Collaborative Motion Planning of Autonomous Robots

Takashi Okada*†‡, Razvan Beuran*†, Junya Nakata*†‡, Yasuo Tan*†‡ and Yoichi Shinoda*†§
*Hokuriku Research Center, National Institute of Information and Communications Technology, Japan
†Internet Research Center, Japan Advanced Institute of Science and Technology, Ishikawa Japan
‡School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa Japan
§Center for Information Science, Japan Advanced Institute of Science and Technology, Ishikawa Japan

Email: tk-okada@jaist.ac.jp

***Abstract*—In disaster areas, office buildings, or at home, multiple autonomous networked mobile robots may act instead of human beings. These robots have to move to their destination so as to perform their function. For this purpose they need to be able to recognize the changes in the surrounding environment. They are equipped with a motion-planning method in order to avoid in real time collisions with other robots or obstacles. In this paper we propose a motion planning method based on PRM (Probabilistic Roadmap) algorithm. To evaluate our method, we constructed an experiment platform based on StarBED, which is a large-scale network testbed. By using the virtual environment manager Map Manager, the WLAN emulator QOMET, and the experiment-support software RUNE we are able to perform emulation of large-scale autonomous networked mobile robot systems. The experimental results confirm the usefulness of collaborative motion planning, which results in reaching faster the destination and in less frequent re-planning.**

***Index Terms*—autonomous mobile networked robot, motion planning, real time, emulation, large-scale simulation**

I. Introduction

In disaster areas or office buildings, autonomous robots may act instead of human beings. Rescue robots are able to accomplish many tasks in dangerous places where humans cannot enter, such as sites where harmful gases or high temperature are present, a hard environment for humans. Cleaning robots can work automatically and save costs by performing various routine tasks. Home assistant robots are expected to support daily activities at home. In all these examples robots have to move to their destination in order to perform their function. For this purpose they need to be able to recognize the changes of environment using various sensors and cameras, and be equipped with a motion planning method in order to avoid collision with obstacles.

The experiments for evaluating such motion planning algorithms are difficult, since the cost of these real autonomous robots is high. This is particularly true if researchers want to experiment with more than a few robots, and need to test systems with tens or even hundreds of robots. Consequently many researchers try to make experiments or evaluate their algorithms or methods on software simulators. But the results from these simulators are not very accurate. The difference between real systems and simulators can be big. In real environments, many kinds of noise affect conditions. For example, the communication in real environments is affected by electro-magnetic waves and so on. Researchers cannot always obtain good results on real systems, even if they have once obtained good results from experiments on simulators. We need solutions for covering this difference.

In this paper, we propose a new motion planning algorithm of autonomous mobile networked robots. And in order to perform large-scale experiments in realistic conditions, we create an emulation platform.

II. Related Work

A. Motion Planning

In the situation when multiple robots act in same environment, they need navigation in order to move to their destination on purpose of accomplishing their tasks and avoid possible collisions with other robots or obstacles. Multiple robot motion planning are usually classified as *centralized* or *decentralized* methods. *Centralized* planners construct plans by one robot and this robot distributes the plan. In *decentralized* method, each robot plans independently.

One traditional solution is sampling-based motion planning [1]. In this approach, samples are organized into regular grids or hierarchical ones. These grids express the location of the free space. The planner remembers the locations which robots already visited. But the size of these grids increases exponentially according the number of degrees of freedom of the robot.

A probabilistic roadmap (PRM) planner has become popular method for planning because of the speed of calculation. Many of PRM planner of multiple robot are *decentralized*. A PRM planner randomly samples the possible destination of robots and registers the collision free samples as milestones. The planner tries to connect pairs of these milestones and saves these collision free connections as the trajectories of robots. Probabilistic roadmap means that in this graph, the edges are the trajectories of robots, the vertices are the milestones, and the undirected graph jointing the trajectories is the probabilistic roadmap. The planner finds the optimal path from the graph; for example it may set some weights on these edge and use Dijkstra's algorithm.

PRM planners are not complete in the traditional sense. But they are *probabilistically complete* under certain assumptions. It means that the probability of failure decreases exponentially to zero with the number of iterations [2].

Now we mention two types of PRM. One is single-query planners. It computes a new roadmap from scratch for each new query [2]. Second one is multi-query. It precomputes the roadmap and re-use the roadmap for answering queries [3]. It has been proved that, under reasonable assumptions about the geometry of the degree of freedom of robots, a relatively small number of milestones picked uniformly at random are sufficient to capture the connectivity of the degree of freedom of robots with high probability.

