WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2011; **11**:1092–1106
Published online 9 October 2009 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/wcm.856

SPECIAL ISSUE PAPER

Architecture and protocol design for a pervasive robot swarm communication networks

Ming Li[1]*, John Harris[1], Min Chen[2], Shiwen Mao[3], Yang Xiao[4], Walter Read[1] and B. Prabhakaran[5]

[1] Department of Computer Science, California State University Fresno, CA 93740, U.S.A.
[2] School of Computer Science and Engineering, Seoul National University, Seoul 151-742, Korea
[3] Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849, U.S.A.
[4] Department of Computer Science, University of Alabama, Tuscaloosa, AL 35487, U.S.A.
[5] Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083, U.S.A.

ABSTRACT

There has been increasing interest in deploying a team of robots, or robot swarms, to fulfill certain complicate tasks such as surveillance. Since robot swarms may move to areas of far distance, it is important to have a pervasive networking environment for communications among robots, administrators, and mobile users. In this paper, we first propose a pervasive architecture to integrate wireless mesh networks and robot swarm networks to build a robot swarm communication network within the areas of special interest. Under the proposed architecture, one or more robots can get connected with a nearby mesh router and access the remote server, while a self-organizing mobile *ad hoc* network is formed within each swarm for communications among the robots. We then address and analyze many important issues and challenges. Finally, we describe our work to enable this architecture through a scalable algorithm for autonomous swarm deployment and ROBOTRAK, a socket-based-swarm monitoring and control toolkit. Extensive simulation results and demonstrations are presented to show the desirable features of the proposed algorithm and toolkit. Copyright © 2009 John Wiley & Sons, Ltd.

KEYWORDS
robot swarm; wireless mesh networks; network architecture

***Correspondence**
Ming Li, Department of Computer Science, California State University Fresno, CA 93740, U.S.A.
E-mail: mingli@csufresno.edu

1. INTRODUCTION

In the past few years, robot swarms have gained considerable attention. Many complicated artificial intelligent algorithms proposed to improve the correctness and efficiency of decision-making of a single robot. In addition, it has been found that using a team of robots has many advantages. First, single robot may not be adequate to fulfill some tasks, especially in complicated geographical environments where multiple obstacles exist on the path of the robots. Second, using a team of robots can achieve better fault tolerance for mission-critical applications such as surveillance and military battlefield where a single robot may be damaged or die due to, say, battery failure. Third, a swarm of robots is an appealing choice for accomplishing surveillance tasks, which would be otherwise difficult for human beings, since it is low risk and flexible. Furthermore, the idea of robot swarm is very natural since most complicated tasks such as moving around a big obstacle and cargo transportation require collaboration among the robots. Finally, with the cost of robots being significantly reduced to several hundred dollars, dispatching multiple robots is becoming a feasible scenario for a broader range of potential users.

Since robot swarms may move to areas of far distance, it is insufficient to deploy robot swarms directly without communications and coordination. Instead, it is highly important to have a pervasive networking environment for communications among robots, administrators, and mobile users. To address this important issue, we present a pervasive architecture for swarm communication in this paper. We then identify and examine several challenging issues for enabling this architecture, e.g., autonomous swarm deployment while keeping swarm's integrity and connectivity, swarm monitoring, and swarm control.

Swarm deployment is a challenging research problem. Once a team of robots are dispatched to an unknown area, it

is important for the robots to move independently but also collaboratively, such that the swarm coverage can be maximized. In the case of a closed area, such as a room, robots should be able to adapt themselves to the shape of the area. Due to the nature of their autonomous behavior, it is nontrivial to design an algorithm that can effectively maintain continuous swarm connectivity. In addition, the efficiency of a swarm highly depends on how fast it is deployed. If robots move slowly and randomly, the time overhead of deployment may be too high compared to the tight delay constraint of fulfilling certain tasks such as target tracking. Finally, mobile robots are battery operated, minimizing the moving distance leads to maximum energy efficiency and extended lifetime. Thus, it is desirable to devise a fast and energy efficient algorithm that deploys robot swarms with continuous connectivity and maximum coverage.

In a typical autonomous swarm deployment scenario, map information needs to be loaded in each robot before it explores an area. Alternatively, a team leader controls the movement of all other robots to achieve a wider coverage while maintaining connectivity. However, neither approach is suitable in practical scenarios where map information may not be available nor the robots may fail due to battery failure or outside attack. Actually, with the limited processing capability and battery power, low time complexity and communication frequency are preferred for swarm deployment algorithms. To mitigate the challenges in swarm communication, we propose a scheme that adopts a differentiated timer for each robot, such that each robot determines its action locally without consulting any central server or group leader inside the swarm. This feature makes the algorithm(s) scalable for various environments. Results show that this approach is fast and energy efficient compared to a random-movement-based strategy. In addition, this scheme can be extended to fulfill other tasks such as undeployment, coordinated movements, etc.

To enable convenient and flexible swarm monitoring, control, and coordination within the proposed architecture, we have designed and developed ROBOTRAK, a useful and secure toolkit for monitoring, control, and coordination of intelligent robotic swarms. Using TCP connections through a wireless mesh backbone, the ROBOTRAK server can exchange information with the robotic swarm reliably and continuously. For *monitoring purpose*, all the bots collect and report wireless signal strength, interference, neighboring bots list, and location information. For *control purpose*, a new robot can be dispatched to join a swarm or an existing robot in a swarm can be guided to move to a specific destination. For *coordination purpose*, the server maintains the connectivity of the swarm under various situations. Furthermore, in order to maintain network privacy and avoid outsider intrusion, *multi-security levels* and a *dynamic password mechanism* have been incorporated in the toolkit. Demonstrations show that the toolkit is very useful and effective for swarm monitoring.

This paper is organized as follows. Section 2 discusses related works in robot swarm algorithms, software, and quality of service (QoS) protocols. Section 3 proposes the robot swarm communication network architecture and discusses the design issues for swarm communication. Section 4 proposes a fast and energy efficient algorithm for autonomous swarm deployment. Section 5 presents ROBOTRAK, a toolkit for swarm monitoring, control, and coordination. Section 6 concludes this paper with future improvements.

2. RELATED WORK

2.1. Robot swarm coordination

Mondada *et al*. [1] and Dorigo *et al*. [2] proposed the concept of SWARM-BOT, a group of autonomous mobile robots called S-BOTs. S-BOTs have a particular assembling capability that enables them to connect physically to each other. With around 10–30 S-BOTs being interconnected, SWARM-BOT becomes much more robust than single bots and can handle difficult tasks even in hostile environment conditions. Poduri and Sukhatme [3] investigated the self-deployment of a mobile sensor network and proposed to maximize the area coverage of the network with a K neighbor constraint. The core idea of the scheme is to control the location of mobile sensors through two forces: F_{cover} that causes nodes to repel each other to increase their coverage and F_{degree} that causes nodes to attract each other to have more number of neighbors. A node stabilizes its position when the two forces are equivalent. Sheng *et al*. [4] proposed a multi-robot exploration model and studied the coordination among the robots in a group. A distance-based-bidding algorithm was proposed to enforce the coordination. Also, a map synchronization algorithm was proposed to minimize the amount of information exchange when two sub-networks merge.

2.2. Robot monitoring/control softwares

Several tools for robot monitoring and control have been available. Correll *et al*. [5] developed SwisTrack, a new software for tracking multi-unit robots and biological behaviors. SwisTrack is a framework that enables users to remotely monitor the movement of robots/insects. First, the video images of the swarm is captured with a camera and then segmented to identify the targets in the background. Then, these data are transmitted over TCP/IP to the user side for post-processing. SwisTrack significantly ease the task of small robot movements. However, it is only used for monitoring purpose and the use of camera limits its application to small areas. It is more desirable to have a framework that can be used in general scenarios such as outdoor surveillance. McLurkin *et al*. [6] proposed a human–robot interface to control iRobot Swarm that includes a team of 112 individual robots. The emphasis of the research is to interact with a group of robots through commands or input from game controllers, rather than the independent behavior of robots. MobileEyes [7] is a commercial graphical user interface

(GUI) software for driving or controlling individual robot by issuing commands and displaying the video captured by the moving robot. However, MobileEyes does not support the coordination and control of a robot swarm.

