Acta Polytechnica CTU Proceedings **6**:18–27, 2016
doi:10.14311/APP.2016.6.0018

available online at http://ojs.cvut.cz/ojs/index.php/app

FRAMEWORK FOR AD HOC NETWORK COMMUNICATION IN MULTI-ROBOT SYSTEMS

Khilda Slyusar*, Miroslav Kulich

Czech Technical University in Prague, Zikova 1903/4, 166 36 Prague, Czech Republic

* corresponding author: slyuskhi@fel.cvut.cz

Abstract. Assume a team of mobile robots operating in environments where no communication infrastructure like routers or access points is available. The robots have to create a mobile ad hoc network, in that case, it provides communication on peer-to-peer basis. The paper gives an overview of existing solutions how to route messages in such ad hoc networks between robots that are not directly connected and introduces a design of a software framework for realization of such communication. Feasibility of the proposed framework is shown on the example of distributed multi-robot exploration of an a priori unknown environment. Testing of developed functionality in an exploration scenario is based on results of several experiments with various input conditions of the exploration process and various sizes of a team and is described herein.

Keywords: multi-robot systems, distributed exploration, ad hoc network, limited communication, cooperation.

1. Introduction

One of the key issues when designing a robotic system involving more than one entity is how these entities communicate. The majority of current multi-robot solutions are centralized (i.e., they have a central element through which all messages between robots are passed) or assume an unlimited communication range allowing to send messages directly between two arbitrary robots. This setup has many advantages, but also several drawbacks. It can be hard, costly or even impossible to prepare external communication infrastructure assuring one-to-one communication accessibility in hostile, unexplored or large environments. Moreover, robustness of centralized systems which are highly dependent on a single node is low. Finally, it might be difficult to exchange information in large systems consisting of many robots as their communication capacities are limited and a number of messages is high. Some other arguments as well as real-world applications where ad hoc networks improved or might improve performance, scalability and robustness of robotic teams can be found in [1].

Elkady et al. [2] give a nice survey of robotic middlewares. Besides centralized client-server architecture employed, e.g. in Player/Stage/Gazebo [3], ACE/TAO library [4], implementation of CORBA (Common Object Request Broker Architecture) standard [5], is used for peer-to-peer communication in middlewares like CLARAty [6], MIRO [7], OROCOS [8]. Alternatively, the Ice framework by ZeroC [9] for remote procedure calling (RPC) can be used as it is already done in the Orca project [10]. Both ZeroC Ice and CORBA do not provide communication between directly inaccessible nodes in a decentralized fashion and although extensions for ad hoc networks exist [11] they were not used in the mentioned robotic middlewares. Communication in Robot Operating System (ROS) [12], the mostly popular robotic framework, is peer-to-peer, but this communication is mediated by so called *rosmaster*. This node provides naming and registration services and enables individual ROS nodes to locate one another. Once the nodes locate each other, they communicate directly. Also ROS does not provide routing of messages between inaccessible nodes.

On the other side, several works were presented which describe application of mobile ad hoc networks with routing in robotics. Witkowski et al. [13] present a cell-based network with master nodes in each cell and routing between distant nodes is provided through these master nodes which are distributed uniformly in the environment. The authors validate the approach on two scenarios. While measuring of the signal quality of WLAN and Bluetooth communication nodes in indoor environments with different kind of wall materials is the aims of the first one, the second experiment deals with radio-based positioning of robots in a dynamic environment. Corell et al. [14] study a wireless coverage problem with minimal requirements to hardware of robots. As a communication middleware they use Optimized Link State Routing (OLSR) [15] which provides TCP/IP routing for the resulting mesh network. A suite of algorithms that cover a defined area by a fully distributed robotic team, detect network disconnections, perform network repair, and repair area coverage is presented in [16]. Finally, Agrawal et. al [17] experimentally evaluate two routing protocols for a robotic swarm. The swarm aims to find a source of heat and a particle swarm optimization algorithm is employed as a search strategy.

The aim of this paper is to give an overview of routing protocols in ad hoc networks and choose the most

suitable for application in the multi-robot domain. The chosen protocol is than used in a design and realization of the framework for exploration of an a priori unknown environment which thus serves as a testbed for experimental evaluation of the protocol implementation.

The rest of the paper is structured as follows. Section 2 contains a structured list of requirements for a routing protocol, a classification of protocols, and describes the main representatives of the categories. Then a comparative analysis of routing protocols is performed. Analysis of three Message-Oriented Middlewares (MOM) describes in Section 3. The general application architecture and using of selected measures is presented in Section 4. Section 5 contains results of the experiments, which allow to verify the implementation performance of the proposed architecture. The final evaluation of the paper is in Conclusion.

