. Plan generation runs one time of two seconds or by the request from the other robot. Robot produces movement in the environment every 100 milliseconds. The optimal frequency of sending the heartbeat is 1 second. Moreover, it is desirable to receive messages from the other robots instantly. Message sending is also necessary to make immediately to reduce the overall time of the terrain exploration. This means that the designed application could not be structured in a single loop, as it is done in the framework. So the application consists of 5 logical parts that are implemented as the threads: motion control/visualization, path planning, timer, message processing and inbox messages thread.

These parts of the application share the three data structures: map, current plan and scans archive. *Map* is the occupancy grid, which is an internal representation of the application's view about the environment. It accumulates the data received by the robot and the other team members. *Current plan* is the path to a goal selected at this iteration of the exploration process. *Scans archive* is a data structure that contains scans received from sensors of any robot of the team. That is, the structure stores local maps created by the robot and local maps created by other robots and transmitted via flooding. If the other robot requests a map, the application sends all accumulated scans to it from which the current representation of its knowledge about the environment can be built. The principle of the transfer of local maps between robots during the terrain exploration of a multi-robot team is described in [2].

The Robot Application role is determined by the using the *motion control thread* to simulate the robot operation. It is responsible for receiving the current plan. Moreover, it sends signals to the robot motors for movement to the next selected goal. It also uses the robot sensors to obtain scans, which are then stored in the scans archive. The motion control thread receives from the scans archive new scans received from other robots. Then the thread conducts a map update on the basis of these data.

If the application is launched for the purpose of visualization of the exploration process, it uses a *visualization thread.* The application of this type does not create its own scans, so the scans archive contains only local maps received from the robots. This thread uses them to update the map, which is displayed on the screen after that.

The path planning thread is the main component of each iteration of the terrain exploration process. This thread selects the next goal for the motion and makes the path to it. This plan is written to the shared structure. The thread uses the map to plan.

The application uses a timer to generate the plan after a certain predetermined period. The timer is located in the *timer thread.*

The message processing thread is the main connection between the robot and the rest of the team. It is responsible for sending messages to other robots about a new generated path or a new laser scan. It also receives messages from the inbox messages thread.

The inbox messages thread deals with receiving messages from other team members and sending them in the messages processing thread.

Hereinafter, each thread will be described individually. Each described operation of the thread has its triggering frequency. That is, the operation is not obliged to carry out during each iteration of the loop. Launch frequency is defined manually and it creates the opportunity to simulate the real work of the team of robots.

3.3.2 The motion control thread

Scheme of the motion control thread is shown in the Figure 3.3(a). It is an loop, which terminates when a signal from the path planning thread is received on completion of the terrain exploration. First, the motion control thread calls

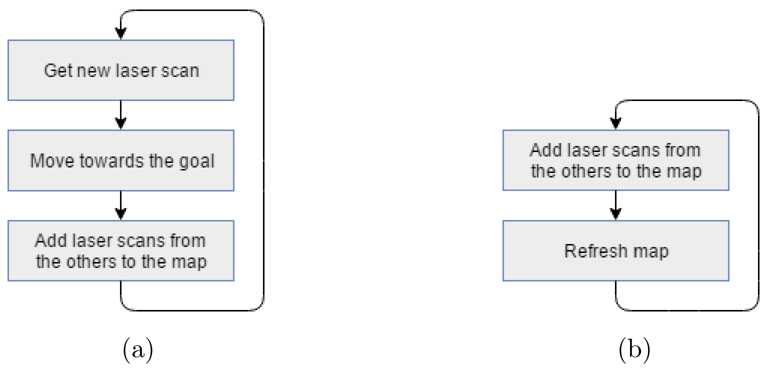

Get new laser scan
Move towards the goal
Add laser scans from the others to the map
(a)
Add laser scans from the others to the map
Refresh map
(b)

Figure 3.3: Scheme of: (a) the motion control thread; (b) the visualization thread

the framework function, which returns a laser scan on the base map and the current robot position. Data obtained from the robot sensors are added to the occupancy grid. Then the thread receives the current plan from the shared structure. It uses its engines to move to the first point of this plan. After that, the thread gets new scans that are received from other team members, and adds them to the occupancy grid. Updating the map by its own or another's scan is the same function of the framework.

3.3.3 The visualization thread

Visualization thread is an infinite loop with the possibility of forced exit. The Display Application itself does not produce the terrain exploration, that is, the application task is only to observe the work of the robots team. The algorithm of the thread is shown in the Figure 3.3(b).

First, the thread update the map with the scans that are obtained from the others. The map is refreshed then and it is displayed using the VTK library. A visual representation of robots is extended with a possibility to display the current plan, which follows the robot. For easier identification of the robot, its unique identifier displays near the point, which represents it on the map. The thread also has the tools to save screenshots of the map.

