Mobile Object Tracking in Wireless Sensor Networks*

Hua-Wen Tsai[1] Chih-Ping Chu[1] Tzung-Shi Chen[2 #]

[1] Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan 701, Taiwan.

[2] Department of Information and Learning Technology
National University of Tainan
Tainan 700, Taiwan
e-mail: chents@mail.nutn.edu.tw

Abstract— **Wireless sensor network is an emerging technology that enables remote monitoring objects and environment. This paper proposes a protocol to track a mobile object in a sensor network dynamically. The previous researches almost focus on how to track object accurately and they do not consider the query for mobile sources. Additionally, they need not report the tracking information to user. The work is concentrated on mobile user how to query target tracks and obtain the target position effectively. The mobile user can obtain the tracking object position without broadcast query. The user is moving and approaching the target when he/she knows the target's position. Wireless sensor networks can assist user to detect target as well as keep the movement information of the target. Sensor nodes establish face structure to track the designated target and keep target tracks. The source follows the tracks to approaching target. To chase the object quick and maintain an accurate tracking route, the sensors cooperate together to shorten the route between target and source dynamically. A source can quickly approach a target along a shortened route. Finally, we compare the proposed scheme with three flooding-based query methods. By the simulation results, the proposed protocol has better performance than that of flooding-based query methods.**

Index Terms—**Mobile computing, object tracking, routing, spatiotemporal guarantee, wireless sensor networks.**

* This work is supported partially by National Science Council of the Republic of China, Taiwan, under Contract Nos. NSC-94-2213-E-024-002 and NSC-95-2221-E-024-012.
Corresponding author

I. INTRODUCTION

Recent advances in wireless communications and electronics have enabled the development of low-cost, low-power, multifunctional sensor nodes which are small in size and communicate un-tethered in short distances. These tiny sensor nodes have sensing, data processing, and communicating components capabilities. A wireless sensor network (WSN) is composed of a large number of sensor nodes and deployed either inside the phenomenon or very close to it. Wireless sensor networks are expected serve as a key infrastructure for a broad range of applications including precision agriculture, surveillance, intelligent highway systems, emergent disaster response and recovery. One of the important application issues for sensor networks is utilized to track mobile object. In such scenarios, the sensor networks may be deployed for military (tracking enemy vehicles, detecting illegal border crossings) and civilian purposes (tracking the movement of wild animals in wildlife protection). To track an object accurately, two or more of sensors are required to sense the object simultaneously. The cooperation is an important issue for object tracking. However, the activated sensors need to consume power because of communication, sensing, or other factors. We would like to select the fewest essential number of sensors dedicated for the task and at the same time other sensors stay in the sleep state. During the tracking, a large number of sensors are involved in cooperation. Such object tracking sensor network provides significant research opportunities in terms of energy management. To simultaneously satisfy the requirements of saving power and improving overall efficiency, large scale coordination and other management operations are needed.

In previous object tracking sensor networks, the sensors are assumed activating. But this assumption causes that sensors in WSN consume too much energy. This is because that a lot of sensors are assigned to detect the moving object and transmit control data at the same time. Hence, we can utilize Collaborative Signal and Information Processing (CSIP) to reduce the energy consumption. For the kind of techniques, CSIP has been proposed in [3][12][24][31]. In general, the object tracking protocols are classified into *cluster-based* and *non-cluster-based* protocols in WSNs. In cluster-based protocols [7][8][25][30], a non-cluster sensor node detected an object and then it forwards an information to its cluster head. Next, the cluster head collects and propagates the information to a sink. This approach reduces the required communication bandwidth and energy consumption. Therefore, WSNs can prolong lifetime. In non-cluster-based protocols, there is not any node to serve as cluster head in WSNs. When a sensor detects an object, it records the object information in its local memory. A user issues a request to WSNs when he/she wants to know the location of tracked object. If a sensor has the information of the tracked object, it replies the information to the user. Kung *et al.* [21] and Lin *et al.* [23] assume that a logical structure of connecting sensors exists in WSNs. They build a hierarchical structure that allows the system to handle a large number of tracked objects. In addition, Tseng *et al.* [26] proposed a novel protocol based on the mobile agent. Once a new object is detected, a mobile agent will be initiated to track the roaming path of object. The mobile agent will choose and stay in a sensor that is the closest to the tracked object. The agent invites some nearby slave sensors to track the position of object cooperatively and inhibits other irrelevant sensors to track object. Both the overhead of communication and the sensing energy are reduced.

For saving energy, the prediction-based methods [13][29][30] are used to predict the location of mobile object. When a sensor detects an object, it forwards the object information to its cluster head. The information contains the location, velocity and moving direction of object. The cluster head calculates and predicts the location of object and then it multicasts wakeup information to the predicted area (forwarding area). The multicast method is called *mobicast*. These sensors in the forwarding area wake to perform sensing task and wait for source arrived. However, the object tracking may be failed when mobicast method meet a hollow region in sensor network. The "hole" problem is one of key issues in object tracking for WSNs.

To solve holes problem for mobicast, some protocols are proposed object tracking protocols that possess temporal delivery guarantee. Some literatures [5][6][15][16][20][22] have been addressed for spatial and temporal delivery guarantee. Chen *et al.* [5][6] builds a new shape of a forwarding zone, called the variant-egg. They utilize the variant-egg shape of the forwarding zone to achieve a high predicted accuracy by considering the factors of moving speed and direction. Huang *et al.* [16] presented a new face-aware mobicast routing protocol. This protocol relies on the notion of spatial neighborhoods and features a novel timed face-aware forwarding method. This protocol uses face routing to achieve high space delivery guarantee and uses timed forwarding for controlling information propagation speed.

The mobicast routing protocols for sensor networks are main designed for predicting the object moving direction. Before the object arrived, some nodes are waked up to prepare for detecting object. They do not consider how to inform mobile user the present location of target. This work proposes a novel object tracking protocol in sensor networks for mobile user. The purpose of this proposed protocol is different than that of other mobicast routing protocols. This protocol guides a mobile user to chase a mobile object and it does not need flooding request to obtain the present location of object. This protocol can track mobile object accurately and save power consumption to prolong wireless sensor network lifetime. A mobile user is called source and a tracked object is called target. The source wants to chase the target. The sensor network assist source in detecting the target and keeping the target's track information. The sensor node keeping the track information acts as a beacon that waits for source and guides source to chase the target. The source follows the track to approach the target. To save power consumption, some sensor nodes are in active state to track target and others are into sleep state. In the course of chasing, source does not need to request the present location of target frequently. The sensor also does not need to tell the source the target location when sensor detects the target. When the source reached the location of beacon sensor, it queries the sensor the next moving position. The next position is the present location of target or the location of next beacon sensor. Source will catch the target along the sequence of beacon sensors. Due to the target moves arbitrarily, the track route does not form a straight line. For accurately tracking the target, this work utilizes face routing component to achieve spatiotemporal guarantee and solve the holes problem. Furthermore, the moving direction and velocity of target are also considered. To abridge the catch time, the sensors can cooperate to adjust the route between the target and the source dynamically. The source would reach the site of target along the adjusted route faster than along the target track. By the experimental results, this proposed protocol can save more energy than other flooding based protocols in object tracking. Therefore, this protocol can extend the lifetime of the entire wireless sensor network. Additionally, the protocol also guides a source to catch a target fast.

The rest of this paper is organized as follows. Section 2 presents the object tracking protocol for wireless sensor networks. We compare the proposed protocol with three flooding-based protocols in Section 3. The conclusions from this work are presented in Section 4.

II. Mobile Object Tracking

This section introduces the proposed protocol in details. The work is to focus on mobile user how to query target tracks and obtain the target position effectively. This work designs an efficient object track protocol that can decrease energy consumption and increase tracking efficacy. Assume the mobile user wants to follow a mobile object in a sensor network. First, the overview and definitions of protocol are presented. Next, the object tracking processes are introduced including *target discovery, target detecting, target tracking, face-track shortening,* and *loop face-track removing.*

A. Assumptions and Definitions

First, several assumptions of the proposed protocol are defined. A mobile user, called source, wants to track a mobile object, called target, in a wireless sensor network. Assume that a source knows the information of target. The target information includes the characterization which is used to identify and to seek the target. When a sensor network identifies the target in accordance with the target/object information, it assigns a unique number for the target. This identification information is exchanged between sensors. A sensor node has three states, *active*, *sleep* and *awaking*. While a sensor is in *active* state, it can sense object, receive or transmit data any time. The sensor continues to work in *active* state until an active time expires. While a sensor is in *sleep* state, the sensor stops sensing, receiving or transmitting. A sensor periodically wakes at a predefined period and changes its state to *awaking*. In *awaking* state, the sensor listens to communication channel to check *request* or *wakeup* packets. If it receives a target discovery request packet, it delivers this packet to sensor network with flooding. If it receives a *wakeup* packet in this period, its state will be changed as *active*. If the sensor does not change its state to *active* in this period, its state returns to *sleep*. The sensor network is assumed synchronization. The sensors periodically synchronize awaking and sleeping. Additionally, a sensor can detect the accurate location of object, because the sensor utilizes trilateration to compute the object's location. The trilateration has been proposed in [2]. Each node knows its location and this information can be acquired from global positioning system (GPS) or other mechanisms. Notation $L(o)$ denotes the location of object o. The communication and sense range for each node are both same. This work assumes that the transmission range is two times of sensing range. In this range ratio, the sensors can track an object cooperatively. When an object leaves the sensing range of sensor X and moves into the sensing range of sensor Y, this range ratio can guarantee that X and Y are neighbors. Sensor Y can directly inform X the object information so sensor network can track object cooperatively.