2.3. Qos support in wireless *ad hoc* networks

Xue and Ganz [8] proposed *ad hoc* QoS on-demand routing (AQOR), a QoS routing protocol with admission control enforced to support QoS in multi-hop *ad hoc* networks. Yang and Kravets [9] proposed to enforce admission control by considering the contention of all nodes within the carrier sensing area. However, expensive power consumption is required to obtain neighboring information in these schemes. Also, they did not consider link rate and traffic for bandwidth estimation. Chen and Heinzelman [10] proposed to calculate available bandwidth by local estimation and message exchange among neighboring nodes. Then, they adopted the result proposed by Li *et al*. [11] to account for intra-flow interference. However, this result is not accurate enough, especially when multi-rate is enabled. With multi-rate medium access (MAC), the distance of a link may be much smaller due to transmission with high data rate, resulting in more intra-flow interference. To consider both the mobility, multi-rate feature, and the multi-link interference, Li and Prabhakaran [12] proposed route available bandwidth (RAB) and route reliability (RR) metrics when admission control decision is made.

Awerbuch *et al*. [13] proposed medium time metric (MTM) for better route selection in *ad hoc* networks. Based on the assumption of complete interference, MTM chooses the routes with minimum accumulate medium time consumption. However, the assumption of complete interference makes it only appropriate for small networks. Also, link loss ratio is not considered in this work. Draves *et al*. [14] further considered multi-rate capability and combine link capacity and *ETX* for better performance in multi-radio, multi-hop wireless mesh networks (WMNs). They propose weighted cumulative expected transmission time (WCETT) to account for the interference among links that use the same channel. Also, WCETT calculation can be tuned for various different QoS objectives.

3. ARCHITECTURE FOR SWARM COMMUNICATION NETWORKS

3.1. Assumptions

Some basic assumptions are made on the design and implementation of the proposed architecture. From the networking aspect, we assume the following:

- All robots are equipped with wireless adaptors (such as WiFi) and can be accessed by administrators or mobile users through the wireless mesh backbone.
- Each wireless adaptor is assigned a unique IP address and port number for the communication with monitoring software running at either administrators or mobile users side. The software can connect to any robot by sending packets to the corresponding IP address and port number.
- All robots are equipped with global positioning system (GPS) so that they can exchange their locations with other robots or report the information to administrators or mobile users.
- Wireless access is ubiquitously available for *at least* one mobile robot. In case some robots cannot access any wireless mesh backbone due to moving outside of the radio coverage of all mesh routers, they can forward this information to the one that has wireless access to the Internet.

From the robot intelligence aspect, we assume that:

- All robots have the necessary intelligence to fulfill whatever jobs assigned individually.
- It is assumed that each robot has sonar installed such that it can detect boundaries and other robots to avoid collision.
- Each swarm may have one leader and at least one member for certain purposes. This leader can be either specified or determined by some voting algorithms.

It should also be noted that a minimum of two adaptors are required for a robot to be accessed from the Internet through access points and to communicate with other robots in the same swarm simultaneously. One adaptor is configured to operate in the *infrastructure* mode and connect to a nearby access point and the other adaptor is configured to operate in the *ad hoc* mode and connect to all the robots within the transmission range.

3.2. Architecture

In this section, we propose a new architecture for robot swarm communication networks. In this architecture, robots are clustered to one or multiple teams/swarms and each swarm can be monitored and controlled by some central servers through a wireless mesh backbone as well as the Internet. Within each swarm, a self-organizing mobile *ad hoc* network (MANET) is formed such that all robots are connected to all other robots despite of movements. Meanwhile, mobile users can also monitor swarms through mobile devices such as laptops and PDAs and can take action immediately based on certain information collected. The monitoring module may collect location, topology, and other related information. Images and videos are also allowed to be streamed from robots to servers or mobile users for appropriate decisions.

The architecture of the proposed robot swarm communication network is illustrated in Figure 1. Wireless mesh routers are deployed on the top of buildings, walls, or

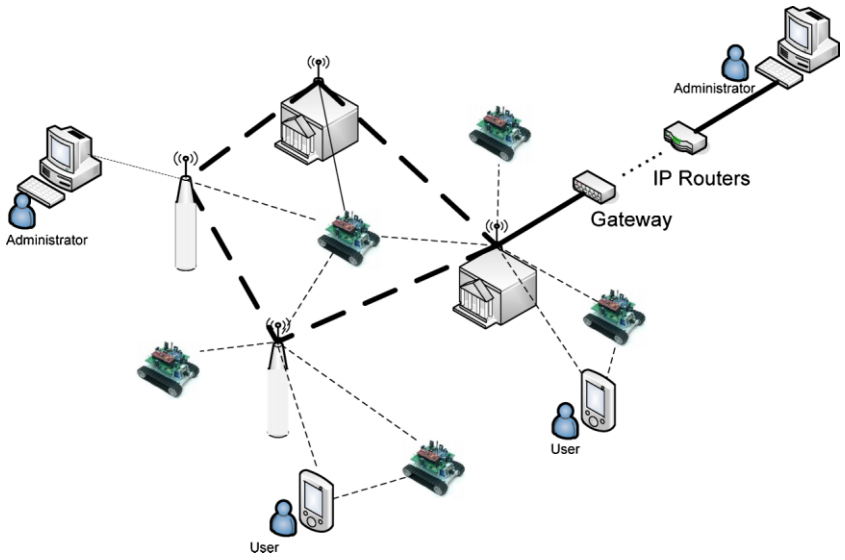


Figure 1. Architecture of the robot swarm communication network.

towers. Each mesh router consists of multiple antennas and can operate over multiple channels to improve the network capacity and coverage. Mesh routers form a wireless backbone, which is further connected to wired Internet through gateways and IP routers. One or more teams of robots are deployed. Each robot is equipped with one or more wireless adaptors for communicating with mesh routers or other robots, digital video camera, and GPS. Each robot swarm maintains continuous connectivity for all of its robots and periodically updates the collected information such as its location, picture, and video to some administrators that reside either locally or remotely. Meanwhile, users such as security guards drive around the area frequently, access wireless devices such as PDA or Palm PC, and monitor one or more robot swarms. If certain emergency is identified, a user can immediately take action. Due to the limited computational power of these devices on image/video analysis, a user may also get certain instructions from the administrators for appropriate actions. Altogether, the proposed architecture presents an interactive coordination framework for real-time monitoring, efficient collaboration, and fast reaction. Furthermore, since mesh routers, servers, robots, and PDAs are all inexpensive and require minimum human labor, such architecture is feasible and very cost effective.

With the proposed architecture, the robot swarms help fulfill the following critical tasks difficult for humans:

- *Continuous surveillance*: A swarm of robots can move around various areas in a non-stopped pattern, which significantly improves the information accuracy and the timeliness of actions taken by security guards. Some robots can be built at very small size and cannot be easily detected, thus enhancing the effectiveness of surveillance.
- *Information collection*: Through effective coordination, a team of robots equipped with GPS, video camera, and sensors can capture image/video periodically, recognize sensitive objects such as enemy and chemical/biological stuff, and report to administrators or security guards instantly.
- *Coverage inspection*: A robot can report to the administrator immediately if it cannot receive wireless signal from the mesh routers, which helps identify certain areas subject to security problems.

An important component of the proposed architecture is a wireless mesh backbone collocated with the robot swarm network. Basically, WMNs [15] and MANETs [16] have been investigated extensively. For WMNs, many schemes have been proposed to address issues such as wireless channel assignments, network capacity, and routing. For MANETs, many schemes have been proposed on mobility and QoS aware routing, energy efficient MAC protocols, and topology control. However, the major issues of this architecture are not on WMNs and MANETs, but on enabling the communications among administrators, users, and robot swarms, especially autonomous coordination among robot swarms and swarm monitoring and control. The mesh backbone actually provides an infrastructure to facilitate such communications with broadband capability.