In addition to PRM method, robots need a technique to avoid collisions with other robots. "*Velocity turning*" [11] method computes the relative velocities of the robots to avoid inter-robot collision. Another way to solve this problem is "*Prioritized Planning*" [12]. In this method, robots avoid possible collisions depending on their priority.

B. Robot Experiment

When researchers are implementing or evaluating something related to autonomous robots, for example path planning algorithms or dynamic network construction and so on, one of the choices is simulation, by using a product such as Webots [9], co-developed by the Swiss Federal Institute of Technology in Lausanne, Switzerland. EyeSim [5] is a multiple mobile robot simulator that allows experimenting with the same unchanged EyeBot programs that run on the real robots. Software simulator models robot components, network connections, motion and time scheduling, and surrounding environment. If researchers select to use such simulators, they have to implement simulator-specific modules by themselves. Surely these software simulators can test various kinds of environment that are hard to implement in a real system. But the results from these software simulators may differ with respect to those that would be observed in a real system. The difference between the results from software simulator and results of a real system may be large.

The second choice is to experiment in real environments. RoboCup [6] involves teams of five small robots, each up to 18 cm in diameter and 15 cm in height. The robot teams are entered into a competition to play soccer against opponent teams fielded by other research groups. Hsu [7] made experiments on his research team testbed. This robots move frictionlessly on an air bearing on a 3 m x 4 m table. ARL (Stanford Aerospace Robotics Laboratory) [10] makes various experiments of real robots. These real environment systems take much cost for real robots and sensing. If researchers want to test large number of robots, it is hard to make ready all robot hardware including processing power and connection to wireless network.

The third alternative is emulation. The difference between simulation and emulation is like the difference between modeling and emulation. A simulation is modeling a system out of approximations or inferences. An emulation is emulating or imitating a different environment of hardware or software. This method can not only get reliable experiment results but also doesn't take much cost. If researchers can emulate robots and other environments on computers, the cost is only computers. And this approach covers the gap between real implementation and software simulation.

III. Motion Planning

A. Proposed Approach

The performance of motion-planning algorithm can be characterized by the following properties: speed, completeness, and optimality. In dynamic and unknown environment, robots must plan and re-plan their motion many times, because the environment dynamically changes in time. Therefore in the case when robots need to continuously plan on-the-fly their trajectory, the algorithm speed is one of the most important properties.

B. Proposed Algorithm

In proposed system, robots are equipped with one of PRM method that base on "Path Planning in Expansive Configuration Spaces" [2]. The basic idea of their algorithm makes two trees from $source$ and $destination$ of robots. The vertices are assumed as the candidates which robots go through and the edges are assumed as the trajectories which robots move. If these two trees can joint, it can be the trajectory from $source$ to $destination$.

This algorithm iteratively executes two basic steps, *expansion* and *connection* until either a path is found or the maximum number of iterations is counted. $Expansion$ is the method which builds and grows two trees. $Connection$ just checks whether two trees can be joined or not.

In the case study of this paper, robots are considered to be placed in environment that are unknown or change in a dynamic manner. The motion-planning method described above is not suited for such a case. For example, their planner can not predict whether there are any collisions or not in the tree built from destination, $T_{destination}$. It means that the planner has no way to know the time when the robot will reach the milestones in the tree, $T_{destination}$. Hence the planner cannot expand the tree, $T_{destination}$, in these cases.

In order to adapt the planner to these conditions of unknown and dynamic environment, the notion of the time is important in these cases. Because even if the planner found some path to destination, the path might be inefficient because of excessive calculating time. In dynamic environment, robots have to plan their path in real time, it is important that the time keeps on running during planning new path. And I adapted the motion-planning method discussed in the following sections.

C. Definitions

At first the parameters of this motion-planning method should be defined. T is the tree which is constructed by V (set of vertex) and E (set of edge). The parameter q is the element of degrees which robots can move free. C is the configuration space, means all the set of this q. A configuration q is *free* if the robot placed at q does not collide with obstacles. The

set of all free configurations are defined the *free space F*. *Configuration time space* is one of configuration space but including the time parameter. Figure 1 describes this. This configuration space covers the other moving robots as moving obstacles, and the robots avoid to pick this configuration.

D. Expansion

As mentioned above, in unknown and dynamic environment, introduced path-planning [2] does not work well. Then my algorithm uses following method :

- **Grows only source tree**
  This means that the planner grows only one tree (T_{source}) , since the planner cannot check the collisions of $T_{destination}$.
- **Adapt time parameter**
  The milestones have the **time** parameter when the robots reach to the milestone. If milestones do not include this value, the planner cannot predict the collision with other autonomous mobile networked robots or obstacles. This is fulfilled by *Configuration time space* explained the previous part.