Next, some notations are given below. To track mobile object and solve the "hole/obstacle" problem, this work utilizes face routing component [1][14][16] to construct a spatial neighborhood for preventing losing the track of object. This work uses the knowledge of spatial neighborhood defined on a planar graph. To let each node find out locally who its spatial neighbors are, we first need a method to planarize the network. It is well known that the Gabriel Graph (GG) and the Relative Neighborhood Graph (RNG) [1][28] are planar graphs. In a geometric graph, an edge $e = (u, v)$ is called a "Gabriel edge" if there is no other node inside the disk which uses e as a diameter. An example is in Fig. 1. A graph is a GG if it contains only Gabriel edges. Gabriel subgraphs of non-planar have been used in [4][19] for unicast geometric routing. A simple distributed algorithm can be found in both papers. We use unit disk graph as an approximation for sensor networks in our simulation. In a unit disk graph, two nodes have a common edge if and only if their Euclidean distance is less than a constant.

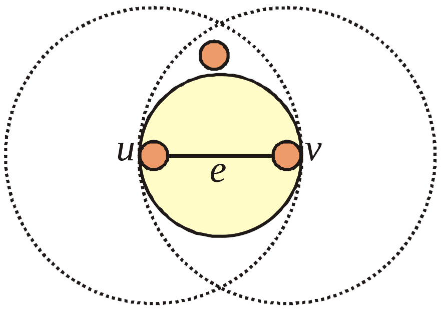


Fig. 1: A Gabriel edge.

Each node exchanges its location information and computes the face neighbors. If a neighbor v is a face neighbor of node u, the e is a Gabriel edge. When a node has n face neighbors, it means that the node has n adjacent faces. Next, it issues a packet to collect and construct adjacent faces. A node does not need to know all of sensors and their location. It only needs to get the information of nodes that are in its adjacent faces. F_i is denoted as an identity of face i in the network. In Fig. 2, node n_1 has four face neighbors n_2, n_3, n_4, and n_5, so it has four adjacent faces F_1, F_2, F_3, and F_4. In same situation, node n_3 has three adjacent faces F_1, F_3, and F_5. Note that the "boundary node" n_{10} has two adjacent faces F_4 and F_7. One of them is the "inner" face F_4 formed by node n_{10}, n_9, n_8, n_4, n_1, and n_5, the other is the "outer" face F_7 formed by node n_{10}, n_5, n_2, n_{16}, n_{15}, n_{14}, n_{13}, n_{12}, n_{11}, n_6, , n_7, n_8, and n_9. These neighbors in F_i are called spatial neighbors. Let $F_i.SN$ is a set of n's spatial neighbors in F_i, and $F_i.sn_1$, $F_i.sn_2$, …, $F_i.sn_j$, be all neighboring face nodes of F_i of n ($F_i.sn_k \in F_i.SN$, $1 \le k \le j$). The identification methods of the "boundary", "inner" and "outer" face are proposed in [16]. The face structure is established in face discovery phase.

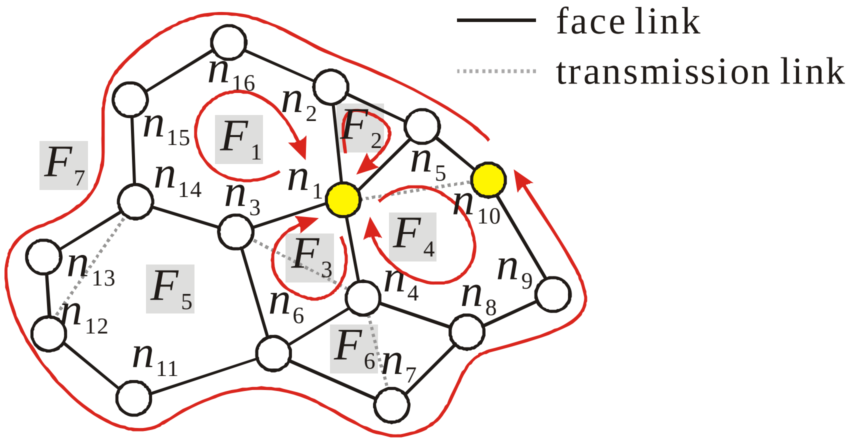


Fig. 2: Planar graph and planar neighborhood.

B. *Overview*

This subsection illustrates the overview of proposed object tracking protocol in Fig. 3. This protocol is applied to assist a source S in chasing a mobile object o. A scenario is that we dispatch a tiny robot (like a bee) to chase an enemy or a wild animal. The robot must follow the target. Assume that the robot can not detect object location, so the robot must employ sensor networks to get target location. When S gets the target location, it chases the target. The moving target maybe has left the original place while S reached the obtained location. But S cannot predict the moving direction of target. If S uses continually a flooding method to obtain the target position, the energy of sensors will be exhausted soon. This work proposes a protocol that not only saves sensor energy but also chases target accurately and fast.

First, S uses a flooding request to get the target location and asks sensor network to track target. Source S obtains the target's present location is near sensor n_1. Next, S moves to the target's location and queries sensor n_1 the target's present location. Because the target moves arbitrary, the sensors need to record the target tracks. This work utilizes face structure to distinguish different areas in sensor network. A sensor n detects a target enters its adjacent face and n is the closest to the target. This kind of sensor is *ingress node*. While the tracked target moves through k faces, k ingress nodes exists in sensor network. The sequence of ingress nodes represents the target tracks. E.g. the target tracks are recorded as <n_1, n_2, n_3, n_4, n_5, n_6 >. When source S reached n_1, it queries n_1 the next position n_2. The ingress node acts a mark to guide source the chase direction, so ingress node also is called *beacon node*. Source S will chase the target along the sequence of beacon nodes. This is similar to that ants (i.e. sensor network) release a substance called pheromone (i.e. beacon nodes) to communicate with other ants (i.e. source). An ant (i.e. S) will follow the scent and reach to a food source (i.e. target).

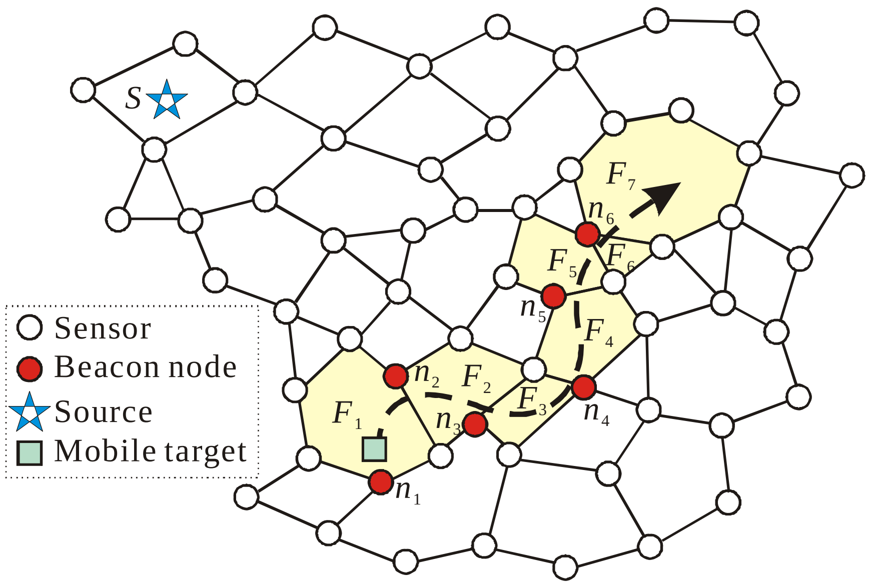


Fig. 3: An overview for object tracking.

C. *Face Discovery*

First, each node collects the neighbor information by hello message. Next, each node computes its *face neighbor nodes* by Gabriel Graph (GG). The face neighbor node means the link between this node and its neighbor conforms to GG. Next, each node informs the computed face neighbor nodes to its neighbors. Each node decides its face neighbor nodes after these processes. Next, each node utilizes the right-hand neighborhood discovery protocol that is proposed in [14][16] to construct adjacent faces. The face maintenance also follows [14][16]. After the face discovery, each node obtains the location information of spatial neighbors. This face discovery phase is performed in initial.