The proposed architecture poses new challenges from both protocol design and system development aspects. On the one hand, how to coordinate robots that move randomly and continuously is a critical issue. Related issues include collaborative object recognition, deployment, etc. On the other hand, how to develop a software system to efficiently monitor, coordinate, and control robot swarms in a real-time manner is important for the system to be put in practice. If robots can be effectively monitored, critical information such as the location of each robot, the connectivity, and other information can be collected at administrators or mobile users. Based on the information from the swarm, the server

may perform certain control and/or coordination to assist the robots if necessary. In addition, streaming video from robots requires QoS support from the WMN backbone.

4. FAST AUTONOMOUS SWARM DEPLOYMENT ALGORITHM

Usually robots need to exchange certain information such as location and status for decision-making in order to fulfill certain complicate tasks such as surveillance. The key challenge of this type of coordination is *how to ensure swarm connectivity without the prior knowledge about robot mobility pattern, since robots can move freely based on its own intelligence and tasks*. Compared to centralized approaches where a central control robot coordinates the movement of other robots, distributed approaches exhibits several advantages such as better scalability, efficiency, and fault tolerance. Based on collected information from neighbors through periodic message exchanges, a robot can determine how to guide its own movement to achieve certain goals while maintaining effective connectivity.

In this section, we address the important issue of maintaining connectivity for swarm deployment. Our proposed approach adopts a differentiated timer such that each robot determines its action locally without consulting any central server or group leader in the swarm. Coupled with a movement adaptor and boundary detector, this approach achieves distributed, fast, energy efficient, and scalable robot swarm deployment for various environments. The contributions of the proposed algorithm are:

- Differentiated timer eliminates many unnecessary movements and thus achieves shorter stabilization phase and less movement distance.
- Network connectivity is always maintained within the swarm.
- The proposed algorithm works well with both open and closed areas with various shapes.

4.1. Autonomous deployment algorithm

In this section, we describe the proposed distributed and autonomous swarm deployment algorithm. To coordinate robot movements and avoid unnecessary movements, we propose a differentiated-timer-based technique. The duration of the timer is determined based on a node's neighbor information such that nodes on the edge of the swarm move first to accelerate the deployment process. Upon the expiration of a timer, a robot is pushed by most critical neighbors so that it can quickly move to farther locations. The speed of the movement depends on its distances from each neighbor but is chosen to make sure that the robot does not get disconnected from these neighbors. In addition, a special measure is taken to prevent certain robots in the center of the swarm from moving back and forth without stabilizing to a specific location.

In addition, we can reasonably assume that each robot is able to identify other peers in the same swarm. This can be easily realized by adding certain group identifier in the packet header. Whenever a robot receives a message from another robot, it will first check the group identifier and then decide whether or not to communicate with the sender. Note that GPS and wireless sensors/adaptors are now quite inexpensive and do not significantly increase the cost of robots. To enable communications among robots, the swarm network can be configured to operate in the *ad hoc* mode.

The proposed algorithm consists of the following four components:

- *Differentiated timer*: Due to the autonomous nature of the wireless enabled robots, each robot follows a wait-and-move mobility pattern where an action is taken after a self-tuning timer expires since the last action.
- *Movement adapter*: Each robot determine which direction and how fast it should move based on the distance from critical neighbors, transmission range, etc., while maintaining continuous connectivity within the swarm.
- *Oscillation detector*: After the swarm has been deployed, there is still a residual oscillation of certain nodes. Due to the repulsion forces of the various walls, some robots may be forced to go back and forth for a long time. Detecting and handling this wobbling effect helps to maintain the battery life of involving robots.
- *Boundary detector*: Each robot needs to determine when a boundary is close and then continue gliding on the right direction.

We discuss the operation of the three key components in the following.

4.1.1. Differentiated timer.

Once a robot swarm arrives at a specific area, robots should start to scatter out to cover the area. Now, the issue is who should move first. If all robots move at the same time, then it is possible that many unnecessary movements will occur. Figure 2 illustrates this situation. The best situation will be that robots at the border of the swarm move first, followed by movements of centering nodes. For this purpose, we define a differentiated timer for each robot i, denoted as T_i, which indicates when will a robot starts to move. T_i is calculated as

$$T_i = \sum_{j=1}^{M_i} \frac{1}{d_{i,j}} = \sum_{j=1}^{M_i} \frac{1}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \qquad (1)$$

where $d_{i,j}$ is the distance from robot i to a neighbor robot j. (x_i,y_i) and (x_j,y_j) are the x and y coordinates of robot i and j, respectively. Here, a neighboring robot is defined as another robot that is located within the transmission range of robot i such that it can communicate with i by exchanging information such as location. M_i is the number of such

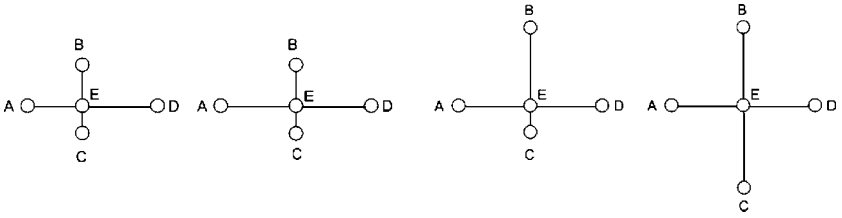


Figure 2. Illustration of swarm deployment.

neighbors of robot i. Clearly, if robot i is a center node (e.g., node E in Figure 2), its differentiated timer will have a larger value due to the fact that it has more closer neighbors than those bordering nodes. By comparison, the timer of a bordering node will be set to a smaller value due to less neighboring nodes. Thus, bordering nodes usually move before centering nodes, which reflects the nature of the scattering behavior (e.g., fish group or bird flock). In addition, since location information is exchanged with neighbors, each robot can easily calculate its own timer locally.

4.1.2. Movement adaptor.

Once the timer value is calculated, a robot has to determine the speed and direction of its movement when its timer expires. To make sure that a robot does not move too far away unnecessarily, we introduce a *pushing speed* $v_{i,j}$, which is defined as

$$v_{j,i} = a(D_0 - d_{i,j}) \tag{2}$$

Where a is a constant (set as 0.2 in our simulations) and D_0 is the threshold distance (set as 80 per cent of R_{TX}) and $D_0 < R_{\mathrm{TX}}$, the radio communication range. Thus, pushing speed can be considered as how fast a robot i should be pushed based on its distance to another robot j. The smaller $d_{i,j}$ is, the faster robot i is pushed. When $d_{i,j}$ is close to D_0, robot i should move very slowly to avoid link breaking. Then, since both robots i and j may be pushed by each other at the same time, it is important that neither robot i nor j can move more than $(D_0 - d_{i,j})/2$. Thus, each robot moves for $1/2a$ seconds such that the threshold is not exceeded.

Let a robot i be pushed by N neighbors (illustrated in Figure 3) and the direction of the combined pushing speed is represented as (ux, uy) such that $ux^2 + uy^2 = 1$. Then, we have that

$$ux = \sum_{j=1}^{N} \frac{x_i - x_j}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \tag{3}$$

$$uy = \sum_{j=1}^{N} \frac{y_i - y_j}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \tag{4}$$

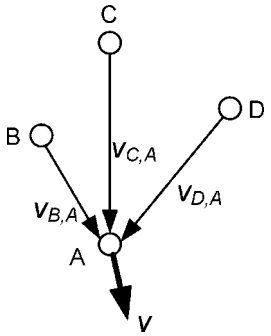


Figure 3. Calculation of pushing speed.