2. Ad hoc routing protocols

A robot team forms an ad hoc network as it does not rely on a pre-existing infrastructure, while interaction between team members is done by a routing protocol. Operation efficient of a distributed robotic system requests a routing protocol to meet certain requirements. The key assumption is fully distributed operations and the protocol should not depend on a central control node. 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 [18].

An important requirement is the absence of routing loops. A routing loop occurs when the packets continue to be routed in an endless circle of the nodes. The protocol should not contain them, otherwise the overall efficiency is reduced, the routing 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.

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 movement. 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 lowest number of other nodes. While building the path and forwarding of messages, it is necessary to effectively avoid outdated routes [19].

The rest of the section contains a brief description of few most commonly used routing protocols and their comparison based on the above requirements. The main purpose of this comparison is to select the routing protocol that is the most suitable for teams of mobile robots which establish a wireless Mobile Ad hoc Network (MANET) without any centralized structure.

The routing protocols can be divided into proactive (table-driven), reactive (on-demand) and hybrid, which combines advantages of the first two categories.

2.1. Proactive routing protocols

Table-driven protocols constantly maintain the relevance of the routes between sources and destinations. To maintain a correct view of the network topology a protocol of this type responds to every change in the structure by sending changes across the entire network. Moreover, the protocol periodically sends route update messages to its neighbours. These protocols require to store routing information in one or more tables in each node [20]. These protocols do not scale well and control overheads are proportional to the number of nodes in the network.

Destination Sequenced Distance Vector (DSDV) [21] is one of table-driven protocols and it is a loop-free modification of the Bellman-Ford routing algorithm [22]. Each node maintains a routing table that contains entries for the next hop on the shortest path to all reachable destinations. The protocol uses a sequence number attribute to ensure the freedom of cycles and to distinguish stale routes. DSDV uses two types of dumps with updates of the routing table which sends to all immediate neighbours. The first type is periodically sent *full dump* which contains all available information about the routing information. The second type is an immediately sent *incremental dump* which sends on route changes and contains only changes that have occurred since the last full dump.

Another proactive protocol is Optimized Link State Routing (OLSR) [15] protocol, which is based on basic link-state algorithm. Every node maintains a network topology graph. Each node has thus shorter routes to every destination immediately, when data transmission begins. To keep the relevance of the topology a node periodically floods *hello* messages about its available links to the others. OLSR uses also *topology control* (TC) messages that contain information about one-hop neighbours. The main optimization lies in using of *multipoint relays* (MPRs). It reduces message overhead, because only nodes which were chosen as MPRs forward broadcast messages during the flooding process.

2.2. Reactive routing protocols

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. 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., it is generally less memory demanding compared to

proactive protocols and requires relatively little control traffic overhead.

Dynamic Source Routing (DSR) [23] is an example of on-demand routing protocols. DSR supports two types of operations: *route discovery*, when the node is required to transfer the data to the destination with an unknown route, and *route maintenance*, that allows to determine that the network topology is changed. After the work of these mechanisms the sender knows the complete hop-by-hop route to the destination. All transmitted routes stored in a route cache decrease time of route discovery. The protocol allows multiple alternative routes to the same destination and allows each node to perform load balancing.

Another example of reactive protocols is Temporally Ordered Routing Algorithm (TORA) [24] based on the link-reversal algorithm. TORA performs three functions: *route creation*, when a node uses a height metric to establish a destination oriented directed acyclic graph, *route maintenance*, when a node reestablishes routes due to topology changes and *route erasure*, when it is necessary to erase invalid routes. The protocol builds loop-free routes and provides multiple routes to alleviate congestion. However, TORA may produce temporary invalid routes, as well as the Light-weight Mobile Routing Protocol [25] which it is based on.

2.3. Hybrid routing protocols

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 [20].

Zone Routing Protocol (ZRP) [26] is a hybrid protocol, which uses a proactive approach to a close neighborhood and a reactive approach to remotely lying nodes of the network. In other words, each node divides the network into *intrazone* and *interzone*. The node immediately knows routes to all nodes within the intrazone. The discovery of such paths is produced by the algorithm similar to DSDV. If an initiator needs to send data to a recipient which is in the interzone the sender proactively maintains a route to the destination. This approach reduces latency in route discovery and decreases the number of control messages.