D. *Target Discovery*

This subsection discusses target discovery process. A source *S* wants to track a target *o* but it does not know the location of target *o*. Source *S* employs the sensor network to discover the target *o*. Source *S* issues a flooding request packet to seek target. The sensor network is synchronization and the sensors periodically synchronize to wake and to sleep. Source *S* issues a request packet in *awaking* period. The format of request packet is ***request***(*packet type*, *source id*, *sequence number*, *target information*), where the packet type is *Request*, the sequence number is used to avoid forwarding the duplicate packet, the target information is used to identify target.

When a sensor n_i receives a request packet, it judges whether it is a *near-node*. Near-node means that the node is the closest to the target than others. If a sensor n_i detects the target *o* that stays in its sense range and it is the near-node, it forwards a reply packet to source *S*. This process is similar to the route discovery process in ad hoc networks. Because every node has collected its all spatial neighbors' location in face discovery phase, it can judge whether it is the closest to the target. The format of *reply* packet is ***reply***(*packet type*, *source id*, *target id*, *the location of target*, *beacon node id*), where the packet type is *REPLY*, target id is an unique number of target, beacon node id is n_i. When the sensor n_i issues a reply to the source, it set its state as *active* and the active time as infinity. It becomes a beacon node and begins to track the target.

If the source does not receive reply at a predefined time, it rebroadcasts request packet at next *awaking* period. This mechanism can prevent the routing failure, node failure and lose tracking. Additionally, the advantage of face structure is that it is suitable for any kinds of node-density networks for being able to solve the hole/obstacle problems. If the target unfortunately stays in a hollow region while the source requires target information, the source cannot get the information and then it will try to rediscover the target after a predefined period. When the target left in the hollow region, the source can obtain the target location at rediscovery process.

E. *Target Detection*

If a sensor *n* is a near-node and target *o* is in face F_a, this sensor *n* becomes an ingress node of F_a and tracks the target. If the sensors do not cooperate to detect the target, it maybe loses the target tracks while the target moves. For tracking the target accurately, sensor *n* asks its spatial neighbors for cooperating tracking target. Sensor *n* issues *wakeup* packet to wake its all spatial neighbors of adjacent faces and asks them for detecting target *o* cooperatively. The *wakeup* packet format is ***wakeup***(*packet type*, *ingress id*, *near-node id*, *object in face id*, *received node id*, *target id*, *the location of target*, *active time*, *face hop count*), where the packet type is *WAKEUP*, received node id is indicates the node to receive this packet and -1 means this packet is broadcast, active time indicates the active period of spatial neighbors, the face hop count indicates the number of faces that the target already has moved. The face hop count is set as 1 initially. The wakeup packet is forward along the face routing.

When the spatial neighbors of *n* receive *wakeup* packet, they stay in *active* state and detect target cooperatively. They do not enter *sleep* state at the predefined *sleep* period. The ingress node forwards *wakeup* packet periodically while it is a near-node. If the target leaf the sensing range of the ingress node and it is still in the same face F_a, another sensor *m* ($m \in F_a.SN$) can detect the target and it is the near-node now. Because every node has the location information of its spatial neighbors, it can check whether it is the closest to the target. Sensor *n* stops issuing *wakeup* packet, but sensor *m* will issue *wakeup* packet to its all spatial neighbors. In other words, every active node detects the target and must to check whether it is the near-node. A sensor, which is the near-node, has to issue *wakeup* packet, but the others have not. The target information is stored in all of spatial neighbors until active time expires. When the active time has expired, the sensor enters *sleep* state. Hence, the *wakeup* packet can wake the *sleep* sensors and refresh the active time of *active* node. We assume that the active time of ingress node (i.e. beacon node) is infinity. This is because this kind of node must to guide source *S* the chasing direction of target *o*. Therefore, this node has to stay in *active* state until source *S* arrives. The near-node and its spatial neighbors form a detecting region, called *envelopment-net*, i.e. a group of sensors besieges the target. While the target is moving, the *envelopment-net* also follows the target. Therefore, this proposed protocol can track target accurately and avoid losing the target traces. This *envelopment-net* utilizes the advantage of face structure to solve holes and obstacle problem. Additionally, *envelopment-net* can also guarantee that the target is in the *envelopment-net* region due to the characteristics of face structure. If we can ascertain that a target has moved into a face which includes a hollow region but loses the target tracks in the hollow region in the face, we can guarantee that the target is still in this face.

An example for detecting the mobile target is illustrated in Fig. 4. Assume a mobile target is in face F_1 at period *T*. Three sensors n_1, n_2 and n_3 can detect the target and n_1 is the near-node. When n_1 receives a *request* packet from a source, it replies a *reply* packet to the source and it becomes an ingress node. Next, n_1 issues *wakeup* packets to wake its all spatial neighbors. In Fig. 4, n_1 broadcasts *wakeup* packet (*WAKEUP*, n_1, n_1, -1, F_1, *o*, $L(o)$, *t*, 1) to its neighbors n_2, n_6 and n_9. When sensor n_2 receives *wakeup* packet from n_1, it means this packet is in F_1. Sensor n_2 forwards this packet to n_3. All spatial neighbors will receive this packet. These spatial neighbors include n_2, n_3, n_4, n_5, n_6 in F_1, n_9, n_2 in F_2 and n_6, n_7, n_8, n_9 in F_3. The target may move to adjacent face at next period (*T*+1), e.g. the target *o* may move to the adjacent face F_2, F_3, F_4, F_5, F_6 or F_7 in Fig. 4 (a).

If the target moves to adjacent face F_4 at *T*+2, it has to cross an edge (n_5, n_6). While the target moves to n_5's neighborhood at *T*+1 and it still stays in face F_1, n_5 broadcasts *wakeup* packet (*WAKEUP*, n_1, n_5, -1,

F_1, o, $L(o)$, t, 1) to face F_1, F_4 and F_5. If an *active* sensor does not receive a *wakeup* packet again in time t, its state returns to *sleep*. When the active time of n_8 and n_9 are expired, the states of sensors n_8 and n_9 return to *sleep* shown in Fig. 4 (b). If the target enters F_4 at T+2 and n_5 is also the near-node as shown in Fig. 4 (b), n_5 broadcasts *wakeup* packet (*WAKEUP*, n_5, n_5, -1, F_4, $L(o)$, t, 2) to its all spatial neighbors to wake up them. These spatial neighbors include n_6, n_1, n_2, n_3, n_4 in F_1, n_{10}, n_{11}, n_{12}, n_7, n_6 in F_4 and n_4, n_{10} in F_5. New, n_5 is the ingress node of face F_4 and the face hop count is set as 2. Sensor n_1 records n_5 as the next beacon node and n_5 records n_1 as the previous beacon node. The face hop count is increased when the target moves into a new face. When the source has arrived n_1, n_1 guides the source to n_5. The source follows a sequence of beacon/ingress nodes (i.e. the target tracks) to chase mobile target.

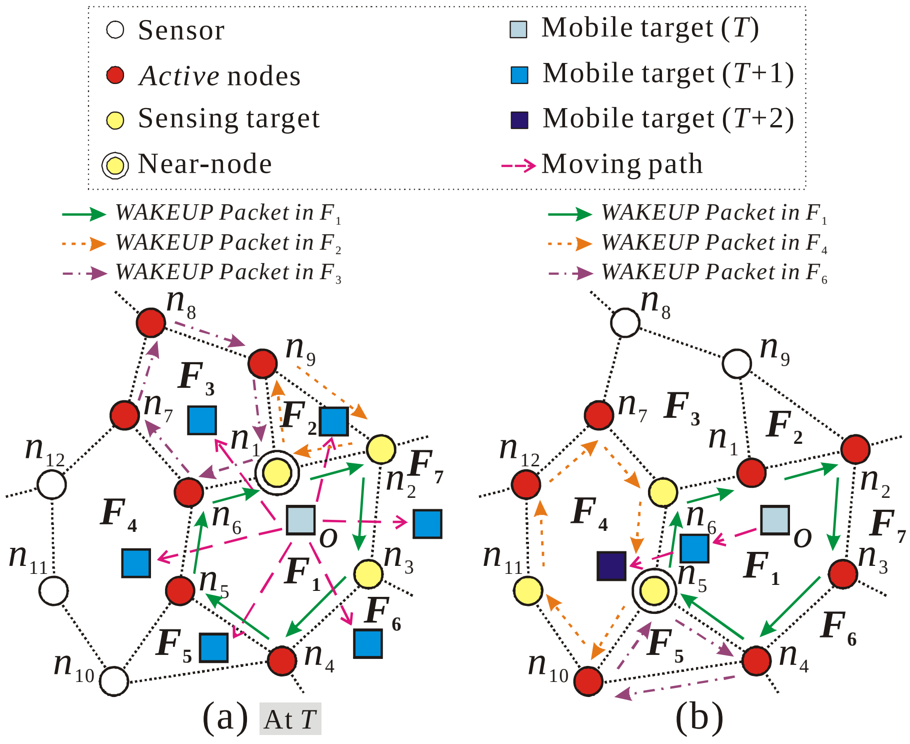


Fig. 4: An example for detecting mobile target.