Algorithm 1 Expansion

```
Pick a node x from V with probability 1/w(x)
Sample K points from N(x, time)
for each configuration y ∈ K do
  calculate w(y) and register y with probability 1/w(y)
  if link(x, y) return YES then
    put y in V and place an edge between x and y
  end if
end for
```

In step 1, a node x is selected from V with the probability $1/w(x)$. $w(x)$ describes weight of node x , it is the number of node around node x within d_w . Now $count(S)$ is assumed as a function which count the number of elements of set S , $dist(x, y)$ returns a distance between x and y .

$$w(x) = count(\{v \in V | dist(v, x) < d_w\})$$

In step 2, number of K nodes are sampled with $N(x, time)$. $N(x, time)$ is the set of milestone following:

$$N(x, time) = \{q \in C(time) | d_{min} < dist(q, x) < D_{max}\}$$

From step 3 to step 6, if the path between the sampled milestone and x is collision free, the milestone is saved as new vertex in V . The function $link(x, y)$ checks whether the path from x to y is collision free or not.

E. Connection

As expressed in *Expansion* method, the planner tries to connect only newly added milestones last step.

In step 1, the newly added x are the milestones which the planner added last *Expansion* step. In step 2, if $link(x, q_{destination}$ returns YES for some x, then a path is found between q_{source} and $q_{destination}$ through x.

parameter	definition
d_w	distance covering weight range
K	sampled number of milestones
d_{min}	minimum distance putting milestone
d_{max}	maximum distance putting milestone

TABLE I
PARAMETER DEFINITIONS OF *Expansion*

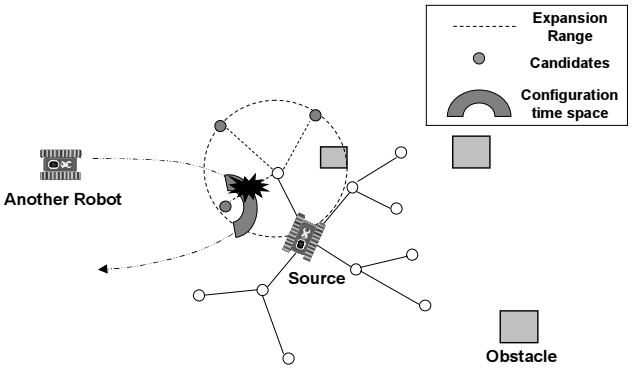

Fig. 1. Expansion phase

Basically the robots cannot know the entire map at once, but the robot use PRM planning for avoiding collisions in some complicated area. Then the idea is going straight trajectory after passing through this complicated area.

F. Prioritized Planning

In addition to these two methods, the system needs to correspond to large-scale multiple robots. Every robot has to communicate and avoid possible collisions with each other if robots detect the collision in their current trajectory. Then these robots avoid the collision according to the priority if there are two robots, *A* and *B*, let's assume the priority of *A* is higher than the priority of *B*. Now their trajectories make collision if they keep moving on their trajectories. Then robot *A* keeps its trajectory, and robot *B* needs to find a trajectory which avoids possible collision.

This prioritized planning means that the lower priority robot regards the higher priority robot as a moving obstacle. Then the configuration space of this lower priority robot is limited. This limited configuration space is "*Configuration time space*",

Algorithm 2 Connection

```
for every newly added x ∈ V_source do
  if link(x, q_destination) return YES then
    register the path as new path
  end if
end for
```

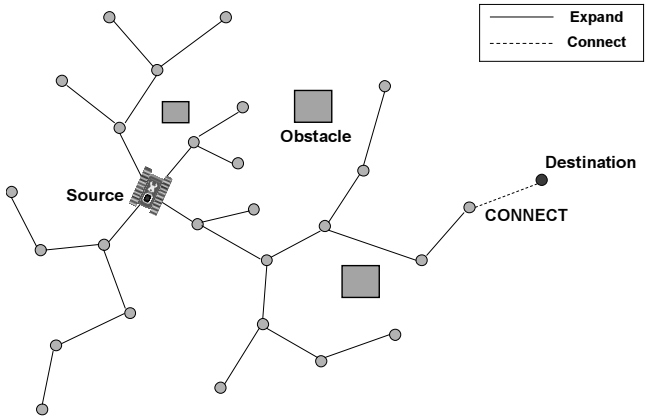

Fig. 2. Connection phase

and all of robots have to consider their trajectory on attention of this limitation. This means that the lower the priority of robot is, the more the configuration space of the robot decreases.