3.3.4 The path planning thread and the timer thread

Figure 3.4(a) shows the scheme of the path planning thread. It uses the framework function to generate the plan. This function uses the map to determine the set of frontiers, selects the next goal from them and builds a path to the selected goal. The function is modified to take account of the goals derived from the other robots. That is, the application does not choose as a goal a zone, which contains the goal of another robot. At the moment the radius of the zone is defined by the laser range. It provides the non-overlapping areas

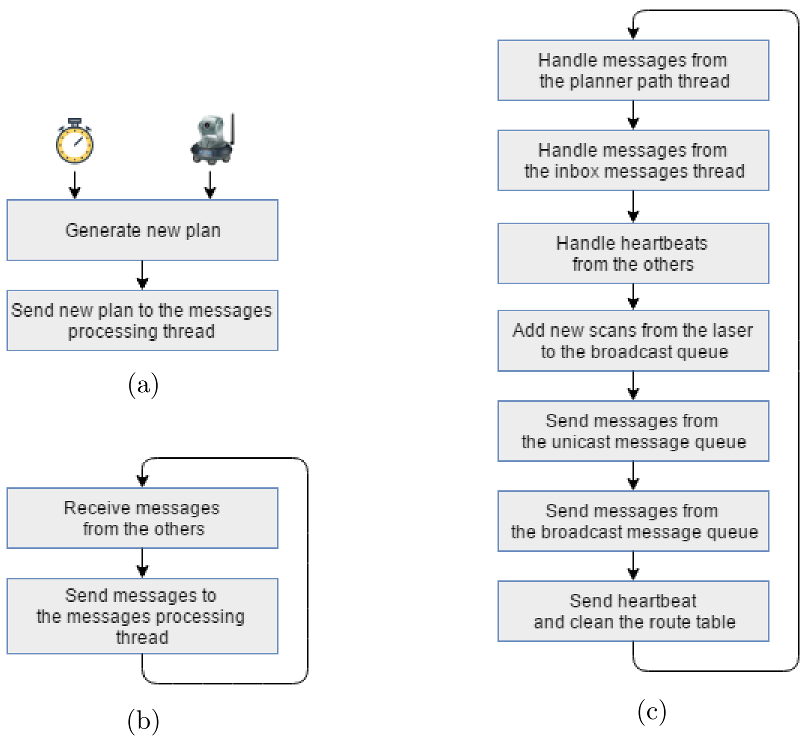

Generate new plan
Send new plan to the messages processing thread
(a)
Receive messages from the others
Send messages to the messages processing thread
(b)
Handle messages from the planner path thread
Handle messages from the inbox messages thread
Handle heartbeats from the others
Add new scans from the laser to the broadcast queue
Send messages from the unicast message queue
Send messages from the broadcast message queue
Send heartbeat and clean the route table
(c)

Figure 3.4: Scheme of: (a) the path planning thread; (b) the inbox messages thread; (c) the message processing thread

during the exploration. The application prefers a zone where there are no other team members goals. The generated plan is stored in the internal structure of the application and it is sent to the message processing thread to notify other robots about its current goal.

The planning process runs periodically. The timer thread includes a timer with pre-set frequency of invocation of the planning process.

If two robots have goals in the same area, they compares the priority of their goals, which is calculated as the distance from the current position to the selected goal. If the goal priority of the other robot is higher, then the robot starts the unscheduled re-planning, which is called from the message processing thread.

3.3.5 The message processing thread and the inbox messages thread

The messages processing thread integrates the work of the current application and the team members' applications. It uses the inbox messages thread for data receiving (see the Figure 3.4(b)). The inbox messages thread receives messages from other robots and forwards them for processing. The separation of incoming and outcoming messages is done for quickly retrieving messages from the others. Otherwise, the sender accumulates all unreceived messages in its queue, which is supported by the ZeroMQ library.

Figure 3.4(c) shows the operation sequence within the thread. First, the thread carries out the processing of messages from the path planning thread. New generated plan broadcasts to all neighbors.

This thread handles any messages from the other robots. The application supports several types of messages. The `SCAN` messages contain a laser scan, which is received by the sensors of another robot. The `NEED_MAP` messages informs about the need to send the entire scans archive to the message sender. The `MAP` messages contain the complete scans archive, which is accumulated by the message sender. The `GOAL` messages report about the current goal of the sender. The incoming `CHANGE_GOAL` messages force to re-plan the current path.

The thread handles incoming heartbeats from the other robots. The application is updated its routing table based on these heartbeating messages. Also, this thread periodically sends own heartbeat using standard sockets and UDP.