To reduce power consumption, this work only needs the near-node and its spatial neighbors to detect the target. Other sensors turn into *sleep* state. Additionally, these nodes have to wake up periodically and check whether it needs to turn into *active* state and detect object. We define a parameter, the frequency of *awaking*. If the *awaking* frequency is high, it can avoid losing target tracks. The sensor network has to frequently wake for detecting target. The high frequency of *awaking* is suitable for the velocity of target be fast. If the frequency of *awaking* is low, it can save more of energy. The low frequency of *awaking* is suitable for the velocity of target be slow.

This work addresses the track losing problem here. If the frequency of *awaking* is low and the target moves very fast, the sensor may be losing the track of target. Additionally, if the covered area of face is very small, it also has the losing track problem. This is because that this work utilities an *envelopment-net* to besiege target. If the update rate of *envelopment-net* is too slow or the covered area of *envelopment-net* is too small, the target will escape from *envelopment-net*. The last ingress node stops searching target when it loses target tracks. When the source reached the last ingress node, the source rediscovers the target again. We show an example of losing track problem in Fig. 5. We assume that the target o is in F_1 at time T-1. The active nodes are n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8 and n_9. At time T, the target o moves close to n_{11}. Sensor n_{11} is sleeping in this moment, so n_5 still is the near-node. If n_5 wakes its spatial neighbors in F_1, F_4 and F_5 at time T, it does not have the losing track problem. Assume time T is *sleep* period and time T+1 is *awaking* period. Sensor n_5 will wake the spatial neighbors in F_4 at time T+1. But the target has got away the sensing range of n_{11} at time T+1. In this situation, the last ingress node n_5 loses the target tracks. Although n_5 loses the target track, the target discovery process is restarted when the source reached n_5. If the awaking frequency is set too low, the proposed algorithm is degraded and the target tracking process is the same as that of *Threshold Flooding*. *Threshold Flooding* only has the target discovery process of our algorithm. This method has to perform the target discovery process repeatedly when the source reached the obtained location.

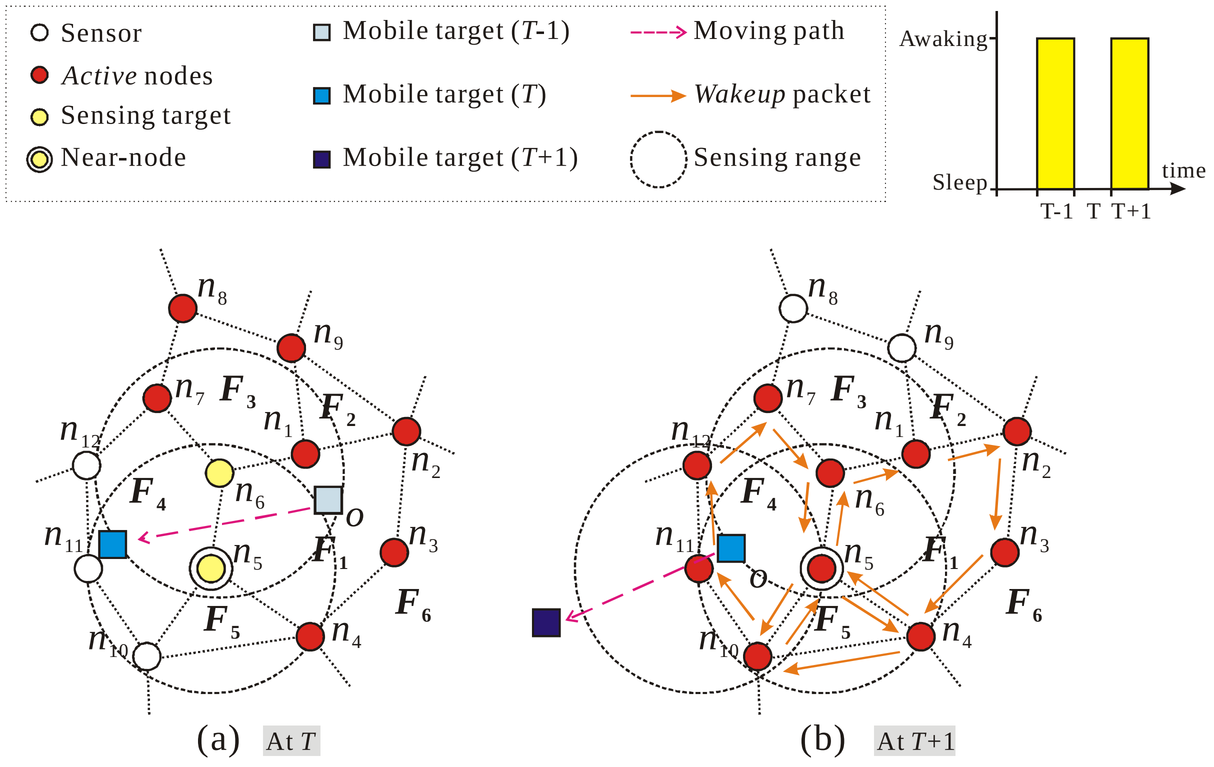


Fig. 5: An example for losing track problem.

F. *Tracking Target*

While a target *o* is tracked by source *S*, the sensor network has to record the target tracks. A source *S* obtains the target location that is informed from the first beacon node *n* (i.e. ingress node) after completing a target discovery process, and then *S* starts to move toward the first beacon node *n*'s location. When *S* reaches the position of the first beacon node *n*, *S* queries the beacon node for next position. The query packet format is ***q_next***(*packet type*, *source id*, *target id*, *received beacon node id*), where the packet type is *Q_NEXT*. The beacon node *n* informs *S* the target's or next beacon node's location information. The reply packet format is ***q_next_rep***(*packet type*, *source id*, *target id*, *next beacon node id*, *the location of target*), where the packet type is *Q_NEXT_REP*. If the target is still here, the source *S* moves to the target location and catches the target *o*. If the target has left, the beacon node *n* informs *S* next ingress node *m*'s location. Then, the source moves toward the next beacon node again. The beacon node *n* also informs the next beacon node *m* the information that the source *S* will reach *m*. The packet format is ***info_first***(*packet type*, *target id*, *sending beacon node id*, *next beacon node id*), where the packet type is *INFO_FIRST*. The beacon node *n* stops its *active* state and turns into *awaking* or *sleep* state. The active time is also set as 0. This node does not need to track target for source *S* anymore. The next beacon node *m* becomes the first beacon node. This process is repeated until the source catches target. The source will pursue the target along the sequence of beacon nodes. An example for tracking target is shown in Fig. 6. The source *S* follows the path of beacon nodes <n_1, n_2, n_3, n_4, …> to pursue the target *o*. This path is called *face-track*.

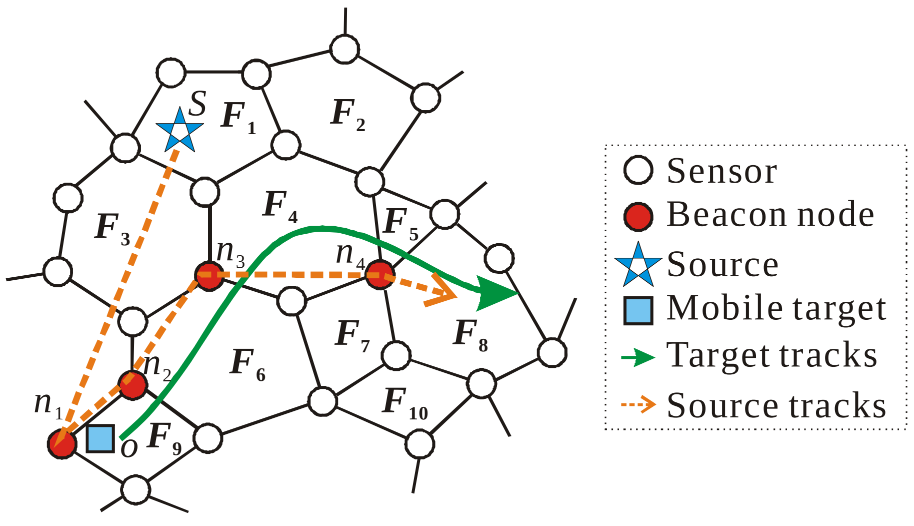


Fig. 6: An example for tracking mobile target.

G. *Face-Track Shortening*

This subsection discusses how to adjust *face-track*. When the velocity of target is faster than that of source, the source is very difficult to catch target. In the course of chasing, the *face-track* may be not the optimal tracks. Therefore, this work proposes a shortening method for *face-track* to make source chase target fast. The length of *face-track* will be shortened in this process. The *face-track* needs to adjust when the face hop count has already accumulated to K hops. When the target moves into the $n \times K$th face ($n \geq 1$), the (1+ $n \times K$)th ingress node needs to shorten *face-track* from the (1+ $n \times K$)th to the (1+ (n-1)$\times K$)th ingress node. The (1+$n \times K$)th ingress nodes are set as *checkface*. Only adjusting a part of *face-track* can reduce adjustment overhead. Additionally, the source is also approaching the target at the same time. So we do not need to shorten the *face-track* to the first ingress node.