Ad hoc On-Demand Distance Vector routing (AODV) [27] is also a hybrid protocol based on the principles of the DSDV and DSR protocols. The protocol sends a periodic hello message, maintains the route table with a maximal single route for each destination and uses the principle of sequence numbering, as DSDV routing protocol does. On the other side, AODV provides a similar route discovery procedure as in DSR. A 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 as compared with the DSDV algorithm. Moreover, the route discovery packets do not contain a complete sequence of nodes, that is an advantage compared to DSR.

2.4. Comparison of routing protocols

There are several criteria [28] 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 [28] discusses three routing protocols - 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 [29]. 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 the reactive OLSR and the hybrid ZRP.

The publication [30] 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 show excellent results for the packet delivery ratio, which

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

TABLE 1. A brief comparison of message oriented middlewares.

ranged from 95 to 100% depending on the network load.

AODV is selected for implementation of the exploration framework on the basis of the above mentioned comparisons. Simulations use robot teams of various sizes and display 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.

3. Message oriented middleware

The exploration application uses message passing to communicate between team members. 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.

The synchronous approach implies that message exchange takes place at a time when applications are running simultaneously. It uses temporary sender blocking before receiving a response from a recipient and infinitely blocking in case of recipient unavailability. The robots move in the environment during the exploration problem solving and can get out of the communication range. In addition, the robot can be turned off, for example, due to the battery discharge. In such cases, the message sender is blocked. That is, synchronous approaches are not suitable for use in distributed robotic systems that solve exploration problem.

The asynchronous approach does not require simultaneous operation of a sender and a recipient. The intermediate level of software provides all the operations for receiving, storing and sending of messages. The most known type of such software is Message-Oriented Middleware (MOM).

Table 1 provides a brief comparison of main parameters of these three widely used asynchronous Message-oriented middlewares: ActiveMQ Apollo [31], ZeroMQ [32] and nanomsg [33].

As can be seen from the table, all three libraries are open-source and release new updates. The first two libraries are well documented and have a large community of users. The latter is improved and rewritten ZeroMQ library, which currently has only the manual pages, but the number of nanomsg users is constantly growing.

All considered libraries support binding for commonly used programming languages as Java, C, C# and C++. ZeroMQ and nanomsg have also bindings for Ruby, Go, Haskell, Perl, PHP and others.

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. All three libraries include both types of message passing.

The libraries support a wide range of transport protocols, but Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and in-proc are relevant for the purposes of the application. All considered libraries support transfer of data over TCP. ActiveMQ Apollo additionally supports UDP, while the other two do not support it. ZeroMQ and nanomsg can use *in-proc* transport protocol. It is an inter-thread transport, which is much faster than TCP and can transfer data between threads of a single application.

The most important difference is that the basic principle of ActiveMQ Apollo is to create and configure a broker. 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. Contrariwise,

ZeroMQ and nanomsg support brokerless model.

Nanomsg library was developed by one of the key creators of the ZeroMQ library and it fixes most of the ZeroMQ drawbacks. The main one is the problem with fault tolerance. Nanomsg provides a new pattern of communication that does not require redundant confirmation from a recipient in the model *Request-Router* that resembles the work of procedure calls. 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.

On the contrary, 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.

The nanomsg library was chosen for use in the application on the basis of the above comparison of the characteristics of the libraries. Because the library doesn't provide an implementation of UDP, the transport protocol is proposed to implement using standard sockets.

4. Distributed exploration

Exploration as a process of autonomous navigation of a mobile robot in order to build a model of an a priori unknown working environment in a shortest possible time is one of the fundamental problems in mobile robotics. In general, exploration is an iterative process that performs the following operations at each step: 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 the 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 [34, 35]. The main source of optimization is preparation of a set of possible goals and selection of a next goal from them, which is the most favorable for exploration purposes.

Besides the tasks mentioned above, the problem of distributed terrain exploration by a multi-robot team requires solving the problem of communication between team members and the problem of coordination their actions. This is why we selected this task as a testbed for evaluation of the AODV routing protocol.

Information about the terrain which is exchanged between team members and the method of determining the set of possible navigation goals depends on an exploration approach that is selected for implementation. Frontier-based exploration [34] is chosen as one of the most common and widely used approaches which also has the extension for the case of multi-robot exploration [36]. The approach is based on the concept of *a frontier*, which is a boundary between the already known part of the terrain and still an unknown part. The internal representation of the map is an occupancy grid, which uses Bayesian based updating technique.