G. Algorithm

By using above *Expansion* and *Connection* algorithm and *Prioritized Planning*, I adapt PRM planner to multiple robot in real time in unknown and dynamic environments as follows :

Algorithm 3 Proposed Algorithm

while $time_{current} < time_{limit}$ **do**
  Expansion
  Connection
end while
if no successful path found **then**
  return $time_{wait}$ time waiting path at current position
else
  return shortest path from all of paths
end if

At first previous method *Expansion* and *Connection* are executed till $time_{limit}$. Next, from step 5, shortest path is selected from all found paths. But if the planner could not find any successful paths, a waiting path is returned. It means that robots wait $time_{wait}$ at current position.

By this limitation of time, when they are planning, robots can keep on moving. That's why the robots plan their future trajectory, that is the robots start to plan from a source position in the near future. This time limitation is the time when they reach the future position.

IV. Experiment Platform

Emulation is one of the effective and appropriate methods to experiment with the algorithm we described. In this section, we propose a new experiment platform to emulate a large-number of autonomous mobile networked robots.

A. Overall Architecture

The robots in the proposed system cooperate in order to reach a destination while avoiding collisions and accomplish some given tasks. To test such robot algorithm, a suitable network testbed is needed. For this purpose we use StarBED [8]. StarBED is large-scale network testbed which has a large number of PCs (more than 800). These all experiment PCs are equipped with multiple network interfaces (100 Mbps or 155 Mbps or 1 Gbps type). These PCs can easily construct large-scale networks by changing switches configuration. All of robots are emulated on StarBED PCs and communicate through Ethernet.

To fulfill executing emulated robots on StarBED, following systems are needed :

- **Sensor function emulation**
  For the system assumes real PCs to be autonomous mobile networked robots, some modeling or emulation are needed to handle the messages from various sensors on robots.
- **Network emulation**
  All of the network on StarBED are wired network. But mobile robots communicate using wireless network. Therefore these robots need to emulate wireless communication.

B. Sensor Function Emulation : Map Manager

For emulating robots on StarBED PCs, the system needs some modeling or emulation to represent the behaviors of the hardware of robots, for example motors, cameras, some sensors and so on. In unknown and dynamic environment the system needs to know dynamic changes of environment and some structure which synchronize the events which happens on every robots at the same time. In this system, Map Manager administrates all of the hardware information by sensor function emulation.

Each robot is able to know the events of their hardware from the emulated sensing function information, for example positions from "*GPS*", images from "*Visual sensor*", detection of WLAN radio signal, some alarm messages from "*Shock sensor*" and so on. These hardware information comes from Map Manager, it means that Map Manager handles the sensor function and converts the events from device driver to comprehensible information by the robot application. Figure 3 describes the architecture of Map Manager.

C. Network Emulation

The mobile networked robots communicate on wireless network, therefore the system has to emulate this network system. Figure 4 shows the overall network topology of this system. There exists two networks which are connected by VLAN, and there are two types of traffic on each different

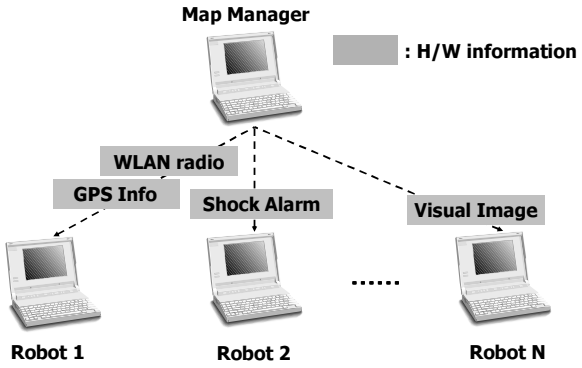

Fig. 3. Map Manager architecture

networks. On "*Management Network*", the sensing function information which are shown in the previous section are sent (visual information, WLAN radio signal detection and so on). This network is Ethernet network , and these sensing function information is immediately received by emulated robot PCs and Map Manager. "*Experiment Network*" is the network which is assumed as WLAN network at disaster areas or office buildings or home. The robots communicate and cooperate by sending the motion-planning messages through *Experiment Network*.