Outcoming messages are accumulated in two queues. The first queue is a queue for the unicast messages. If the routing table does not contain the route to the necessary destination, the thread starts the route discovery by AODV routing protocol. The message is stored in the queue until it finds a route to the destination or the message lifetime expires. The second queue broadcasts messages to all active neighbors.

3.4 The problem with fault tolerance of the ZeroMQ library

Implementation of a general structure with the use of the library ZeroMQ clearly shows the problem with fault tolerance. Since the REQ-ROUTER pattern is selected to implement communication between the robots, the sender waits on a response from the ROUTER after sending a request. The ZeroMQ library provides asynchronous approach at intermediate level, that is, in the case of recipient falling, the library expects its recovery. However, the robot's disabling with high probability means the end of energy, i.e. the robot is switched off permanently, but the sender continues to expect the destination recovery.

ZeroMQ does not provide the possibility to obtain information about the current status of the recipient before sending a message. Moreover, the library does not allow to set a timeout on an attempt to send a message. Thus, in case of a fall, the whole system of robots gradually stops because of waiting a response from each other.

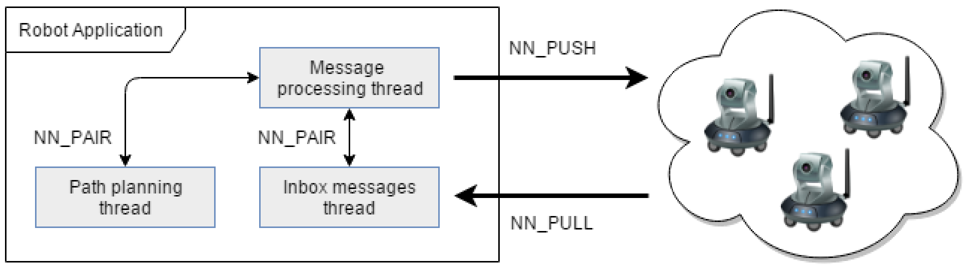


Figure 3.5: The general scheme of communication with the use of the nanomsg library

Various advanced patterns designed for ZeroMQ to avoid problems with the device failure have been studied to solve the problem. The search for alternative libraries was also produced. So the use of nanomsg [28] library becomes the solution to the problem. This library was developed by one of the key creators of the ZeroMQ libraries and it fixes most of the ZeroMQ drawbacks.

The nanomsg library provides several basic communication patterns, among which appears a `PIPELINE`. This pattern allows to collect messages from many senders and to distribute the load between multiple destinations. The pattern is used in the application to collect messages from multiple senders. It is important to note that this pattern does not require confirmation from the destination, unlike the REQ-ROUTER pattern in ZeroMQ. PIPELINE consists of two types of sockets: `NN_PUSH`, which allows to send a message and `NN_PULL`, which allows to receive a message. As shown in the Figure 3.5, the messages processing thread uses the NN_PUSH type of socket to send messages to the other team members, while the NN_PULL type of socket is used for receiving messages from the others.

Moreover, the library allows to set a timeout for an operation of sending messages. It allows to refuse the message sending, if it is not sent for some predetermined amount of time.

Communication between threads within the application is based on the bidirectional one-to-one communication pattern called `PAIR`, which uses the `NN_PAIR` type of socket on both endpoints. Thus, the internal communication continues to use the same pattern. Transport mechanisms also remain the same. The application uses inproc transport for internal communication and TCP to send messages between the robots.

Chapter 4

Experiments

This chapter experimentally evaluates behavior and properties of the implemented application, which is described in detail in the Chapter 3. An additional purpose of the chapter is to compare the characteristics of the work of a single robot and a multi-robot team during the exploration of an unknown terrain.

4.1 Conditions

Experiments were conducted using a device which has 32 Central Processing Units (CPU) and has 8 GiB of memory. Computational resources were provided by the CESNET LM2015042 and the CERIT Scientific Cloud LM2015085, provided under the program "Projects of Projects of Large Research, Development, and Innovations Infrastructures".

All tests were performed on the map of an empty terrain that has a size of 20 by 20 meters. Each cell of the occupancy grid has a size of 5 by 5 centimeters. It means that the occupancy grid has a size of 400 by 400 cells. Various numbers of robots in the team are used to test the loading of the network. The team size is varied from one to ten. The application makes the terrain exploration as a single robot, if the team consists only of one robot. Various ranges of communication links are used to test the effectiveness of teamwork. The communication range takes one of the values: 5, 10, 15 or 20 meters. The laser has a range of 1.5 or 2 meters. It allows to verify the functionalities performance with a larger number of steps of the exploration process on the same map.