The main advantage of the environment exploration by a multi-robot team is the ability to easily share knowledge about the terrain between its members. Exploration of non-overlapping areas by different robots allows to raise efficiency of teamwork and reduce the total exploration time. Each robot in the team conducts the current goal selection independently on the basis of its current knowledge of the environment and the goals of the other team members. A greedy algorithm is chosen as the most primitive and native approach. It means, that the closest frontier to the current position of the robot is selected as the goal. In addition, the selected goal should not lie in the areas of the current goals of the other team members, which have active communication link to the robot. After the goal selection the robot informs the others about the selected goal and reports the priority for this goal. If an other team member has the goal in the same area with a higher priority, it sends information about its current goal and forces the robot with a lower priority to choose another goal and generate a new plan.

Description of the software architecture of a single robot performing distributed exploration and communicating with other robots making use of AODV follows in the next paragraphs.

While designing the application structure for a single robot it was assumed that the application can operate in two roles: *Robot* and *Display*. The application for a robot role controls a robot as it carries out reading of data from its sensors, produces the robot movement in the terrain, performs path planning and communication with other team members. The application for a display role is used to monitor and display the status of the multi-robot exploration process and can vary from the specific implementation. The application of this type is also a part of the mesh network, but it does not directly participate in the exploration.

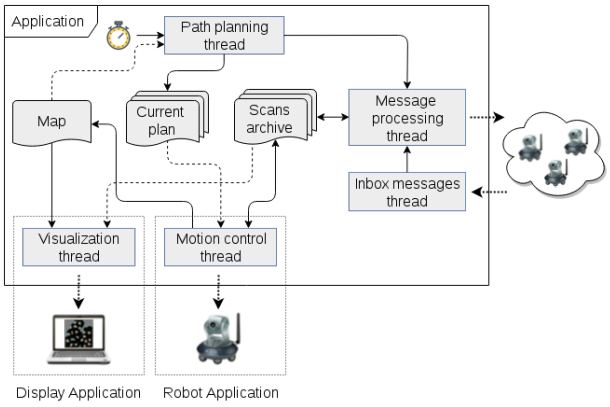


Figure 1. Proposed application structure.

It is recommended to use AODV routing protocol based on the analysis result of routing protocols

from Section 2. The application should have the tools to unicast and broadcast messages for AODV implementation. TCP is the most appropriate protocol for the purposes of sending messages between robots. Flooding and sending hello messages can be realised with using less reliable UDP, which however supports broadcasting. According to the results of a comparative analysis in Section 3 nanomsg recommended for implementation as a MOM. Thus, the nanomsg library provides the transfer of messages between robots via TCP and uses in-proc for transfer of information between different parts within the application itself. Because this library does not support the UDP, broadcasting should be implemented by means of language in which the application will be realized.

Fig. 1 shows the proposed 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. E.g., plan generation usually runs once per two seconds or on the request from the other robot. A command to control robot movement can be sent every 100 milliseconds. The optimal frequency of sending the heartbeat for the purposes of AODV 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. For these reasons, the structure is divided into 5 logical parts, which can be implemented as threads: motion control/visualization, path planning, timer, message processing and inbox messages part.

These parts of the application share three data structures: map, current plan and scans archive. *Map* keeps an internal representation of the application's view about the environment and depends on the selected type of the exploration. In the case of frontier-based exploration, the map will be in the form of the occupancy grid. *Current plan* is the path to a goal selected at this iteration of the exploration process in the form of a polyline. *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. The principle of the transfer of local maps between robots during the terrain exploration of a multi-robot team is described in [36].

The application for a robot role is determined by using the *motion control thread*. It is responsible for receiving the current plan, sending signals to the robot motors for movement to the next selected goal, using the robot sensors to obtain scans, which are then stored in the scans archive and receiving new scans from the scans archive received from other robots. Finally, 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.