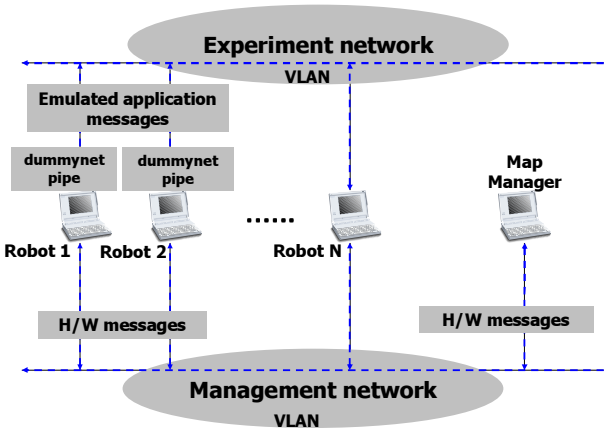

Fig. 4. General system overview

When robots receive WLAN radio signal detection information, WLAN communication module is activated. To reproduce the WLAN communication, the system applies some emulated WLAN configuration values (for example, the bandwidth or the delay or the jitter and so on) to Ethernet cable network. This emulation can be realized by WLAN emulator QOMET [13]. This will be detailed in Section V.

In order to implement such complex experiment system, the system includes an experiment-support software, called RUNE (Real-time Ubiquitous Network Emulation environment) [15]. It provides additional functionality which supports large-scale emulation system. This will be detailed in Section VI.

V. WLAN EMULATION: QOMET

The WLAN communication emulation engine QOMET is deployed in the emulated robots to allow recreating network conditions similar to those occurring in a real WLAN environment by scenarios.

The scenario-driven architecture for WLAN emulation has two stages. In the first stage, from a real-world scenario representation QOMET create a network quality degradation (ΔQ) description which corresponds to the real-world events (see Figure 5).

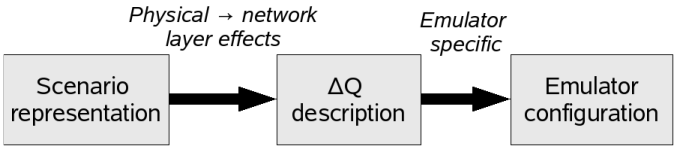

Fig. 5. Two-stage WLAN emulation

By quality degradation we mean the change in network service quality between two measuring points; we denote this degradation by the shorthand ΔQ. Since the ΔQ description represents the varying effects of the network on application traffic, the WLAN emulator's function is to reproduce it. The ΔQ description calculated in the first stage is therefore converted into an emulator configuration that is used during the effective emulation process to replicate the user-defined scenario in a wired network. This makes it possible to study the effects of the scenario on the real application under test. This WLAN emulation model is an aggregation of several models used at the various steps of the conversion of the scenario representation to the network ΔQ description which is needed to recreate those scenario conditions. Models exist for each level of the conversion: real world scenario to physical layer, physical layer to data link layer, and, finally, data link layer to network layer [13]. Modeling stops at network layer because it is at this level that QOMET introduce the quality degradation using a wired network emulator.

In order to adapt these WLAN configuration value to Ethernet cable network, *dummynet* [14] is used. *Dummynet* applies varying conditions of network through IP queues at constant time intervals. Emulated robots manages these queues for every robots other than itself. And these queues limit the communication according to WLAN configuration values.

VI. EXPERIMENT INTEGRATION: RUNE

A. General description

In order to fulfill requirements of ubiquitous networked system on the StarBED2 testbed, the experiment-support software Rune is being currently developed. Rune provides an API set which controls experiment environments. The fundamental goal of Rune is to implement a test environment in which a number of "spaces" that emulate each experiment target

can work on either single or multiple nodes. Rune provides a reasonably abstracted interface for easily implementing emulation targets as spaces without much concern about the interaction between emulation nodes. Rune has the following roles: (i) experiment environment setup/cleanup and progress management; (ii) procedure invocation; (iii) interaction between spaces; (iv) time synchronization; (v) mutual exclusion.

Figure 5 shows the structure of an experiment implemented using Rune. The "Rune Master" module manages the configuration of each experiment, and controls the progress of the experiment. The execution of all spaces deployed on multiple nodes is initiated by Rune master via modules called "Rune Manager". The Rune manager is deployed on every emulation node and mediates communication between them through objects called "conduits". Spaces implementing emulation targets exist on emulation nodes in the form of shared objects, loaded dynamically by the Rune manager.