Then, we define (vx, vy) as the representation for the combined pushing speed, we have that

$$vx = \min_{1 \le j \le N} v_{j,i} \cdot ux \tag{5}$$

$$vy = \min_{1 \le j \le N} v_{j,i} \cdot uy \tag{6}$$

It should be noted that the reason for using the minimum pushing speed is to make sure the robot i does not move beyond the farthest pushing neighbor. Thus, it is not good to let too many neighbors to push robot i since doing so will make the moving speed too slow. Therefore, we choose to limit the number of pushing neighbors to a small value. Actually, this number is set to the desired node degree we would like to maintain. In this paper, we set the minimum node degree to 3. From the simulations, we found that there is high probability that the network is strongly connected with this configuration. We will study the effect of node degree to swarm connectivity in the future.

4.1.3. Oscillation detector.

In Figure 4, we demonstrate a scenario where the two center robots will oscillate. The robots on the outside perimeter push the two robots toward the center, but the two robots in the center try to get away from each other. The result is an oscillation between the two center robots trying to push each other apart while the robots on the outside perimeter push the two center robots closer together. The oscillation can lead to battery failure very quickly because the robots will not stop moving even though they are close to their ideal final positions.

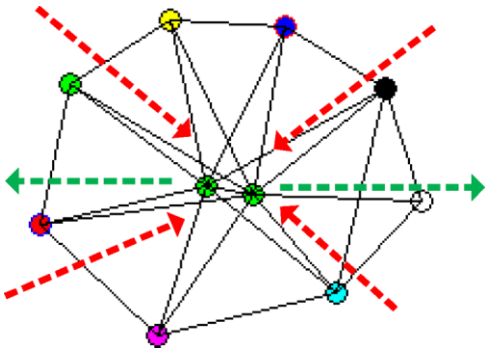


Figure 4. Illustration of node oscillation.

To address this problem, we propose a solution as follows. All robots keep a history of their last six movements. If a robot detects that it changes its moving direction more than twice, it identifies itself as wobbling and then reset its timer to twice the value of its last used timer. The more times a robot is identified to be oscillating, the greater the timer will be. In this way, we can ensure that the wobbling effect is reduced to a minimum.

However, it should be noted that for the function to work properly, six movements must have already occurred. Also, even if a robot is found to be oscillating, it will attempt to move again after its timer has expired. There may be a case where a robot will be able to move in the appropriate direction after it has been classified as an oscillating robot. In that case, the wobbling status flag is removed after it has successfully performed six movements in the same direction. Meanwhile, the robot will once again be able to move using the regular differentiated timer according to Equation (1) without any additional delay.

4.1.4. Boundary detector.

We have also considered the deployment over a closed area. In this case, whenever a robot moves close to boundaries such as a wall, it simply glides along the wall based on which direction it is being pushed. However, the speed will be much lower to avoid the collision with the boundary. In case a robot hits a boundary and gets stuck at a corner, it backs off and randomly chooses a direction to glide.

4.1.5. Discussions.

The major part of the proposed algorithm is for maintaining swarm connectivity. It can be extended to address following issues with appropriate modifications:

- *Swarm deployment with obstacles*: In this case, the boundary detector module can be modified such that the gliding may proceed beyond corners of the obstacles.
- *Swarm undeployment*: After the job is fulfilled, robots will gather together and may move to another location to be deployed again. The algorithm can be easily implemented by changing the 'push' operations in the algorithm to 'drag' operations.
- *Movement-based-swarm coordination*: In this case, undeployment can be first performed followed by a dragger-pusher-based-spanning tree rooted at the swarm leader.

Even though we calculate speed and direction based on the location information, the actual location information may not be accurate due to certain errors. In our algorithm, we can easily compensate location errors by reducing the moving speed such that nodes do not move outside of the transmission range of the neighbors.

For very small robots, it may be difficult to equip GPS with it due to various limitations. In this case, it is difficult to obtain location information. However, with wireless communication, it is still possible to collect some useful information to guide the swarm movements. We will investigate this issue in our future work.

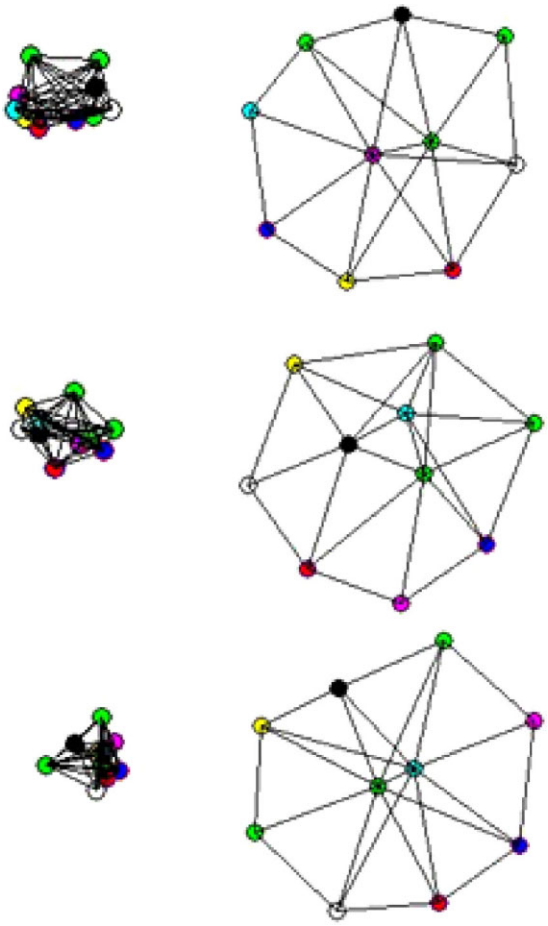


Figure 5. Deployment results in open area. Left: initial positions. Right: final positions. Minimum node degree is 3.

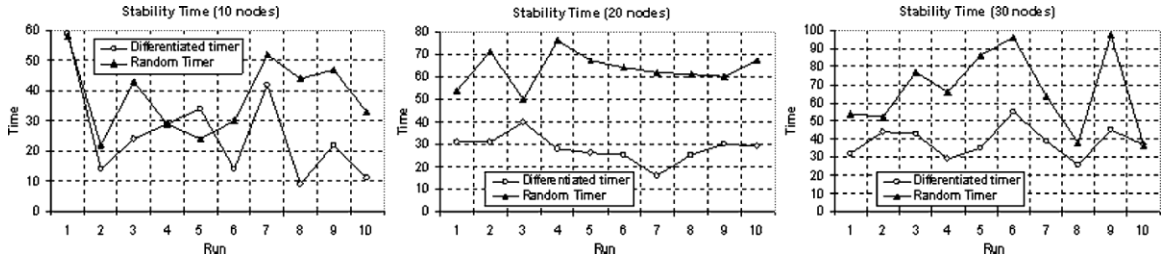


Figure 6. Time (in seconds) taken for a deployment to stablize.

4.2. Performance evaluation

We have conducted simulations to evaluate the performance of the proposed algorithm. We focus on two aspects: (i) How long it takes for a robot to reach a stable deployment status where the movement speed of robots is negligible. (ii) Whether or not continuous connectivity is maintained in both open and closed areas. The transmission range is set to 100 m, with a corresponding threshold distance of 80 m. The differentiated timer of each robot is calculated according to Equation (1). In our simulations, this value varies from 35 ms to over 1 s, depending on the robot distances. For comparison purpose, we also conduct the same scenarios but assign each robot a randomly generated timer within the range from 0 to 2 s. In the remainder of this section, we refer to it as random timer approach.

4.2.1. Simulation in open areas.

For each scenario, we conducted 10 experiment runs with robots initially positioned randomly around the center of a 500 m × 500 m open space. Figure 5 shows the topologies of the robot swarms before and after the deployment with the proposed differentiated timer. It can be seen that connectivity is well maintained with different initial positions. Due to certain randomness, the final locations are slightly different.

Figure 6 shows the deployment speed by varying robot number from 10 to 30. We can see that with larger robot number, the differentiated-timer-based scheme achieves much faster deployment than the random timer approach. In Figure 7, it can be observed that exploiting random timer yields significantly longer moving distance than the differentiated timer. Since power consumption for physical movement of robot is proportional to its moving distance, the proposed scheme achieves better energy efficiency due to the reduced moving distance. Furthermore, from Figure 8, we can see that the stability time for different robot number is quite similar compared to random-timer-based scheme, indicating that the proposed scheme has better scalability.