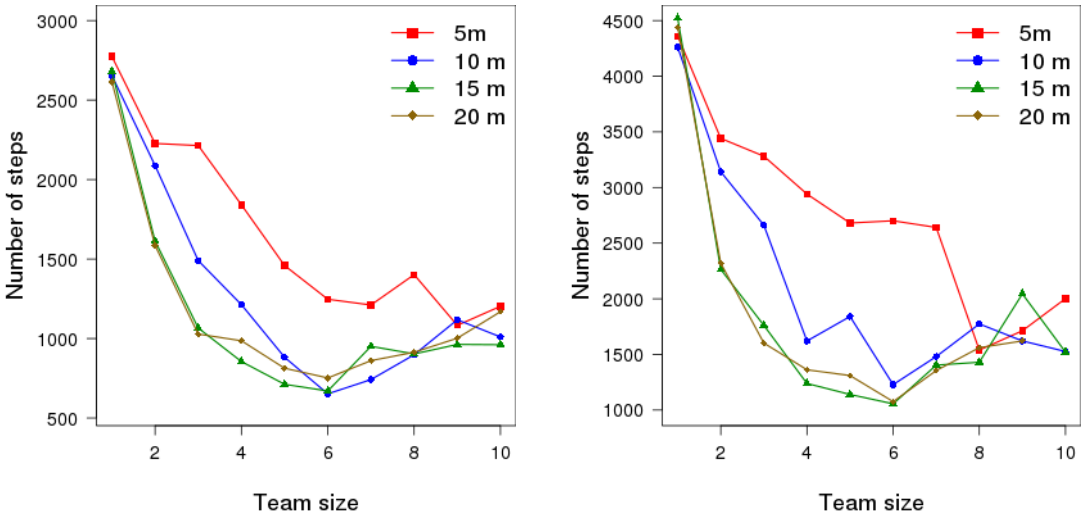
The inbox messages thread deals with receiving messages from other team members and sending them in the messages processing thread.

5. Experimental results

For functional testing of the proposed framework, the application of a multi-robot team member was implemented. It is implemented in C++ and simulates the work of an exploration robot for experimental verification of the effectiveness of the proposed application structure. It imitates operations of the real robot equipped with a laser range-finder. The robots are controlled by a simple controller which guides the robot exactly along a planned path with a constant speed. Ray casting was used to simulated a laser range-finder with no noise incorporated.

Behavior and properties of the implemented application were experimentally evaluated and its characteristics were compared with a single-robot exploration. The experiments were conducted using a device which has 32 Central Processing Units (CPU) and has 8 GB of memory.

All tests were performed in simulation 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.



(a). (b).

Figure 2. Dependence of the steps on the team size. The value of the laser range is: (a) 2 meters; (b) 1.5 meters.

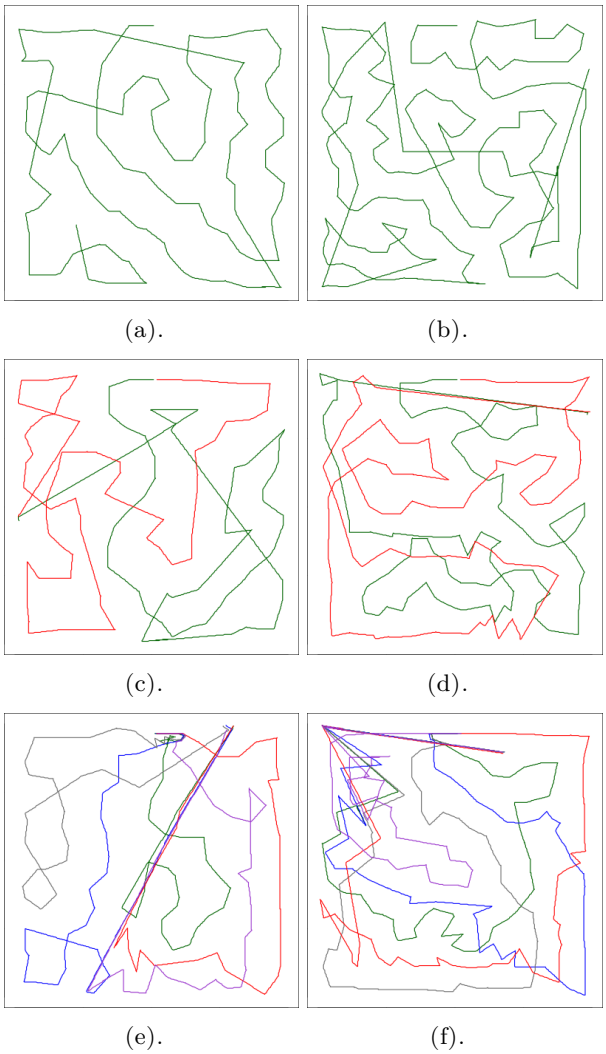
The time measurement of the exploration is made in *steps*, i.e the number of calls of the function that changes the robot position. This function is triggered periodically after a certain period of time. Thus, a step is a convenient alternative to time. In addition, the usage of steps metric allows to maintain usable statistics of events that pass between sequential steps.

5.1. Time of the exploration

Fig. 2 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 Fig. 3, 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 Fig. 2(a) and Fig. 2(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 about the environment. Thus, it is necessary to make larger movements



(a). (b).

(c). (d).

(e). (f).