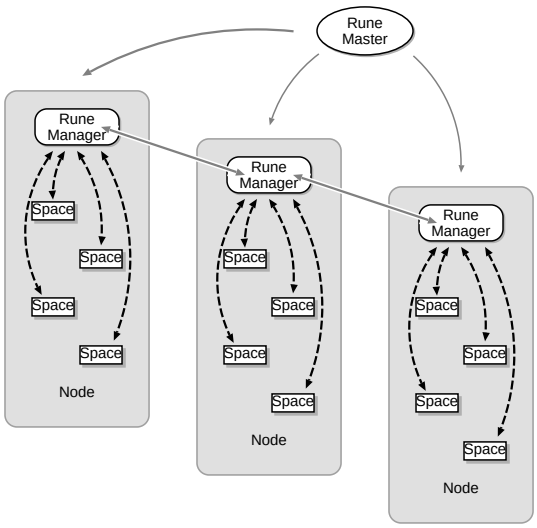

Fig. 6. Structure of experiments using RUNE

B. Emulation process

The emulation process performed by Rune takes place as follows. First of all Rune master is compiled with the experiment definition file, which includes the information regarding spaces and conduits. When run, Rune master sends the instruction "attach process" to the Rune manager executed on each node. A space then returns its entry point information to the Rune manager, which includes pointers to the available functions.

When Rune manager notifies Rune master of completion of the "attach" process, the latter indicates the initialize process of all spaces to Rune managers on each node. After the initialization of all spaces is finished, Rune master instructs the managers to start the iterated invocation of the "step" function, which represents the main body of a space. Accordingly, spaces start to execute the emulation step by step, and inform the corresponding Rune master of execution status. At the end of the experiment, Rune master starts the finalization process by notifying all nodes. Subsequently, spaces release the work area allocated in the initialization process.

VII. Experiment

A. Robot Definition

As explained in a previous section, autonomous mobile networked robots act in disaster or dangerous areas like rescue robots, and in office buildings or home like cleaning or assisting robots. These robots are equipped with motors to move around and accomplish their tasks. Table II describes the parameter definitions of the hardware and the application of autonomous robots of this system.

definition	parameter
shape of robots	circle
radius of robots	$1.0(m)$
velocity of robots	$0.5(m/sec)$
visual sensor range	$10.0(m)$
WLAN sensor range	depending on environment
degree robots can move	omnidirectional
time step of execution	$250(msec)$
priority of robots	same as robot number

TABLE II
PARAMETER DEFINITIONS OF ROBOT

B. Scenario Definition

We already accomplished real-time large-scale experiments with emulated autonomous robots (more than 100 robots and 80 obstacles) [16]. Each robot has a task to move from a source to a destination in an environment where lots of obstacles exist.

In this paper, we improve Map Manager to collision-avoidance with support moving obstacles. This functionality is important because human or other types of machine cannot communicate with autonomous robots, then autonomous robots should avoid possible collisions with such obstacles.

To confirm usefulness of robot communication, we test two types of experiments using above functionality, in scenarios as follows:

- **Non-Collaborative Scenario**
  No robot is equipped with any WLAN cards, it means that robots cannot communicate with each other. Then robots recognize other robots as moving obstacles. Tasks to move from a source to a destination are allocated to all robots.
- **Collaborative Scenario**
  All robots are equipped with WLAN card and can communicate each other. Robots exchange planned trajectories in real time and modify their trajectory depending on the priority.

Figure 7 describes the sources and the destinations of every robot. (0,0) means a relative coordinate. S# 7 ,D# 6 mean the source of the robot 7, the destination of the robot 6. The radius of the obstacles is also $1.0(m)$. These initial settings are very complicated to force robots to replan many times and check motion planning.