An overview for adjusting *face-track* is shown in Fig. 7. Fig. 7 (a) shows that the target o has passed through faces F_9, F_6, F_4, F_5 and F_8. The *face-track* is <n_1, n_2, n_3, n_4>, where the 4th and 5th ingress nodes are both n_4. This *face-track* is not the optimal tracks. Here, we assume that the value of K is 4 and n_1 is the first ingress node (*checkface*). The source S is moving toward the first ingress node n_1 but S has not reached n_1. When the target has passed through 4 faces from the last *checkface*, the 5th ingress node (n_4) is the new *checkface* and it performs the *face-track shortening* process. Because the *face-track* is not a beeline, we shorten the moving path from n_1 to n_4. If the source S has reached the 1st ingress node n_1, S will directly move to n_4 shown in Fig. 7 (b). The adjusted source track is shorter than the original one.

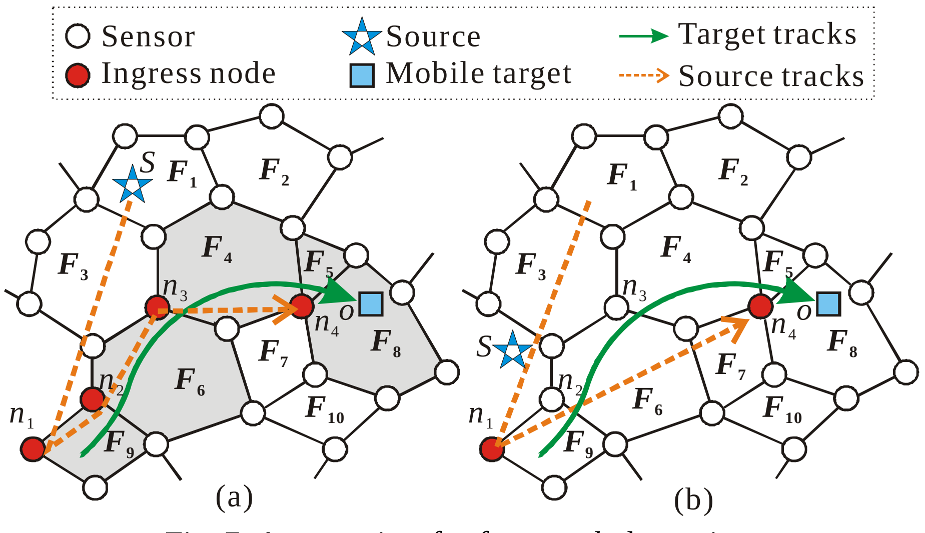


Fig. 7: An overview for *face-track* shortening.

When a target o enters to the $(1+n\times K)^{th}$ face, the $(1+n\times K)^{th}$ ingress node issues an *infoshort* packet to the $(1+(n\text{-}1)\times K)^{th}$ ingress node. The *infoshort* packet format is ***infoshort*** (*packet type, the last ingress node id, the last ingress node location, present face hop count, the previous ingress node*), where the packet type is *INFOSHORT*. When an ingress node that is between the $(1+(n\text{-}1)\times K)^{th}$ and the $(1+n\times K)^{th}$ ingress nodes receives this packet, it cancels the ingress node role. In other words, it is not an ingress node anymore. And then it forwards this packet to the previous ingress node. This process is repeated until the $(1+(n\text{-}1)\times K)^{th}$ ingress node received this packet. When the $(1+(n\text{-}1)\times K)^{th}$ ingress node receives this *infoshort* packet, it changes its next ingress node from $(1+(n\text{-}1)\times K+1)^{th}$ to $(1+n\times K)^{th}$. An example for *face-track* shortening is shown in Fig. 8. We assume that K is set as 4, n_1 is a *checkface* (i.e. the first ingress node) and n_4 is the $(1+K)^{th}$ ingress node of face F_8. In Fig. 8 (a), n_4 issues an *infoshort* packet to the previous *checkface* (i.e. n_1) by unicast. This packet is delivered along face routing. Ingress nodes n_2 and n_3 cancel ingress node role. When n_1 receives the *infoshort* packet, it changes its next ingress node from n_2 to n_4.

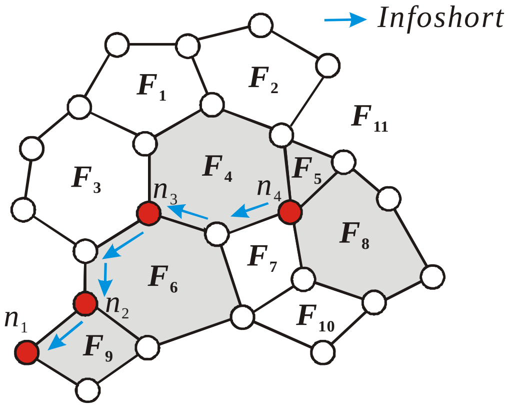


Fig. 8: An example for *face-track* shortening.

H. Loop Face-Track removing

This subsection discusses the loop *face-track* problem. When the *face-track* forms a loop, we have to remove track loop. An example for loop *face-track* removing is illustrated in Fig. 9. Assume that the object moving path is $<F_6, F_7, F_8, F_{10}>$ and the sequence of ingress nodes is $<n_1, n_2, n_3, n_4, n_5>$ shown in Fig. 9 (a). When the target o enters to face F_7, the new ingress node is n_5 and then n_5 issues a *wakeup* packet to its spatial nodes. When the old F_7's ingress node n_2 receives the *wakeup* packet, n_2 knows the *face-track* that has formed a loop. Therefore, n_2 delivers a *deletion* packet to n_5. Sensor n_5 cancels its ingress node state and it becomes a near-node of face F_7. Next, n_5 forwards the *deletion* packet to n_4. This process is repeated until n_2 received this packet. In other words, the *deletion* packet is forwarded backtracking to n_2. The ingress nodes in the loop are deleted. The *deletion* packet format is ***deletion***(*packet type, detecting loop ingress node id, the previous ingress node id*), where the packet type is *DELETION*. In Fig. 9 (b), the loop has deleted and new *face-track* is $<n_1, n_2>$.

III. Experimental Results

This section compares the proposed dynamic object tracking (DOT) protocol with the flooding-based object tracking protocols. The experiments are implemented in ns2 simulator [27]. The version of ns2 is 2.27. The simulations use CMU's wireless extensions [9] for the ns2 simulator. The nodes use the IEEE 802.11 radio and MAC model [17] provided by the CMU extensions. Additionally, we extend *NRL*'s Sensor Network [11] to ns2 simulator. This foundation consists of dual-homed sensor nodes that are data channel and phenomenon channel. The data channel is followed standard of 802.11. The project adds a phenomenon channel for detecting some physical phenomenon. The project facilitates to simulate sensor network environment. The value of noise or error models is default in ns2 simulator.

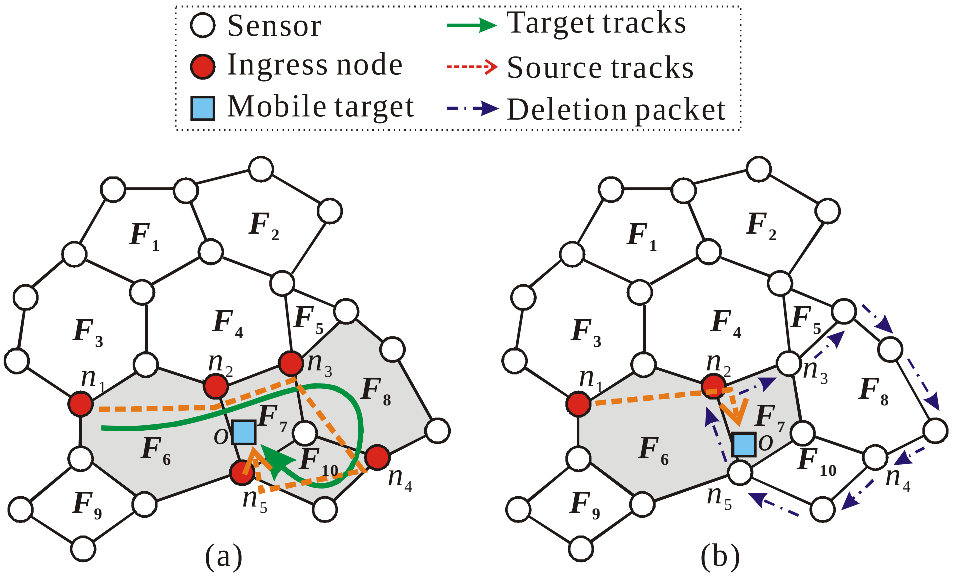


Fig. 9: An example for removing loop face-track.