Figure 3. The resulting paths after the exploration. The laser range of the first column is 2 meters, the second column is 1.5 meters. Rows are separated by team size: (a)+(b) 1 robot; (c)+(d) 2 robots; (e)+(f) 5 robots.

and get more scans to obtain a complete map of the terrain.

5.2. Plan generations

Fig. 4 shows the frequency with which the application is launching a new request for a new generation of the plan. A plan generation runs periodically by a timer part of the application, as described in Section 4. In addition, it can be run by a request from an another team member. In this implementation, if two robots have the goal at the same area, the robot with a higher priority of the goal sends a request to the robot with a lower priority to select a new navigation goal and generate a plan to the selected goal. The launching frequency of generation of the plan is 20 steps in the application. The graphs shown on Fig. 4 contain all plan generations, which had one member of the robot team during complete exploration of the environment. Thus, all function values under the value of 20 steps means that the corresponding plan generation is launched on request from an another robot.

Fig. 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 graph shown in Fig. 4(d) corresponds to Fig. 5(a). The robots start the goal selection. 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 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 Fig. 5(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 Fig. 5(b).

Then, most of the terrain is explored and the robots are in a small unexplored part of the environment (see Fig. 5(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 then leads to increase of the total exploration time.

6. Conclusion

This paper addresses the problem of communication in distributed multi-robot teams forming a mobile ad hoc network. A list of requirements for a routing protocol was drawn up. The comparative analysis of routing protocols has shown that AODV routing protocol is most suitable. In addition, the analysis of three MOMs was conducted and the nanomsg library was chosen as the most suitable for implementation. Multi-robot exploration has been selected as a proof-of-concept task. The general structure of the robot application which carries out exploration of an unknown environment in a multi-robot team and which communicates with other robots employing AODV has been proposed. To verify the proposed architecture and offered tools the application has been implemented and has been tested by series of experiments. Thus, the paper describes the architecture of the application and possible means of implementation in a generalized form. The specific implementation can be done for different systems and written on a wide range of languages.

In 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 paper as it was out of scope of it. Analysis and comparison of a variety of sophisticated goal allocation mechanisms will be another stream we would like to go in future.

Acknowledgements

This work has been supported by the Technology Agency of the Czech Republic under the project no. TE01020197 "Centre for Applied Cybernetics". Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005), is greatly appreciated.

References

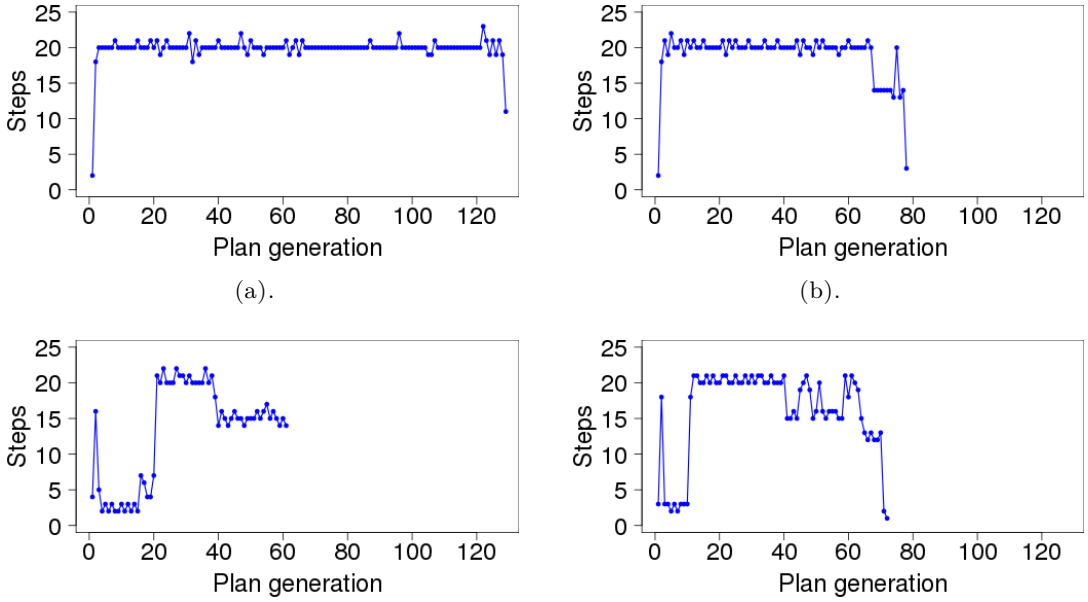
[1] W. Zhigang, L. Liu, M. Zhou. Protocols and applications of ad-hoc robot wireless communication networks: An overview. *International Journal on Intelligent Control Systems* **10**(4):296–303, 2005.