The experiments were performed three times for each system configuration. The attached CD contains the statistics that are collected during the conducted experiments. All graphs in this chapter are based on these data.

The time measurement of the exploration is made in *steps*, i.e the number of calls of the function that changes the robot position. This function is triggered periodically after a certain period of time. Thus, a step is a con-

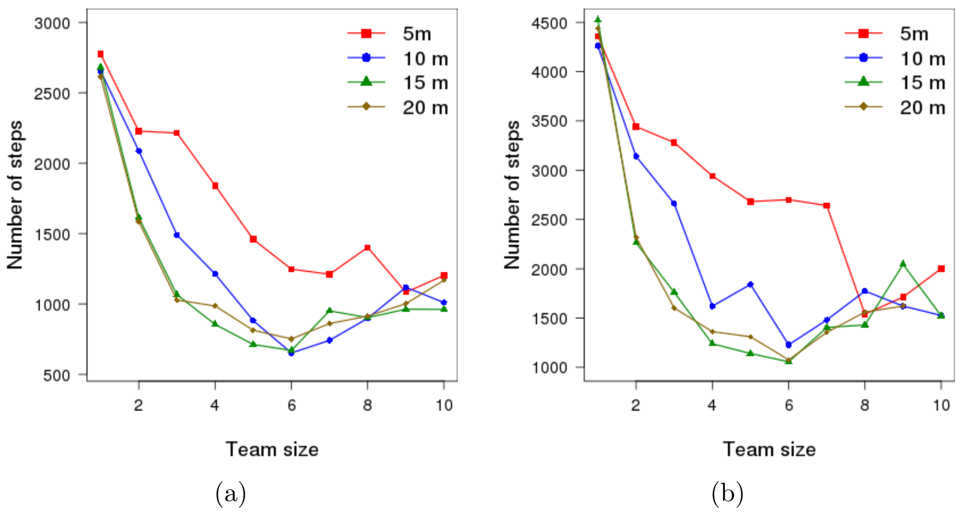


Figure 4.1: The dependence of the steps from the team size. The value of the laser range is: (a) 2 metres; (b) 1.5 metres

venient alternative to time. In addition, the usage of steps metric allows to maintain usable statistics of events that pass between sequential steps.

4.2 Time of the exploration

The Figure 4.1 shows the dependence of the average exploration time for a team member on the number of robots in the team. The graph shows that any team work helps to reduce exploration time. In the beginning, both graphs show a reduction of the number of steps, then there is an increase in steps quantity. This is due to the fact that the robots have to excessively agree on the choice of a goal in bigger teams. As can be seen from the Figure 4.2, a small team disperses at the beginning of the exploration and explores non-overlapping areas of the terrain. However, an excessive number of robots in the team causes the problem of determination of the non-overlapping areas. So robots are required to negotiate the rescheduling of their goals.

The graph also shows that the decrease in the communication range increases the number of steps that are required to complete the terrain exploration. Some scans from the other robots could not reach the robot because of lack of communication link due to the small value of the communication range. That is, the robot is forced to explore the given area by itself. It increases the overall time of the exploration.

Comparison of the Figure 4.1(a) and the Figure 4.1(b) shows that the ratio of the steps number and the team size remains constant regardless of the laser range. However, reducing the value of laser range increases the total number of steps. If the laser has a shorter range, then each scan provides less information