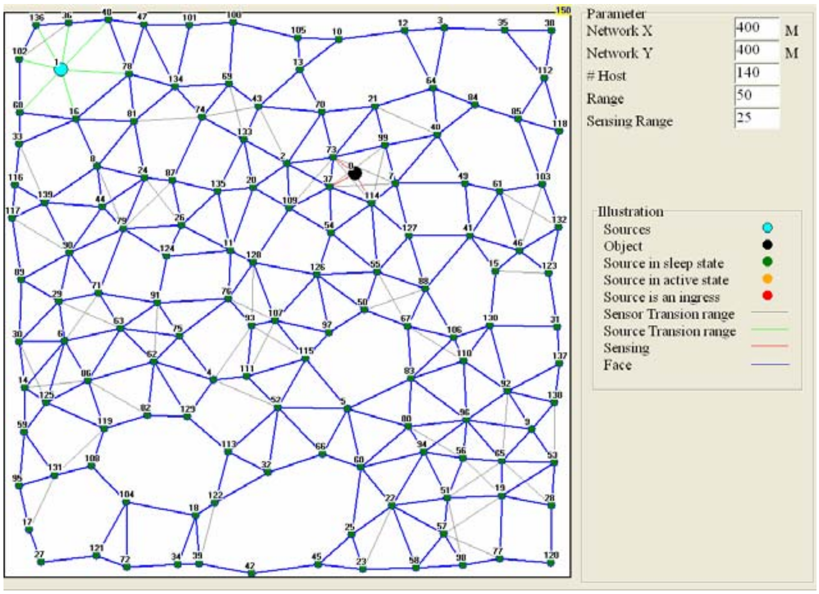
A. *Protocols Based on Flooding*

Due to the previous researches which are not considered in querying object tracks, this work compares the proposed protocol with three querying methods based on flooding. Three flooding query methods are *Threshold Flooding* (called TF), *Schedule Flooding* (called SF), and *Schedule Updating* (called SU). Our proposed protocol is called dynamical object tracking (called DOT for short).

In TF protocol, the target discovery is similar to our protocol, but this protocol does not maintain the target tracks. When the source gets the location of mobile target, it moves toward the obtained location. This method has to perform the target discovery process repeatedly when the source reached the location. In SF protocol, the source performs the target discovery process with a predefined period. The time interval is set as 2 second in the simulation. In SU protocol, the source broadcasts a query packet to all sensors at first. This query asks that the sensors that detect the object report the object location. After the first query, the source does not need to query the object tracks by flooding. The detecting object sensors update the object location information with a predefined period. The time interval is set as 2 second in the simulation. Because the sensor does not know the location of source, it updates the object's location by flooding.

B. *Simulation Model*

There are 140 sensors in this simulation network. These nodes are deployed uniformly and randomly in a 400m $\times$ 400m square region. In the simulation energy model, the initial battery is 30*J*, the transmission power is 700*mW* and the received power is 360*mW*. The communication range is 50 meter and the sensor range is 25 meter. The object node moves according to the "random waypoint" model [18] with velocity of 0~30 *meter/sec* and the pause time is 5 second. The velocity of source node is varied 5, 10, 20 and 30 *meter/sec*. Each run simulates 2000 seconds. During the 0~150 seconds, the sensor network performs face discovery phase. In the 150th second, the source performs target discovery phase. In a simulation run, the source chases the mobile target until the simulation time terminated. Three snapshots are shown in Fig. 10. We have evaluated four key performance metrics: (i) remaining energy and energy consumption (ii) maximum life time (iii) packet overhead (iv) average distance.


Parameter

Network X	400	M
Network Y	400	M
# Host	140	
Range	50	
Sensing Range	25	

Illustration

Sources	●
Object	●
Source in sleep state	●
Source in active state	●
Source is an ingress	●
Sensor Transion range	—
Source Transion range	—
Sensing	—
Face	—

(a) A snapshot in 150 sec

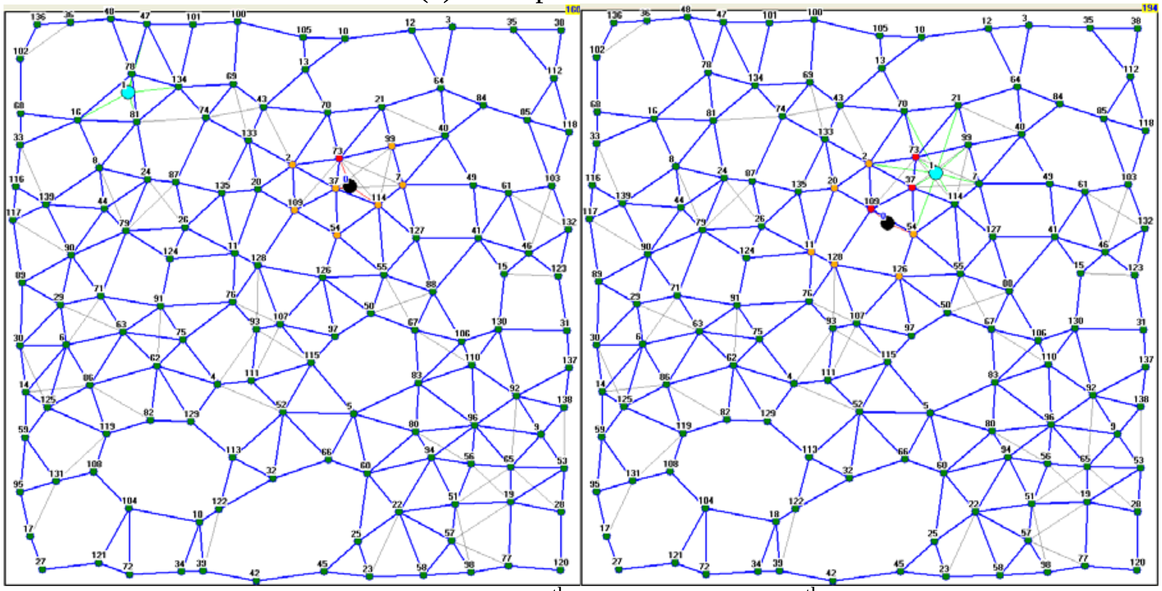

(b) The snapshot in the 160th sec (left) and 194 th sec (right)

Fig. 10: The snapshot for DOT simulation.

C. Simulation Results

First, we compare DOT with different *K* value. We present the performance of different *K* in simulation illustrations. Next, we compare DOT with three flooding-based query methods. The *K* value is varied 3~7. The notation DOT-k*i* means DOT with *K* = *i*. The *K* value affects the length of track path for source. When *K* value decreases, the sensor network adjusts *face-track* with increasing frequency. When *K* value increases, the sensor network also needs more of overhead for shortening the length of *face-track*. The source speed is varied 5, 10, 20 and 30 *meter/sec*. We simulate different scenarios with object speed. In slow moving scenario, the object speed is varied 0~15m/s. In medium moving scenario, the object speed is varied 10~30m/s. In fast moving scenario, the object speed is varied 10~30m/s. Fig. 15 shows the remaining energy of sensor with different object and source speed. In slow and medium moving scenarios, the remaining energy performance for different *K* does not have obvious difference. In fast moving scenario, DOT-k5 saves more energy than others when source speed is set as 10 and 20 m/s. Because the source has to follow the object and it cannot detect object, the source utilizes the sensor network to obtain the object location. When the source speed increases, the source can be very close to follow the object. To avoid losing the target tracks, the source has to query the beacon node about the target location periodically. The period is 2 seconds in the simulation. Therefore, the network has to consume more of energy when the source speed is fast. When the object speed increases, the *envelopment-net* also updates quick for tracking the target.