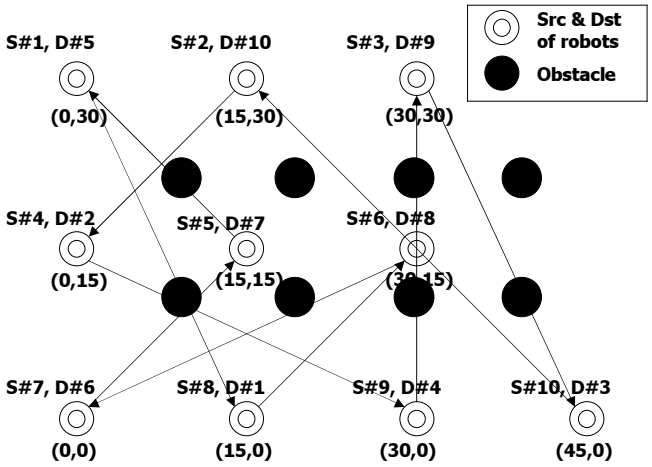

Fig. 7. The sources and the destinations of every robot

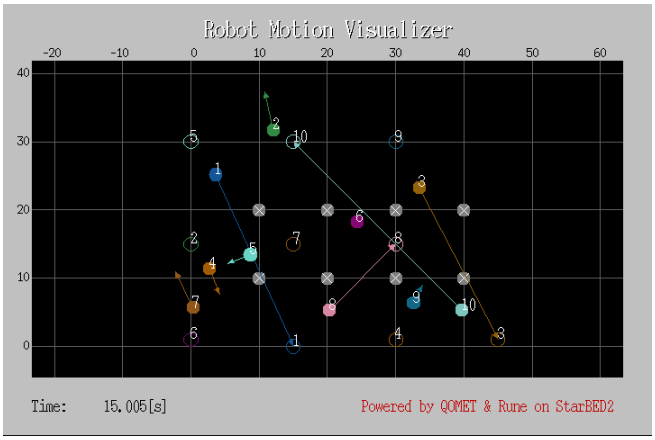

Fig. 8. Robot Motion Visualizer : ten robots and eight obstacles

C. Experiment Results

In the experimental results with every scenario we measure the time the robots take to move from their source to their destination, and the frequency of robot re-planning. Figure 8 describes the visualizer which show the trajectories of robots of ten robots scenario in real time at a certain moment.

First comparison is about the time to reach their destination. Figure 9 describes the comparison of Non-Collaborative Scenario and Collaborative Scenario. We did experiments in each case. All robots except 8 of Non-Collaborative Scenario take longer time to reach their destination, because the unexpected moving obstacles made robots change trajectories many times. For robot 8 only circumstances lead to a shorter time to reach destination. From our visualizer or log file, this situation can be confirmed.

Figure 10 clearly shows the difference. All robots except for 4 re-planned more frequently than in Collaborative Scenario. This emphasises the advantages of Collaborative motion planning for multiple robot scenarios.

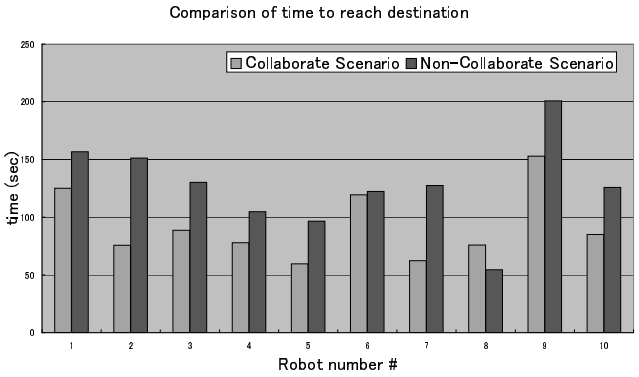

Fig. 9. Comparison of average time to reach destination

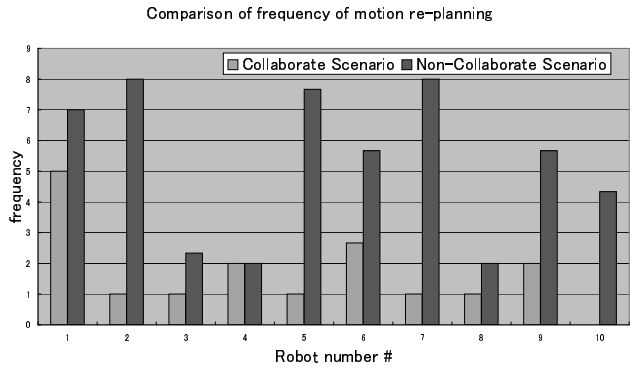

Fig. 10. Comparison of average frequency of motion re-planning

VIII. Discussion

The problems of experiments with software simulator and real hardware are mainly scale, accuracy, and cost. If the experiment takes much cost, researchers cannot expand scale. If accuracy is not sufficient, results are meaningless when applied to real situations. Following are the solutions of our experiment platform to above three problems.

- **Scale**
  In this experiment, the most essential point is to accomplish the execution of a large number of emulated autonomous mobile networked robots experiments. In such experiments with even 100 robots, they communicated and cooperated with each other and avoided collisions. And they all reached their destinations. This scale is much larger than previous researches.
- **Accuracy**
  The conditions of this experiment are emulated as if coming from a real environment. The sensing function information is handled by Map Manager. Map Manager and robots share these conditions, they are able to receive the hardware messages in real time. From network emulation point of view, robots communicate using real packets. These communication conditions are driven by WLAN emulation, and the limitations depend on emulated parameters. The robots cannot transmit packets if they are not in connection range. This system is also

running in real time.