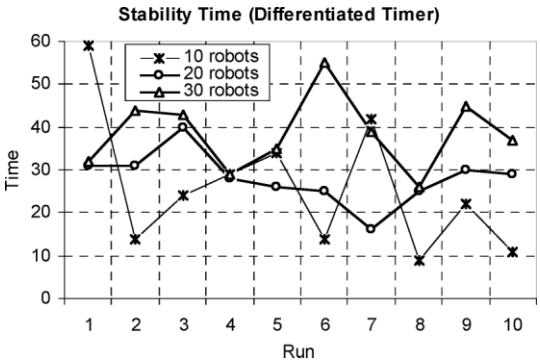


Figure 8. Stability time (in seconds) *versus* number of robots.

4.2.2. Simulation in closed areas.

We have also considered deployment over closed areas. Robots are initially positioned randomly around the center of the closed space. Figures 9 and 10 show the simulated deployment topology when the closed areas are rectangle

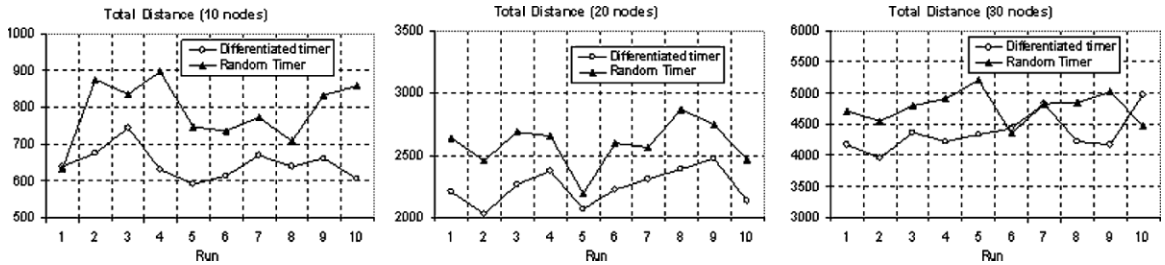


Figure 7. Total moving distance (in meters) *versus* number of nodes.

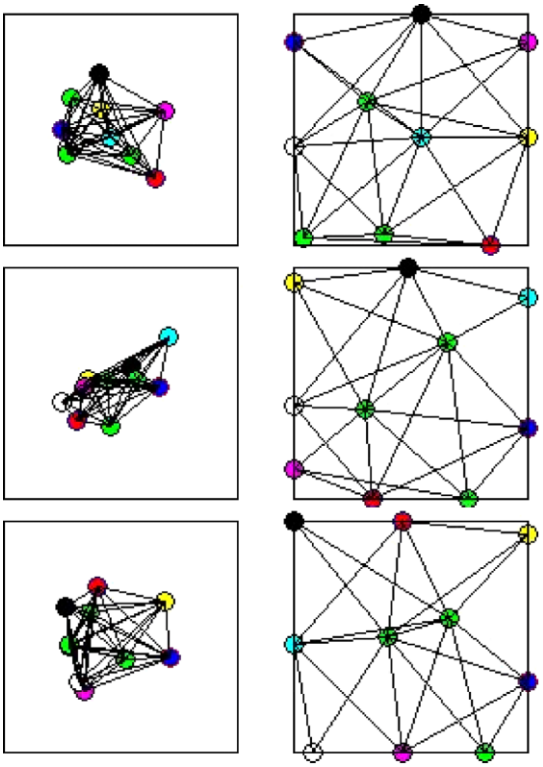


Figure 9. Deployment results in rectangular area. Left: initial positions. Right: final positions. Minimum node degree is 3.

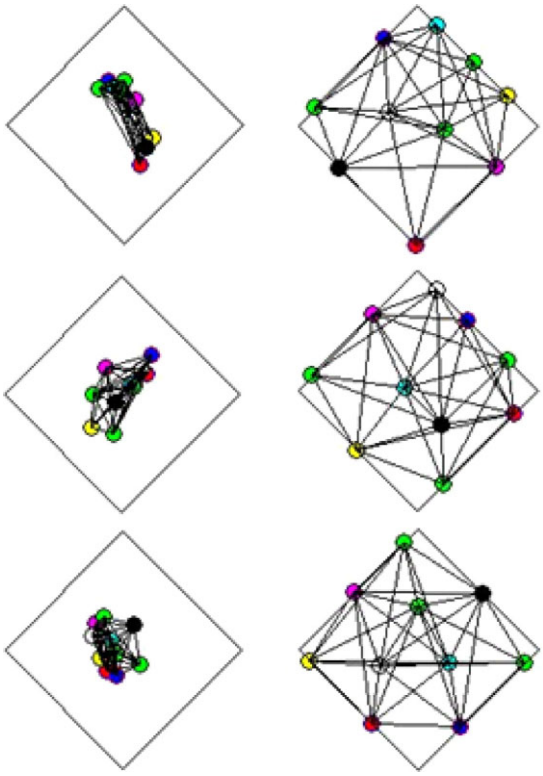


Figure 10. Deployment results in rhombus area. Left: initial positions. Right: final positions. Minimum node degree is 3.

and rhombus, respectively. For the square area, all sides have length of 125 m. For rhombus area, the distance between diagonal corners is 150 m. In this case, whenever a robot moves close to a boundary such as a wall, it simply glides along the boundary based on which direction it is being pushed. However, the speed will be much lower to avoid collision with the boundary. In both scenarios, a successful deployment is achieved.

5. ROBOTRAK FOR SWARM MONITORING, CONTROL, AND COORDINATION

To meet the software needs arising from the wide deployment of network enabled robot swarms, we have designed and developed ROBOTRAK, a secure software for monitoring, control, and coordination of intelligent robotic swarms. Using TCP connections through wireless medium, the ROBOTRAK server can exchange information with the robotic swarm reliably and continuously. For *monitoring purpose*, all the bots collect and report wireless signal strength, interference, neighboring bots list, and location information. For *control purpose*, a new robot can be dispatched to join a swarm or an existing robot in a swarm can be guided to move to a specific destination. For *coordination purpose*, the server maintains the connectivity of the swarm under various situations. Furthermore, in order to maintain network privacy and avoid outsider intrusion, *multi-security levels* and a *dynamic password technique* were implemented.

The major advantages of ROBOTRAK are:

- flexibility with comprehensive features on robot monitoring, control, and coordination;
- user friendliness with a nice GUI;
- multi-level security at both application layer and transport layer; and
- robustness with the implementation of reliable socket communication.

ROBOTRAK can run on any server with Internet connection and requires the robots in a target swarm to be equipped with wireless adaptors (such as WiFi) and GPS. All the ROBOTRAK messages can be transmitted/received through typical wired/wireless multi-hop network connections. The software was implemented with *Microsoft Visual Basic* and can be modified to provide web-based versions. All the designed features and functionalities have been implemented with user friendly GUI and extensively tested in various scenarios such as multiple robots and multiple instances of running the ROBOTRAK software. Results show that ROBOTRAK is user friendly and can greatly help to monitor, control, and coordinate robotic swarms timely,

Wirel. Commun. Mob. Comput. 2011; **11**:1092–1106 © 2009 John Wiley & Sons, Ltd.
DOI: 10.1002/wcm

effectively, and efficiently. We introduce the key modules of ROBOTRAK in the following.

5.1. Monitoring module

The monitoring module basically analyzes the messages from each robot and decides the desired information the administrator intends to track such as physical location. Since connectivity within the swarm is critical for the performance of a swarm, each robot should report its neighboring robots within the same swarm. This neighboring information can be obtained through communication among mobile robots in a self-organized wireless *ad hoc* network. In summary, each robot will report following basic information to the server for monitoring purpose:

- *Physical location*: The coordinates obtained from the GPS system is sent from each robot to the server.
- *Neighbor list*: Whenever a robot receives or overhears messages from another robot in the same swarm, it will add the robot to its neighboring list. This list is periodically refreshed to remove old neighbors and add new neighbors.