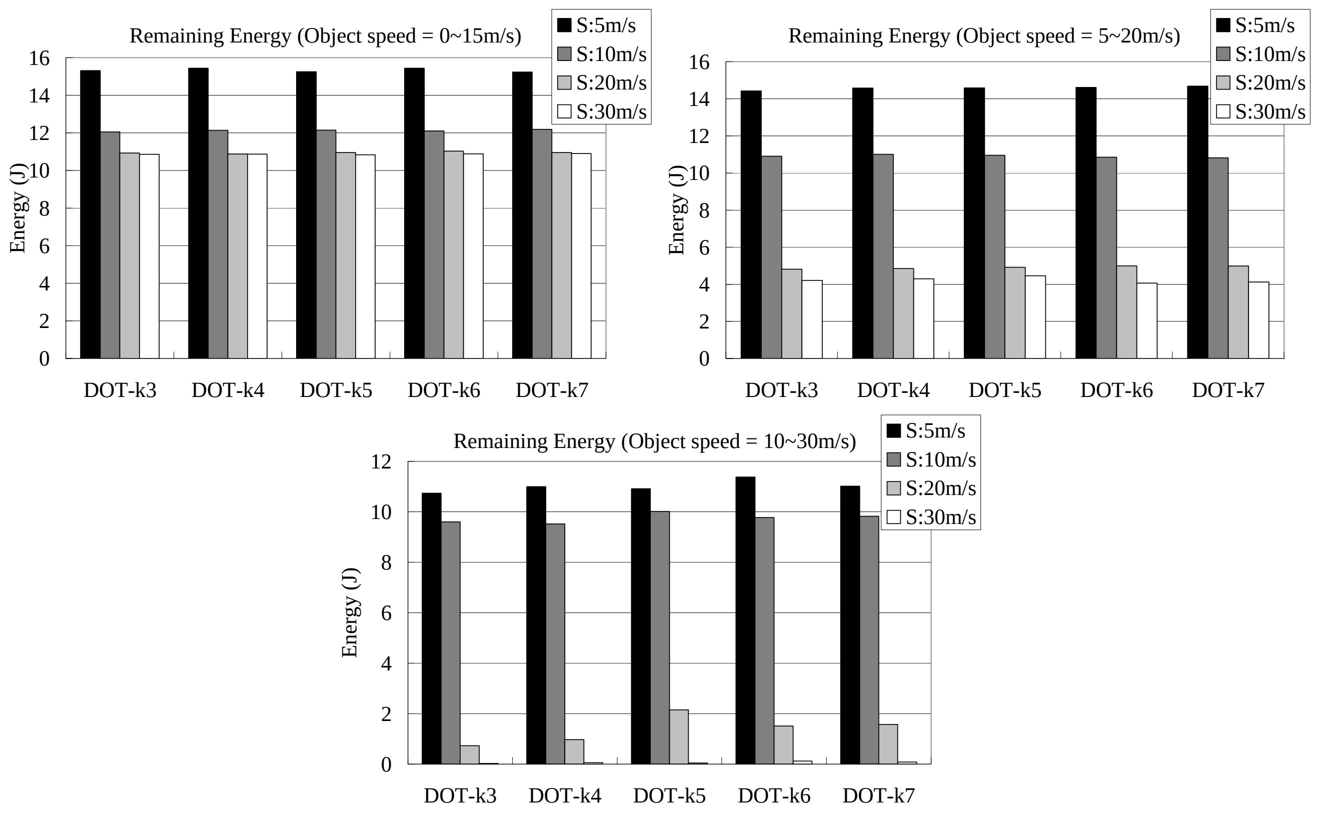


Fig. 11: The remaining energy of sensor for different object moving scenarios.

Because the experiment is simulated in same face structure, we compare the packet overhead for tracking object shown in Fig. 12. The performance of packet overhead for object tracking is similar to that of the remaining energy. In slow and medium moving scenarios, the packet overhead for different *K* does not have obvious difference. In fast moving scenario, DOT-k5 has less packet overhead than others when source speed is 20 m/s. When more of packets are issued in sensor network, sensor network has to consume more of energy. When the source speed increases, the *q_next* and *q_next_rep* packets are increased. When the object speed increases, the *wakeup* packet and face maintain packets are increased.

Fig. 13 shows the average relative distance between source and object. The average relative distances for different *K* do not have obvious difference. When the source speed is slow, the source is difficult to catch the target. Therefore, the average distance of S-5m/s is higher than that of S-30m/s. When the object speed increases, the average relative distance is increased. The source is not easy to catch the target in the course of chasing.

Fig. 14 shows the relative distance in a simulation that performs in slow moving scenario. When the object speed is very slow, the relative distance is very small. When the object speed increases, the relative distance is increased. We can observe this situation at the 414th second. During 476~551 seconds, the source chases the object along the *track-face*. We observe that DOT-k5 is moving toward the target faster than others. This result shows that the source is moving along short *track-face*. Summarized the above results, this work selects DOT-k5 to compare with other flooding-based query methods.

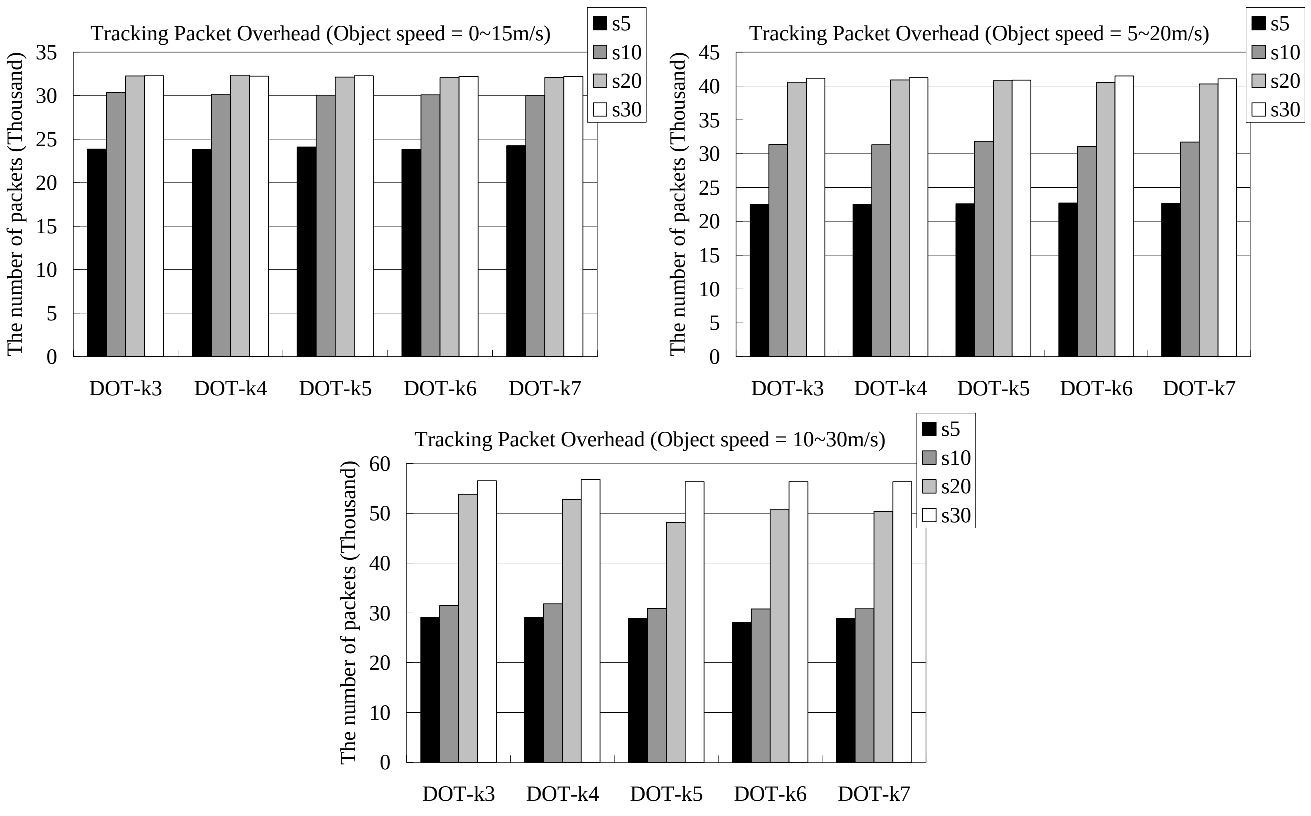


Fig. 12: The packet overhead for tracking object.