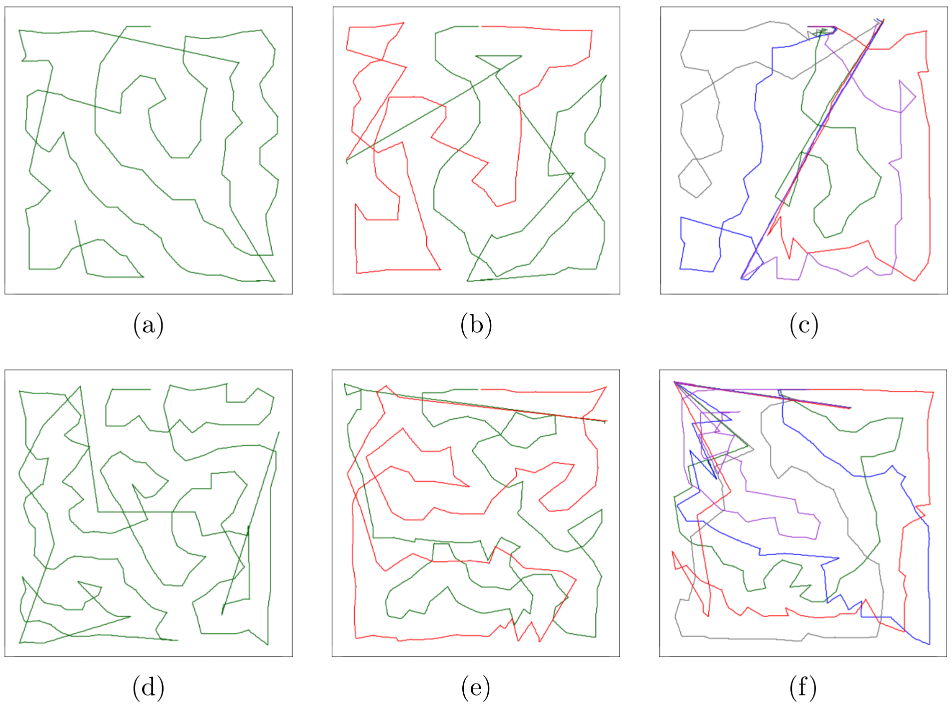


Figure 4.2: The result paths after the exploration. The laser range of the first row is 2 meters, the second row is 1.5 meters. Columns are separated by team size: (a)+(d) 1 robot; (b)+(e) 2 robots; (c)+(f) 5 robots

about the environment. Thus, it is necessary to make larger movements and get more scans to obtain a complete map of the terrain.

4.3 Plan generations

The Figure 4.4 shows the frequency with which the application is launching a new request for a new generation of the plan. A plan generation starts after a certain time, as described in the previous chapter. The Figure 4.4(a) shows that a single robot starts a process of path generation by approximately the same intervals. The increase of the team size leads to the fact that robots can compete for the goals in a particular region of the terrain. This can have both positive and negative effects.

Let's consider the situation of the terrain exploration by a team that consists of ten robots and the communication range is 20 meters. This means that all robots agree on the choice of goals with the other robots. The first low part of the Figure 4.4(d) corresponds to the Figure 4.3(a). Robots start the goal selections. However, the current implementation of the goal choice requires to select the goal that does not lie close to the goals of other team members. This

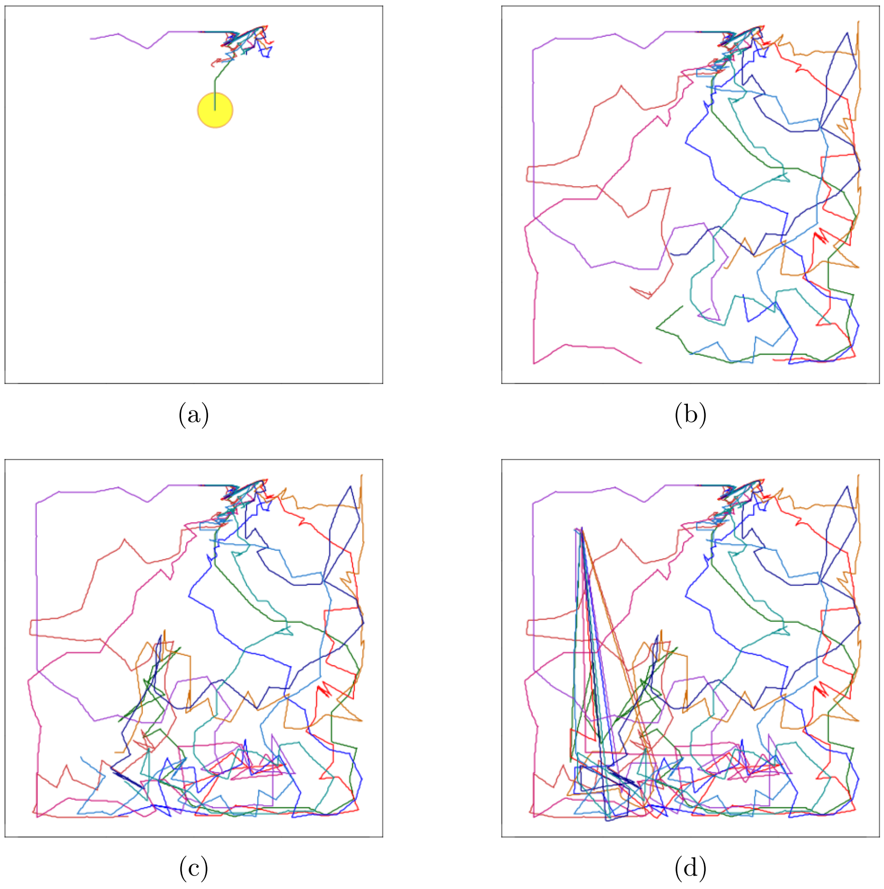


Figure 4.3: The passed paths for the team of 10 robots and the laser range=20 meters at the step: (a) 80; (b) 580; (c) 800; (d) after finishing exploration

means that the distance between the selected goal and goal of another robot should be greater than the laser range. An example of such area is depicted in the Figure 4.3(a) as the yellow circle.