It should be noted that more information may be required for monitoring, depending on the specific application. The applications of ROBOTRAK are somewhat analogous to the case of wireless Internet service providers (ISPs) or network administrators of commercial and military wireless networks. There, the most important issue is how to place mesh routers such that all targeted areas are covered sufficiently and efficiently, i.e., all areas are covered with good signal strength with minimum number of mesh routers. Typically, some areas may experience high interference and some areas may experience weak or no signal, due to inappropriate installation of many mesh routers. Furthermore, it is possible that an outsider may install their mesh routers and thus has significant interference with the commercial wireless network. In the case of ROBOTRAK, a robot team can be dispatched to report the *WiFi signal strength* and *SSID* of all nearby WiFi access points for potential intrusion detection.

5.2. Control module

The control module is designed to only provide assistance to the swarm in certain unusual situations. As mentioned earlier, the control from the server is not supposed to replace the basic operations of each individual robot or the whole swarm, but to help them. In ROBOTRAK, we consider three situations where outside assistance is desirable:

- *Robot joining*: Dispatching a new robot to join the swarm by providing the destination. While the robot is moving toward the team, the center location of swarm is periodically updated for guidance.
- *Robot leaving*: Requesting one robot in a swarm to leave the swarm. The robot may go back to storage, or go to a location specified by the server.
- *Swarm movement*: Requesting the whole swarm to move to another location to perform some other tasks. There may be several reasons for this. The administrator may identify certain dangerous situations at the current location and thus wants the robots to move immediately. Or the administrator may have another urgent task for the swarm to fulfill.

5.3. Coordination module

Similar to the control module, the coordination module is also designed to only provide minimum assistance for swarm connectivity. The basic coordination is still performed by the swarm itself independently with certain intelligent algorithms. However, with the convenient monitoring module, ROBOTRAK can easily identify certain swarm partition and acts promptly. In ROBOTRAK, we consider the following three representative situations:

- *Isolation of a regular robot*: If one robot is isolated from the rest of the team due to various reasons such as entering a building or the other side of an obstacle, it will report its neighbor list as empty to the server. Based on this information, the server can quickly direct the robot to move toward the swarm with higher speed.
- *Isolation of a leader robot*: We assume that the leader should guide the whole swarm to move. However, in case the leader gets isolated due to similar reasons for a regular robot, the server can request all other robots to move to the leader and maintain the swarm connectivity.
- *Swarm partition*: In the worst case, swarm may be divided to multiple isolated partitions and cannot perform any meaningful task for quite some time.

We can design a simple algorithm for fixing swarm partitioning. In this algorithm, it is assumed that every robot has a unique identifier from 1 to N. When the server receives the neighboring information from each robot, it will run the algorithm. Initially, there are N partitions with robot i included in partition i. Then, two partitions P_i and P_j are merged to P_i if $i < j$ and robot j is in the neighboring list of robot i. Thereafter, this merging will continue until all partitions have been examined. Finally, if there is only one partition that includes all robots in the swarm, the server knows that the swarm is connected. Otherwise, the swarm identifies a major partition that includes the maximum number of robots, get its center location, and requests robots in all other partitions to move toward that location.

5.4. Security enforcements

There are two possible security issues with robot swarm monitoring, control, and coordination. On the one hand, the

ROBOTRAK administrator should possess the appropriate permission for certain functionalities. On the other hand, it is important to identify malicious outsiders who may steal the passwords and try to control, hack, and eventually damage the swarm. To address these issues, we enforce three security policies.

5.4.1. Multi-level password checking.

We classify the access levels of administrators to *general*, *high*, and *highest*, which are determined by the passwords provided. Then, we restrict their permissions as follows:

- *General*: Under this access level, the software can *only monitor* the basic information of the swarm and cannot issue control or coordination commands to any robot.
- *High*: Under this access level, the software can *monitor and coordinate* the robot swarm, and *watch the image/video* from each individual robot, if applicable.
- *Highest*: Under this access level, the software can perform *all functionalities including robot control*.

With this multi-level password checking, we can avoid unauthorized interruption to the swarm. This policy can be changed according to the application and user requirements.

5.4.2. Dynamic password update.

One special situation is that an outside intruder may get the password somewhere and try to maliciously connect and hack a robot that has been connected with an authorized server running ROBOTRAK. In this case, we adopt a simple approach to avoid simultaneous connection of an individual robot from two or more administrators with the highest access levels. Each time after an administrator with the highest access level gets connected to a robot with a predefined password, it will assign a new password to the robot periodically (every minute). So, when an outsider attempts to connect to the same robot with the predefined password, the robot will reject the connection and is still safely under the control of the authorized administrator.

Of course, it is possible that a hacker may get connected to the swarm with the highest access level and prevents the authorized user from connecting. In this case, the authorized user will know the situation and take certain actions. However, given the assumption that ROBOTRAK is running continuously at the authorized server, this does not occur frequently.

5.4.3. Server information reporting.

To further enforce security, it is desirable to prevent those hackers who are not trying to control the swarm, but simply monitor robots positions. To address this potential vulnerability, each server running the software will require every connected robot to report all the servers it is communicating with. Then, authorized users can identify possible intruders by checking the reported servers.

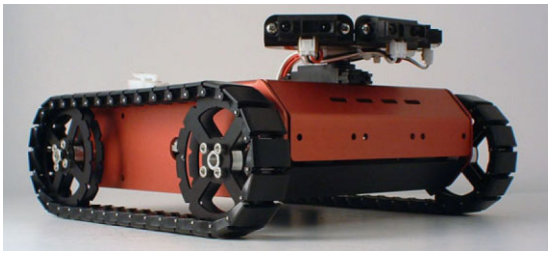


Figure 11. The Traxster Robot.

5.5. Feature demonstration

In this section, we demonstrate the features of ROBOTRAK through distributed emulation. In this emulation, a robot agent program is running to emulate each individual robot. We will test ROBOTRAK with Traxster Robot (Reference [17] and Figure 11) that will run on Window CE system and is equipped with multiple WiFi adaptors and GPS receivers. However, since these agent programs have exactly the same message format and operating system requirements as real robots, the simulation does provide an accurate demonstration of the features of ROBOTRAK.

5.5.1. Swarm monitoring.

In this section, we illustrate the swarm monitoring feature. Figure 12 shows the monitored topology and sonar ranges of selected robots. We can see that the swarm is strongly connected. The neighborhood is established when two robots are within 250 m from each other. In real scenarios, each robot can send broadcast message to discover its actual neighbors and report the list to the administrator for displaying the topology information. Also, the sonar can detect the distance from obstacles such as walls and provide specific information about the environment.

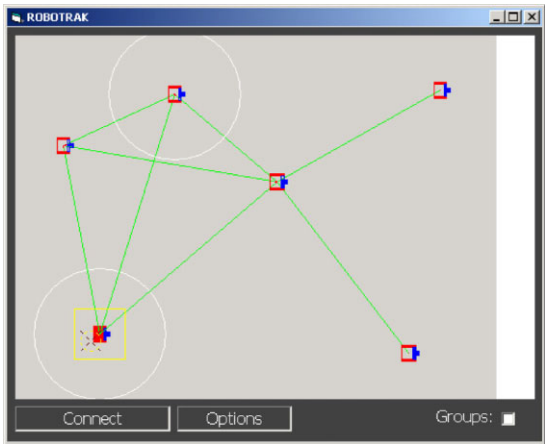


Figure 12. Illustration of swarm monitoring.