[2] A. Elkady, T. Sobh. Robotics middleware: A comprehensive literature survey and attribute-based bibliography. *Journal of Robotics* **2012**:1–15, 2012. DOI:10.1155/2012/959013.

[3] B. P. Gerkey, R. T. Vaughan, A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *In Proceedings of the 11th International Conference on Advanced Robotics*, pp. 317–323. 2003.

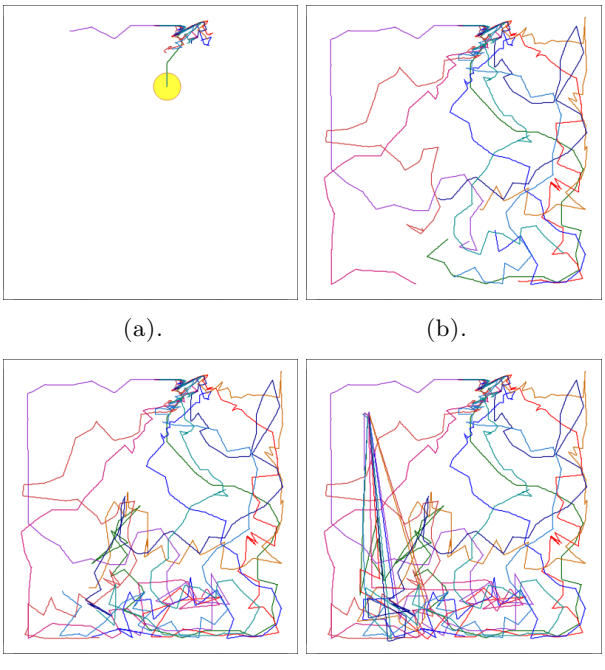
[4] ACE/TAO, 2016. `http://www.cs.wustl.edu/~schmidt/TAO.html`.

[5] CORBA, 2016. `http://www.corba.org/`.



(a). (b). (c). (d).

Figure 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.



(a). (b). (c). (d).

Figure 5. The passed paths for the team of 10 robots and the laser range is 20 meters at the step: (a) 80; (b) 580; (c) 800; (d) after finishing exploration.

[6] I. A. D. Nesnas. *The CLARAty Project: Coping with Hardware and Software Heterogeneity*, pp. 31–70. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. doi:10.1007/978-3-540-68951-5_3.

[7] H. Utz, S. Sablatnög, S. Enderle, G. K. Kraetzschmar. Miro - middleware for mobile robot applications. *IEEE Transactions on Robotics and Automation* **18**(4):493–497, 2002. doi:10.1109/tra.2002.802930.

[8] H. Bruyninckx, P. Soetens, B. Koninckx. The real-time motion control core of the orocos project. In *IEEE International Conference on Robotics and Automation*, pp. 2766–2771. 2003. doi:10.1109/robot.2003.1242011.

[9] ZeroC. Ice, 2016. `https://zeroc.com/products/ice`.

[10] A. Brooks, T. Kaupp, A. Makarenko, et al. Orca: A component model and repository. In *Springer Tracts in Advanced Robotics*, pp. 231–251. Springer Berlin Heidelberg, 2007. doi:10.1007/978-3-540-68951-5_13.

[11] L. Lima, A. Calsavara. A framework for corba interoperability in ad hoc networks. In *ACM Symposium on Applied Computing*, pp. 930–934. ACM, 2007. doi:10.1145/1244002.1244206.

[12] M. Quigley, K. Conley, B. P. Gerkey, et al. ROS: an open-source Robot Operating System. In *IEEE International Conference on Robotics and Automation (ICRA) – Workshop on Open Source Robotics*. 2009.

[13] U. Witkowski, M. El-Habbal, S. Herbrechtsmeier, et al. Ad-hoc network communication infrastructure for multi-robot systems in disaster scenarios. In *Proceedings of the International Workshop on Robotics for Risky Interventions and Surveillance of the Environment*. 2008.

[14] N. Correll, J. Bachrach, D. Vickery, D. Rus. Ad-hoc wireless network coverage with networked robots that cannot localize. In *IEEE International Conference on Robotics and Automation*, pp. 3878–3885. 2009. doi:10.1109/ROBOT.2009.5152742.