If each robot chooses a goal which does not overlap with the goals of the other robots, the plan generation is invoked periodically by the timer thread. That is, the robot does not encounter other robots and it made the right goal choice. The straight part of the plot with a step value, which is equal to 20, corresponds to the Figure 4.3(b).

Then, most of the terrain is explored and the robots are in a small unexplored part of the environment (see the Figure 4.3(c)). The weak point of the implemented simple prioritizing between the robots appears. At this point, the frequency of plan generation begins to increase.

Moreover, increasing the number of plan generations leads to increase in the number of steps that are necessary to complete the terrain exploration. It

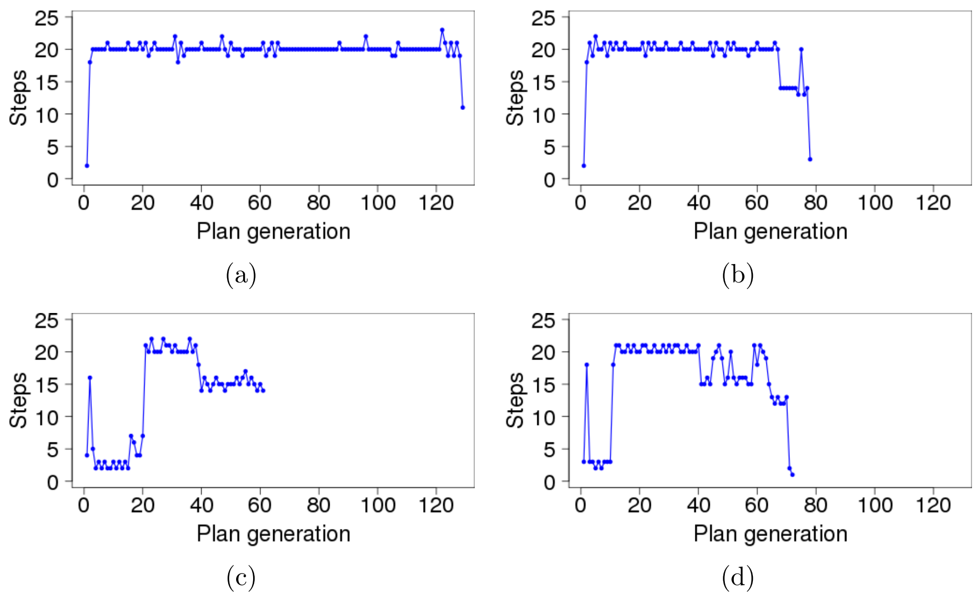


Figure 4.4: The frequency of plan generations for an average robot from the team that consist of: (a) 1 robot; (b) 2 robots; (c) 5 robots; (d) 10 robots

then leads to increase of the total exploration time. Thus, the lengths of the graphs in the Figure 4.4 correspond to the graph from the Figure 4.1(a) for points with the values of the team size equal to 1, 2, 5 or 10.

Conclusion

This bachelor's thesis deals with the problem of the decentralized exploration of an unknown environment. Formulation of the problem was described at the beginning of the Chapter 1. The benefits of a team of robots to one robot to solve the problem have been described then.

Teamwork requires the establishment of communication links between particular robots. The thesis describes in detail the ad hoc networks that have no pre-determined structure and are suitable for the establishment of a decentralized system. The work gives a list of requirements to the routing protocol, which allows to maintain connectivity between nodes of the ad hoc network. The division of routing protocols into different categories was described. DSDV, DSR and AODV routing protocols were considered in detail and were described their positive and negative sides.

In addition to communication, teamwork requires coordination. The thesis discusses different types of approaches, special attention was paid on the prioritized planning. Possible implementations were compared to implement this scheme. The result of the comparison is the choice for the implementation of the algorithm of Asynchronous Decentralized Prioritized Planning.

The Chapter 2 focuses on the technical aspect of communication between nodes of a distributed system. The beginning of the chapter describes the requirements that are imposed on the relationship between the nodes in the network. Message-Oriented Middleware was used for providing the messages passing functionalities. The thesis presented a comparative analysis of two most famous representatives of the MOMs. The ZeroMQ library was chosen as the most suitable for implementation on the basis of this comparison results. The analysis of possible configurations of communication between robots using the communication patterns of the selected library was carried out. The process of establishing communication between the robots was also designed. At the end of this chapter, AODV was chosen as the most appropriate routing protocol for real communication in distributed multi-robot systems.