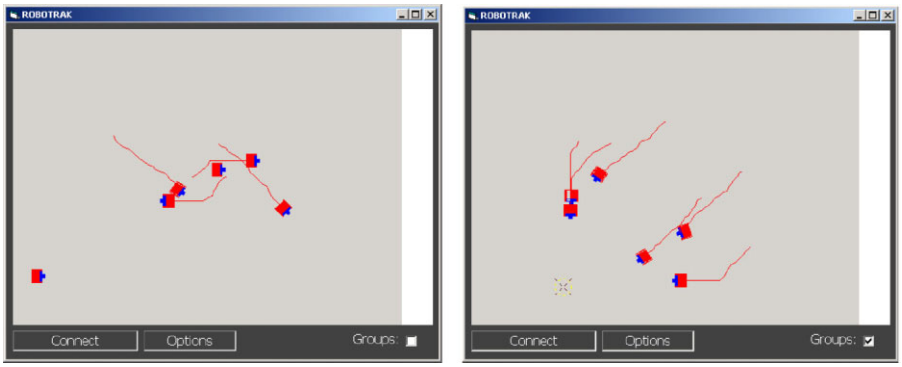


Figure 13. Illustration of swarm control. Left: robot movement. Right: swarm movement.

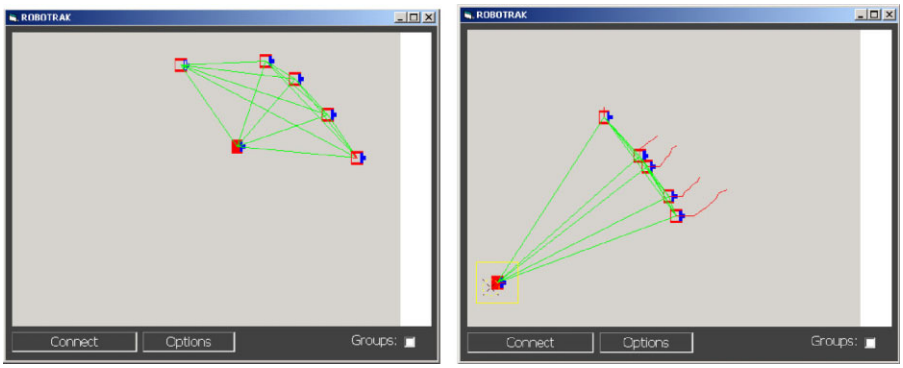


Figure 14. Leader (dark color) movement coordination. Left: before movement. Right: after movement.

5.5.2. Swarm control.

In this sub-section, we demonstrate two types of control: *robot movement* and *swarm movement*. Figure 13 shows the robot movement where each individual robot is separately selected and directed to a specific destination. The tails shows the trail of the robot in the last few seconds, which is inspired by SwisTrack [5]. When the robot arrives at the destination, it will stop and the trail gradually disappears. Figure 13 also shows the swarm movement. When the 'Groups' option is checked, the swarm can be selected by entering 'CTRL + A' command from the keyboard and directed to move to around the same destination by right clicking the mouse at the specific location on the screen.

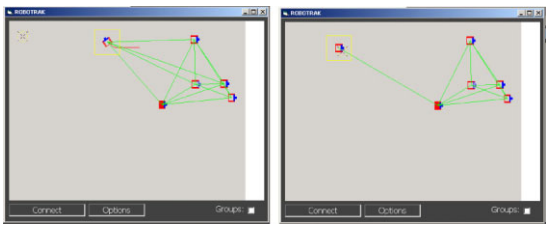


Figure 15. Swarm member movement coordination. Left: before movement. Right: after movement.

5.5.3. Swarm coordination.

To demonstrate the feature of swarm coordination, we consider two situations: (i) *Leader movement*, where all other robots need to follow the leader to avoid swarm partitioning. (ii) *Member movement*, where a specific robot cannot move too far away from the rest of the swarm. Figure 14 illustrates the leader (dark colored) movement. Initially, all the swarms are within the range of the leader. When the leader moves toward the left bottom corner, all other robots follow the leader and moves toward the same direction to maintain the swarm connectivity. Figure 15 illustrates the coordination on a swarm member movement. It can be seen that as the robot moves toward the left, more and more other robots get disconnected from it. However, after it moves out of the range of all other robots, it is 'dragged' back to a destination within the transmission range of the closest robot by the software with the coordination algorithm, in order to maintain connectivity of the swarm. More demonstrations of the swarm coordination are omitted due to limited space.

6. CONCLUSION

How to monitor, control, and coordinate robot swarms in real time is a challenging and imperative issue. In this paper, we proposed a pervasive network architecture to integrate

wireless mesh backbone and wireless enabled robot swarms to facilitate communications among robots, administrators, and mobile users. Then, we identified several challenging issues, especially autonomous swarm deployment and swarm monitoring and control toolkit development and presented our solutions. Our efforts have established a solid basis for implementation and performance improvement of the proposed architecture. In the future, we will extend the proposed distributed autonomous swarm deployment algorithm to address various issues such as swarm movement coordination and swarm coordination. Furthermore, we will refine the functionalities of the proposed ROBOTRAK toolkit and incorporate more desirable features.

ACKNOWLEDGEMENTS

Ming Li's work is supported by the Provost Research Activities Awards of California State University Fresno. Min Chen's work is partially supported by KRCF and by the Ministry of Knowledge Economy, Korea, under the Information Technology Research Center support program supervised by the IITA (Grant Number IITA-2009-C1090-0902-0006). Shiwen Mao's research in part by the NSF under Grant ECCS-0802113 and through the NSF Wireless Internet Center for Advanced Technology (WICAT) at Auburn University. Prof. Yang Xiao's work is supported in part by the US National Science Foundation (NSF) under the Grant Numbers CNS-0737325, CNS-0716211, and CCF-0829827. Dr. Prabhakaran's work is supported in part by US Army Research Office Grant 48645-MA and NSF under Grant Number 0237954 for the project CAREER: Animation Databases.

REFERENCES

1. Mondada F, Pettinaro G, Guignard A, *et al*. SWARM-BOT: a new distributed robotic concept. *Autonomous Robots (Special Issue on Swarm Robotics)* 2004; **17** (2–3): 193–221.
2. Dorigo M, Tuci E, Groß R, *et al*. The SWARM-BOTS project. *Swarm Robotics: SAB International Workshop, Lecture Notes in Computer Science*, Vol. 3342, 2004; 31–44.
3. Poduri S, Sukhatme GS. Constrained coverage for mobile sensor networks. *IEEE International Conference on Robotics and Automation*, 2004; New Orleans, LA, U.S.A.
4. Sheng W, Yang Q, Tan J, *et al*. Distributed multi-robot coordination in area exploration. *Robotics and Autonomous Systems* 2006; **54**(12): 945–955.
5. Correll N, Sempo G, Lopez de Meneses Y, *et al*. SwisTrack: A Tracking Tool for Multi-Unit Robotic and Biological Systems, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* 2006; 2185–2191.
6. McLurkin J, Smith J, Frankel J, *et al*. Speaking Swarmish: human–robot interface design for large swarms of autonomous mobile robots. *Proceeding of AAAI Spring Symposium* 2006; 72–75.
7. MobileEyes: www.mobilerobots.com/MobileEyes.html.
8. Xue Q, Ganz A. *Ad hoc* QoS on-demand routing (AQOR) in mobile *ad hoc* networks. *Journal of Parallel Distributed Computing* 2003; **63**: 154–165.
9. Yang Y, Kravets R. Contention-aware admission control for *ad hoc* networks. UIUCDCS-R-2003–2337, 2004.
10. Chen L, Heinzelman W. QoS-aware routing based on bandwidth estimation for mobile *ad hoc* networks. *IEEE Journal on Selected Areas of Communication* 2005; **23**(3): 561–572.
11. Li J, Blake C, Couto D. S. J. D, *et al*. Capacity of *ad hoc* wireless networks. *Proceedings of the 7th annual international conference on Mobile computing and networking*. ACM: Rome, Italy; 2001; 61–69.
12. Li M, Prabhakaran B. On supporting reliable QoS in multi-hop multi-rate mobile *ad hoc* networks. *Proceedings of the First IEEE International Workshop on Next Generation Wireless Networks (WoNGeN)*, 2005; Goa, India.
13. Awerbuch B, Holmer D, Rubens H. High throughput route selection in multi-rate *ad hoc* wireless networks. *Proceedings of WONS*, 2004; Madonna di Campiglio, Italy.
14. Draves R, Padhye J, Zill B. Routing in multi-radio, multi-hop wireless mesh networks. *Proceedings of ACM MobiCom*, 2004; Philadelphia, PA.
15. Akyildiz IF, Wang X, Wang W. Wireless mesh networks: a survey. *Computer Networks and ISDN Systems* 2005; **47**(4): 445–487.
16. Toh CK. *Ad Hoc Mobile Wireless Networks: Protocols & Systems*, Prentice Hall PTR: Upper Saddle River, NJ, USA, ISBN 0130078174, January 2002.
17. Traxster Robot: www.roboticsconnection.com/pc-15-3-traxster-robot-kit.aspx.