[15] P. Jacquet, P. Muhlethaler, T. Clausen, et al. Optimized link state routing protocol for ad hoc networks. In *Proc. IEEE International Multitopic Conference (INMIC'01)*, pp. 62–68. Institute of Electrical and Electronics Engineers (IEEE), Lahore, Pakistan, 2001. doi:10.1109/inmic.2001.995315.

[16] A. Derbakova, N. Correll, D. Rus. Decentralized self-repair to maintain connectivity and coverage in

networked multi-robot systems. In *IEEE International Conference on Robotics and Automation*, pp. 3863–3868. 2011. DOI:10.1109/ICRA.2011.5980367.

[17] R. Agrawal, R. Tripathi, S. Tiwari. Performance evaluation and comparison of aodv and dsr under adversarial environment. In *IEEE International Conference on Computational Intelligence and Communication Networks*, pp. 596–600. 2011. DOI:10.1109/CICN.2011.129.

[18] B. B. Maqbool, M. A. Peer. Classification of current routing protocols for ad hoc networks - a review. *International Journal of Computer Applications* **7**(8):26–32, 2010. DOI:10.5120/1270-1749.

[19] H. Ehsan, Z. A. Uzmi. Performance comparison of ad hoc wireless network routing protocols. In *IEEE International Multitopic Conference*, pp. 457–465. 2004. DOI:10.1109/inmic.2004.1492924.

[20] A. Hinds, M. Ngulube, S. Zhu, H. Al-Aqrabi. A review of routing protocols for mobile ad-hoc NETworks (MANET). *International Journal of Information and Education Technology* **3**(1):1–5, 2013. DOI:10.7763/ijiet.2013.v3.223.

[21] C. E. Perkins, P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Comput Commun Rev* **24**(4):234–244, 1994. DOI:10.1145/190314.190336.

[22] C. Cheng, R. Riley, S. P. R. Kumar, J. J. Garcia-Luna-Aceves. A loop-free extended bellman-ford routing protocol without bouncing effect. *SIGCOMM Comput Commun Rev* **19**(4):224–236, 1989. DOI:10.1145/75246.75269.

[23] D. B. Johnson, D. A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*, pp. 153–181. Springer US, 1996. DOI:10.1007/978-0-585-29603-6_5.

[24] V. D. Parka, M. S. Corsonb. A highly adaptive distributed routing algorithm for mobile wireless networks. In *IEEE Computer and Communications Societies. Driving the Information Revolution.* 1997. DOI:10.1109/infcom.1997.631180.

[25] Z. Wang, Y. Chen, C. Li. PSR: A lightweight proactive source routing protocol for mobile ad hoc networks. *IEEE Transactions on Vehicular Technology* **63**(2):859–868, 2014. DOI:10.1109/tvt.2013.2279111.

[26] Z. J. Haas. A new routing protocol for the reconfigurable wireless networks. In *IEEE International Conference on Universal Personal Communications Record (ICUPC)*, vol. 2, pp. 562–566. 1997. DOI:10.1109/icupc.1997.627227.

[27] S. R. Das, C. E. Perkins, E. M. Belding-Royer. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, 2013. DOI:10.17487/rfc3561.

[28] S. Kumar, D. Singh, M. Chawla. Performance comparison of routing protocols in manet varying network size. *International Journal of Smart Sensors and Ad-hoc Networks (IJSSAN)* **1**(2):51–54, 2011.

[29] R. Ahuja. Simulation based performance evaluation and comparison of reactive, proactive and hybrid routing protocols based on random waypoint mobility model. *International Journal of Computer Applications* **7**(11):20–24, 2010. DOI:10.5120/1291-1765.

[30] J. Broch, D. A. Maltz, D. B. Johnson, et al. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pp. 85–97. Association for Computing Machinery (ACM), 1998. DOI:10.1145/288235.288256.

[31] Apache. Apollo 1.7.1, 2016. `https://activemq.apache.org/apollo/`.

[32] P. Hintjens. *ZeroMQ: Messaging for Many Applications*. O'Reilly Media, 1st edn., 2013.

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

[34] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. IEEE Computational Intelligence in Robotics and Automation*, pp. 146–151. 1997. DOI:10.1109/CIRA.1997.613851.

[35] M. Kulich, J. Faigl, L. Přeučil. On distance utility in the exploration task. In *IEEE International Conference on Robotics and Automation*, pp. 4455–4460. 2011. DOI:10.1109/icra.2011.5980221.

[36] B. Yamauchi. Frontier-based exploration using multiple robots. In *Second International Conference on Autonomous Agents (AGENTS)*, pp. 47–53. 1998. DOI:10.1145/280765.280773.