The Chapter 3 describes the structure of the realized application, which uses the MOM to transmit messages and AODV as a routing protocol, and it gives a detailed description of each part of the developed application. Each part of the developed application was described in detail. The library ZeroMQ has shown to be ineffective in the problem with fault tolerance during the process of the application implementation. Upon further review, the application started to use the nanomsg library. The scheme of communication between robots has also been revised to reflect the use of the new library.

The Chapter 4 focuses on the conducted experiments and their results. They showed that the terrain exploration by a multi-robot team allows to decrease the total time of the exploration. The developed application showed less exploration time in the case of choosing the optimal number of robots in the team.

In the future, this application will be used for testing on the group of real robots that solves the problem of the exploration of an unknown environment. Goal allocation was not addressed in the thesis as it was out of scope of it. Instead, a simple method to goal allocation was employed to demonstrate that the realized approach to multi-robot coordination and communication is feasible. More sophisticated goal allocation mechanisms can be developed and incorporated into the presented framework, which will be another stream we would like to go in future.

Bibliography

[1] Kulich, M.; Přeućil, L.; Bront, J. J. M. Single robot search for a stationary object in an unknown environment. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, ISSN 1050-4729, pp. 5830–5835, doi:10.1109/ICRA.2014.6907716.

[2] Yamauchi, B. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, Jul 1997, pp. 146–151, doi:10.1109/CIRA.1997.613851.

[3] Moravec, H.; Elfes, A. High resolution maps from wide angle sonar. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, Mar 1985, pp. 116–121, doi:10.1109/ROBOT.1985.1087316.

[4] Stepan, P.; Kulich, M.; Preucil, L. Robust data fusion with occupancy grid. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, volume 35, no. 1, Feb 2005: pp. 106–115, ISSN 1094-6977, doi:10.1109/TSMCC.2004.840048.

[5] Fox, D.; Ko, J.; Konolige, K.; et al. Distributed Multirobot Exploration and Mapping. *Proceedings of the IEEE*, volume 94, no. 7, July 2006: pp. 1325–1339, ISSN 0018-9219, doi:10.1109/JPROC.2006.876927.

[6] Yamauchi, B. Frontier-based Exploration Using Multiple Robots. In *Proceedings of the Second International Conference on Autonomous Agents*, AGENTS '98, New York, NY, USA: ACM, 1998, ISBN 0-89791-983-1, pp. 47–53, doi:10.1145/280765.280773. Available from: `http://doi.acm.org/10.1145/280765.280773`

[7] Maqbool, B. B.; Peer, M. Classification of current routing protocols for ad hoc networks-a review. *International Journal of Computer Applications*, volume 7, no. 8, 2010: pp. 26–32.

[8] Peterson, L. L.; Davie, B. S. *Computer Networks, Fifth Edition: A Systems Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., fifth edition, 2011, ISBN 0123850592, 9780123850591, 243–258 pp.

[9] Ehsan, H.; Uzmi, Z. A. Performance comparison of ad hoc wireless network routing protocols. In *Multitopic Conference, 2004. Proceedings of INMIC 2004. 8th International*, Dec 2004, pp. 457–465, doi:10.1109/INMIC.2004.1492924.

[10] Perkins, C. E.; Bhagwat, P. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *SIGCOMM Comput. Commun. Rev.*, volume 24, no. 4, Oct. 1994: pp. 234–244, ISSN 0146-4833, doi:10.1145/190809.190336. Available from: `http://doi.acm.org/10.1145/190809.190336`

[11] Murthy, S.; Garcia-Luna-Aceves, J. J. An Efficient Routing Protocol for Wireless Networks. *Mob. Netw. Appl.*, volume 1, no. 2, Oct. 1996: pp. 183–197, ISSN 1383-469X, doi:10.1007/BF01193336. Available from: `http://dx.doi.org/10.1007/BF01193336`

[12] Chen, T.-W.; Gerla, M. Global state routing: a new routing scheme for ad-hoc wireless networks. In *Communications, 1998. ICC 98. Conference Record. 1998 IEEE International Conference on*, volume 1, Jun 1998, pp. 171–175 vol.1, doi:10.1109/ICC.1998.682615.

[13] Pei, G.; Gerla, M.; Chen, T.-W. Fisheye state routing: a routing scheme for ad hoc wireless networks. In *Communications, 2000. ICC 2000. 2000 IEEE International Conference on*, volume 1, 2000, pp. 70–74 vol.1, doi:10.1109/ICC.2000.853066.

[14] Johnson, D. B.; Maltz, D. A. Dynamic source routing in ad hoc wireless networks. In *Mobile computing*, Springer, 1996, pp. 153–181.