AUTHORS' BIOGRAPHIES

Ming Li has been a faculty in the Department of Computer Science, California State University, Fresno, since August 2006. He received his M.S. and Ph.D. degrees in Computer Science from The University of Texas at Dallas in 2001 and 2006, respectively. His research interests include QoS strategies for wireless networks, robotics communications, and multimedia streaming over

wireless networks. He is the recipient of the Best Student Paper Award in the First IEEE International Workshop on Next Generation Wireless Networks (WoNGeN'05). He has served as the TPC co-chair of The first International Workshop on Pervasive Computing Systems and Infrastructures (PCSI 2009), The Second International Workshop on Sensor Networks (SN'09), Multimedia Networking track in ICCCN'08, the IEEE International Work-shop on Data Semantics for Multimedia Systems and Applications (DSMSA'08), and the third IEEE International Workshop on Next Generation Wireless Networks (WoNGeN'08). He is a guest editor of a special issue on Recent Advances in Sensor Integration for International *Journal of Sensor Networks (IJSNet)*, a special issue on Data Semantics for Multimedia Systems in Springer Multimedia Tools and Applications, and a special issue in *Journal of Multimedia*. He has served the technical program committees of several international conferences such as ICC, ICCCN, ISM, ROBOCOMM, VTC, and ChinaCom. He is a member of ACM and IEEE.

John Harris was born in Greece and came to the United States to pursue a higher education. He got a B.S. degree in Computer Science from California State University Bakersfield in 2005 and is currently pursuing his Masters degree in Computer Science at California State University Fresno. His research interests include computer networks, computer architecture, robotics, and computer security.

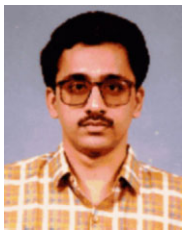
Min Chen is an assistant professor in School of Computer Science and Engineering at Seoul National University (SNU). He received the Ph.D in Electrical Engineering from South China University of Technology in 2004, when he was 23 years old. He has been a Research Associate in the Department of Computer Science at University of British Columbia (UBC) for half year, and worked as a Post-Doctoral Fellow in Department of Electrical and Computer Engineering at UBC for three years, where his faculty advisor is Prof. Victor C.M. Leung. Before joining UBC, he was a Post-Doctoral Fellow at SNU for one and half years. He received the Best Paper Runner-up Award from QShine 2008. He was interviewed by Chinese Canadian Times where he appeared on the celebrity column in 2007. He is the author of OPNET Network Simulation (Tsinghua University Press, 2004). He has served as session chairs for several conferences, including VTC-08, QShine-08, ICACT-09, Trientcom-09, and ICC-09. He serves as TPC co-chair and web chair for BodyNets-2010, workshop co-chair for CHINACOM 2010. He is the co-chair of MMASN-09 and UBSN-10. He was the TPC chair of ASIT-09, TPC co-chair of PCSI-09 and PCSI-10, publicity co-chair of PICom-09. He is a member in ICST Information Technology Committee. He is the corresponding guest editor for IJSNet Special Issue on 'Recent Advances in Sensor Integration'. He is the corresponding guest editor for IJCNDS Special Issue on 'Mobile, Multimedia, Ad Hoc & Sensor Networks'. He is the guest editor for ACM WINET SI on 'Ubiquitous Body Sensor Networks', and IJCS SI on 'Advances on Multimedia Communications'. Currently, his research focuses on multimedia and communications, such as multimedia transmission over wireless network, wireless sensor networks, body sensor networks, RFID, ubiquitous networking, intelligent mobile agent, pervasive computing and networks, etc.

Shiwen Mao received the Ph.D. in Electrical and Computer Engineering from Polytechnic University (now Polytechnic Institute of New York University), Brooklyn, NY in 2004. Currently, he is an Assistant Professor in the Department of Electrical and Computer Engineering at Auburn University, Auburn, AL. His research interests include algorithmic, optimization, and performance issues in wireless networks and multimedia communications. He is a co-recipient of the 2004 IEEE Communications Society Leonard G. Abraham Prize in the Field of Communications Systems and the Best Paper Runner-up Award of QShine 2008.

Yang Xiao worked in industry as a MAC (Medium Access Control) architect involving the IEEE 802.11 standard enhancement work before he joined Department of Computer Science at The University of Memphis in 2002. He is currently with Department of Computer Science at The University of Alabama. He was a voting member of IEEE 802.11 Working Group from 2001 to 2004. He is an IEEE Senior Member. He is a member of American Telemedicine Association. He currently serves as Editor-in-Chief for *International Journal of Security and Networks (IJSN)*, *International Journal of Sensor Networks (IJSNet)*, and *International Journal of Telemedicine and Applications (IJTA)*. He serves as a referee/reviewer for many funding agencies, as well as a panelist for NSF and a member of Canada Foundation for Innovation (CFI)'s Telecommunications expert committee. He serves on TPC for more than 100 conferences such as INFOCOM, ICDCS, MOBIHOC, ICC, GLOBECOM, WCNC, etc. He serves as an associate editor for several journals, e.g., *IEEE Transactions on Vehicular Technology*. His research areas are security, telemedicine, sensor networks, and wireless networks. He has published more than 300 papers in major journals, refereed conference proceedings, book chapters related to these research

areas. Dr. Xiao's research has been supported by the US National Science Foundation (NSF), U.S. Army Research, Fleet & Industrial Supply Center San Diego (FISCSD), and The University of Alabama's Research Grants Committee.

Walter Read began teaching at CSU Fresno in the Department of Mathematics in 1969. He was one of the founding faculty of the Department of Computer Science and retired in summer 2009.



B. Prabhakaran is an Associate Professor with the faculty of Computer Science Department, University of Texas at Dallas. He has been working in the area of multimedia systems: animation & multimedia databases, authoring & presentation, resource management, and scalable web-based multimedia presentation servers. He received the prestigious National Science Foundation (NSF) CAREER Award in 2003 for his proposal on Animation Databases. He is also the Principal Investigator for US Army Research Office (ARO) grant on 3D data storage, retrieval, and delivery. He has published several research papers in various refereed conferences and journals in this area.

He has served as an Associate Chair of the ACM Multimedia Conferences in 2006 (Santa Barbara), 2003 (Berekeley, CA), 2000 (Los Angeles, CA) and in 1999 (Orlando, FL) He has served as guest-editor (special issue on Multimedia Authoring and Presentation) for ACM Multimedia Systems journal. He is also serving on the editorial board of journals such as *Multimedia Tools and Applications* (Springer), *Journal of Multimedia* (Academy Publishers), and *Journal of Multimedia Data Engineering and Management* (Information Resources Management Association (IRMA)). He is also the Editor-in-chief of the ACM SIGMM (Special Interest Group on Multimedia) Online magazine. He has also served as program committee member on several multimedia conferences and workshops. He has presented tutorials in ACM Multimedia and other multimedia conferences.

He has served as a visiting research faculty with the Department of Computer Science, University of Maryland, College Park. He also served as a faculty in the Department of Computer Science, National University of Singapore as well as in the Indian Institute of Technology, Madras, India.