- **Cost**
  Our platform especially saves costs on various points. Real autonomous mobile networked robots equipped with motors and PC and WLAN card are expensive. If researchers try to experiment large-scale robots, these costs become huge. Next are the costs of the time. Some experiments on real environment take much time to prepare the system environment. Software simulators of complex system execute many times longer than the running time of real experiment. And if researchers implement their method on software simulators, they have to create modules or codes depending on software simulators. These methods and programming languages are many different kinds, and they have to do this in unfamiliar ways. When using the system we propose, they just run their application on PCs.

IX. Conclusion & Future work

This research focuses on two main themes. First is the motion-planning algorithms of a large number of autonomous mobile networked robots. Second one is a real-time experiment platform for a large number of robots in unknown and dynamic environment.

We proposed an algorithm for motion planning based on PRM planner. This algorithm can be used both in a collaborative manner, in which robots exchange trajectories in real time as they move, or in a non-collaborative manner, in which other robots sinfully represent moving obstacles.

From the results of the experiments, we could confirm usefulness of our algorithm in real time. The robots which collaborate with each other can effectively select their trajectory by priority as needed, and the importance of the task which robots have affects their selection. The cooperation comes from communication between robots, therefore robots need common or generic communication method and protocol.

In non-collaborative execution robots can still reach their destination without having collisions, but it takes a larger time and more replannings.

The experiment on our system designed platform enabled to accomplish real time experiment with more than a hundred robots. On our way to test the system, some new aspects about the algorithm could be found. This came from our motion visualizer which can show the results in real time.

The comparison of ”Non-Collaborative Scenario” and ”Collaborative Scenario” shows positive effects of collaboration. A higher collaboration is needed to accomplish effective works, we will implement multi-hop MANET protocol on our system. MANET supply more connectivity and cooperation to system compared to the point to point communication used in the current experiments.

References

[1] B.R. Donald. *A search algorithm for motion planning with six degrees of freedom.* Artificial Intelligence, 31(3):295-353, 1987

[2] D. Hsu, J.C. Latombe, and R. Motowani *Path Planning in Expansive Configuration Spaces.* In Proc. IEEE Int. Conf. on Robotics and Automation, pages 2719-2726, 1997

[3] L. Kavraki, J.C. Latombe, R. Motwani, and P.Raghavan. *Randomized query processing in robot path planning.* In Proc. ACM Symposium on Theory of Computing, pages 353-362, 1995.

[4] L. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars *Probabilistic roadmaps for path planning in high-dimensional configuration space.* IEEE Transactions on Robotics and Automation, 12(4):566-580, 1996.

[5] EyeSim *EyeBot Simulator.* http://robotics.ee.uwa.edu.au/eyebot/doc/sim/sim.html

[6] RoboCup *The Robot World Cup Soccer Games and Conferences.* http://www.robocup.org/

[7] D. Hsu, R. Kindel, J.C. Latombe, S.M. Rock *Randomized Kinodynamic Motion Planning with Moving Obstacles.*

[8] StarBED *A Large Scale Experiment Network Environment.* http://www.starbed.org/

[9] Webots *Fast Prototyping and Simulation of Mobile Robots.* http://www.cyberbotics.com

[10] Stanford Aerospace Robotics Laboratory http://arl.stanford.edu

[11] K.Kant and S.Zucker. *Toward efficient trajectory planning: The path-velocity decomposition.* International Journal of Robotics Research, 5(39:72-89, 1986.

[12] T.Zheng, D.K. Liu, P.Wang *Priority based Dynamic Multiple Robot Path Planning* 2nd International Conference on Autonomous Robots and Agents December 13-15, 2004 Palmerston North, New Zealand

[13] R. Beuran, L.T. Nguyen, K.T. Latt, J. Nakata, Y. Shinoda *QOMET: A Versatile WLAN Emulator* IEEE International Conference on Advanced Information Networking and Applications (AINA-07), Niagara Falls, Ontario, Canada, May 21-23, 2007

[14] Rizzo, L *Dummynet FreeBSD network emulator.* http://info.iet.unipi.it/ luigi/ip_dummynet.

[15] J. Nakata, T. Miyachi, R. Beuran, K. Chinen, S. Uda, K. Masui, Y. Tan, Y. Shinoda *StarBED2: Large-scale, Realistic and Real-time Testbed for Ubiquitous Networks* TridentCom 2007, Orlando, Florida, U.S.A., May 21-23, 2007.

[16] R. Beuran, T. Okada, J. Nakata, T. Miyachi, K. Chinen, Y. Tan, Y. Shinoda *Network-enabled Sensing Robot Emulation* demonstration, 4th International Conference on Networked Sensing Systems (INSS07), Braunschweig, Germany, June 6-8, 2007, pp. 308