[15] Park, V. D.; Corson, M. S. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*, INFOCOM '97, Washington, DC, USA: IEEE Computer Society, 1997, ISBN 0-8186-7780-5, pp. 1405–. Available from: `http://dl.acm.org/citation.cfm?id=839292.843010`

[16] Haas, Z. J. A new routing protocol for the reconfigurable wireless networks. In *Universal Personal Communications Record, 1997. Conference Record., 1997 IEEE 6th International Conference on*, volume 2, Oct 1997, ISSN 1091-8442, pp. 562–566 vol.2, doi:10.1109/ICUPC.1997.627227.

[17] Perkins, C.; Belding-Royer, E.; Das, S. Ad Hoc On-Demand Distance Vector (AODV) Routing. 2003.

[18] Alex Hinds, S. Z., Michael Ngulube; Al-Aqrabi, H. A Review of Routing Protocols for Mobile Ad-Hoc Networks (MANET). *International Journal of Information and Education Technology*, volume 3, no. 1, Feb 2013: pp. 1–5, ISSN 2010-3689.

[19] Parker, L. E. *Encyclopedia of Complexity and Systems Science*, chapter Multiple Mobile Robot Teams, Path Planning and Motion Coordination in. New York, NY: Springer New York, 2009, ISBN 978-0-387-30440-3, pp. 5783–5800, doi:10.1007/978-0-387-30440-3_344. Available from: `http://dx.doi.org/10.1007/978-0-387-30440-3_344`

[20] Erdmann, M.; Lozano-Pérez, T. On multiple moving objects. *Algorithmica*, volume 2, no. 1, 1987: pp. 477–521, ISSN 1432-0541, doi:10.1007/BF01840371. Available from: `http://dx.doi.org/10.1007/BF01840371`

[21] Cáp, M.; Novák, P.; Selecký, M.; et al. Asynchronous decentralized prioritized planning for coordination in multi-robot system. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, ISSN 2153-0858, pp. 3822–3829, doi:10.1109/IROS.2013.6696903.

[22] Velagapudi, P.; Sycara, K.; Scerri, P. Decentralized prioritized planning in large multirobot teams. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, Oct 2010, ISSN 2153-0858, pp. 4603–4609, doi:10.1109/IROS.2010.5649438.

[23] Hintjens, P. *ZeroMQ: Messaging for Many Applications*. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, first edition, 3 2013, ISBN 978-1-449-33406-2, 81–310 pp.

[24] Kumar, S.; Singh, D.; Chawla, M. Performance Comparison of Routing Protocols in MANET Varying Network Size. *International Journal of Smart Sensors and Ad-hoc Networks (IJSSAN) ISSN*, , no. 2248-9738, 2011: pp. 51–54.

[25] Ahuja, R. Simulation based performance evaluation and comparison of reactive, proactive and hybrid routing protocols based on random waypoint mobility model. *International Journal of Computer Applications (0975–8887) Volume*, 2010.

[26] Broch, J.; Maltz, D. A.; Johnson, D. B.; et al. A Performance Comparison of Multi-hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom '98, New York, NY, USA: ACM, 1998, ISBN 1-58113-035-X, pp. 85–97, doi:10.1145/288235.288256. Available from: `http://doi.acm.org/10.1145/288235.288256`

[27] Avila, L. S. *The VTK user's guide*. Clifton Park, N.Y.: Kitware, 11th edition, 2010, ISBN 9781930934238.

[28] Sustrik, M. The nanomsg library. 2015. Available from: `http://nanomsg.org/documentation.html`

Appendix A

Acronyms

2D	Two Dimension(al)
ADPP	Asynchronous Decentralized Prioritized Planning
AODV	Ad hoc On-Demand Distance Vector
CPP	Centralized Prioritized Planning
CPU	Central Processing Units
DSDV	Destination Sequenced Distance Vector
DSR	Dynamic Source Routing
MANET	Mobile ad hoc network
MOM	Message-Oriented Middleware
OLSR	Optimized Link-State Routing
SDPP	Synchronized Decentralized Prioritized Planning
TCP	Transmission Control Protocol
TORA	Temporally Ordered Routing Algorithm
ZRP	Zone Routing Protocol
UDP	User Datagram Protocol

Appendix B

Contents of enclosed CD

```
 README.............................the file with CD contents description
├── packages..................the directory with source of external libraries
├── src.......................................the directory of source codes
│   ├── robotApplication..........................implementation sources
│   └── thesis..............the directory of LaTeX source codes of the thesis
└── text.........................................the thesis text directory
    ├── BP_Khilda_Slyusar_2016.pdf.........the thesis text in PDF format
    └── BP_Khilda_Slyusar_2016.ps............the thesis text in PS format
```