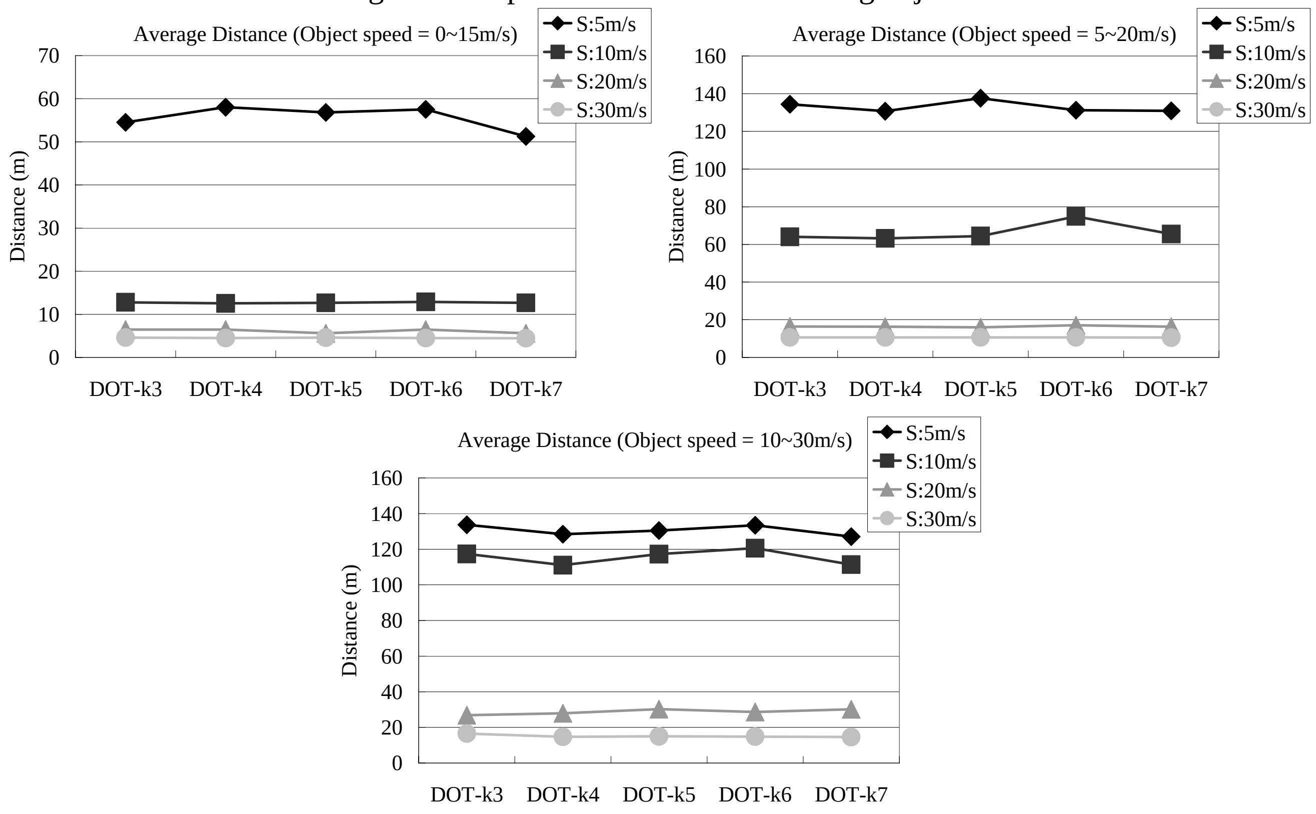


Fig. 13: The average distance between source and object.

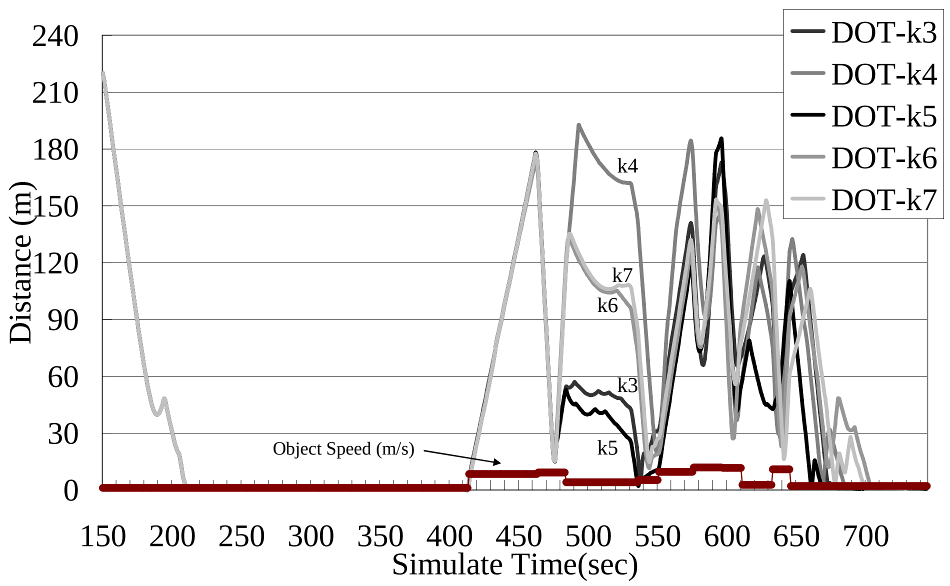


Fig. 14: the relative distances in a simulation.

Next, we compare DOT with three flooding-based methods. Fig. 15 shows the energy consumption for different source speed. This simulation performs in slow moving scenario. The results of other moving scenarios are similar to slow moving scenario. The SU and SF query methods periodically update object location by sensors and source, respectively. Therefore, the source speed does not affect the energy consumption. In SU method, several sensors that detect the object have to broadcast the object location. In SF method, only source needs to broadcast packet. Hence, SU consumes more energy than others. In TF method, the source has to query the object location when it reached the obtained location. When the source speed increases, the source catches the object fast. Then, the source has to obtain the object location every 2 seconds. Therefore, TF has to consume more of energy when the source speed increases. When source speed = 30m/s, the energy consumption of TF is similar to that of SF. The DOT does not increase a lot of energy consumption while the source speed increases. During 0~150 seconds, DOT needs to construct face routing, so DOT must to consume more energy than others. In the course of tracking and chasing, DOT can guide source by beacon nodes to chase object. The source does not query object location by frequent broadcast. Hence, DOT can decrease a lot of power consumption. Therefore, DOT can save more energy than others.

Next, we compare the maximum life time because of the sensor networks with flooding methods died in simulation run. The results of maximum life time are illustrated in Fig. 16. If the maximum life time = 2000, it means that the network does not die in simulation run. In fast moving scenario, SU method lets sensor network die at the 684th second. The SU method has shortest life time than others. The SF method also let sensor network die in the simulation run. The TF let sensor network die in the simulation run when the source speed increases. This is because that the source needs to obtain object location periodically. The TF method degrades the query efficiency as same as SF method. The DOT method saves packet overhead and power consumption, so it does not let sensor network die in the simulation run except in object fast moving scenario and the source speed = 30m/s.

Next, we compare total packet overhead that includes routing maintain, tracking object, query and reply packets. The results are shown in Fig. 17. The SU method has the highest packet overhead than others. The TF has more of packet overhead while the source speed increases. In high source speed, the packet overhead of TF is similar to SF. It means that the TF method degrades the query efficiency as same as SF method. The DOT has better performance than others. The DOT increases a little bit of packet overhead while the source or object speed increases.

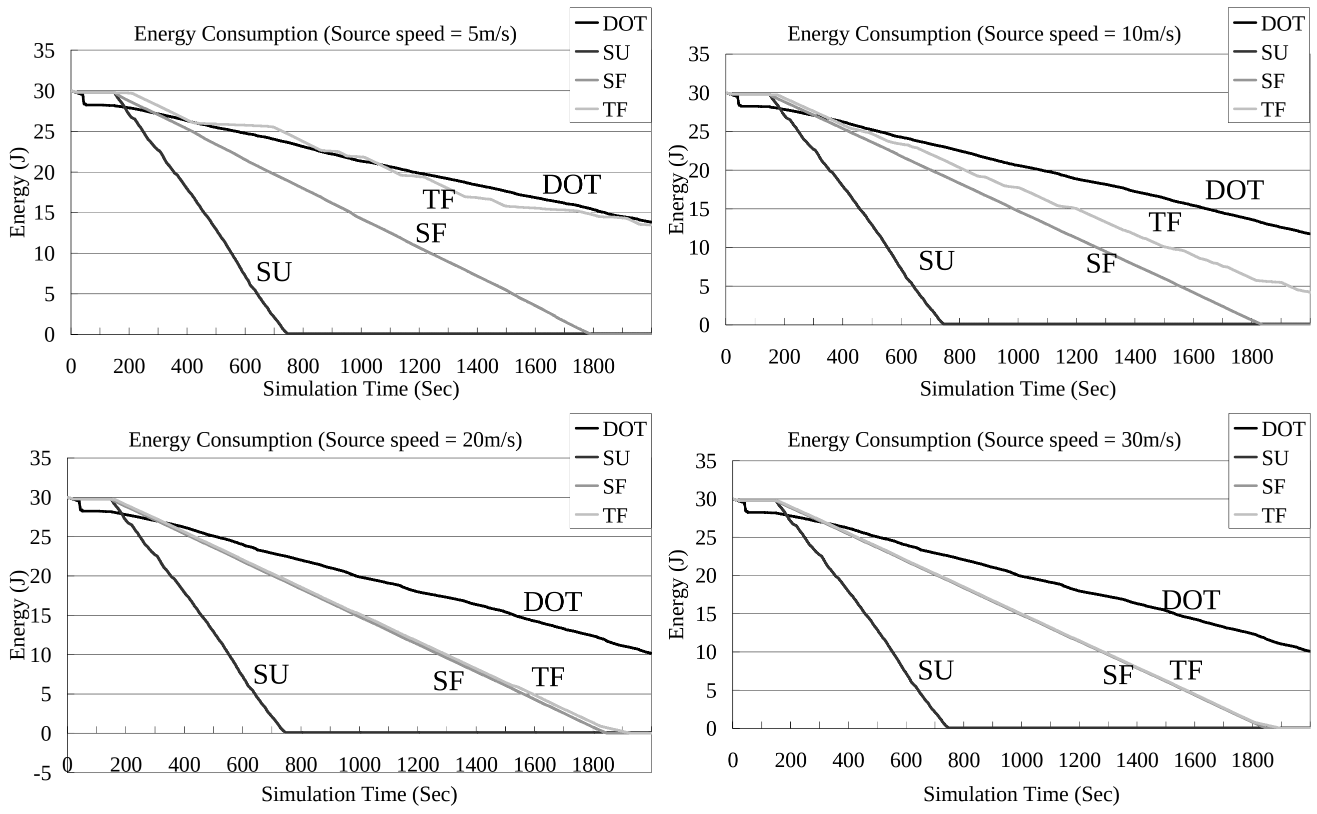


Fig. 15: Energy consumption with different source speed.

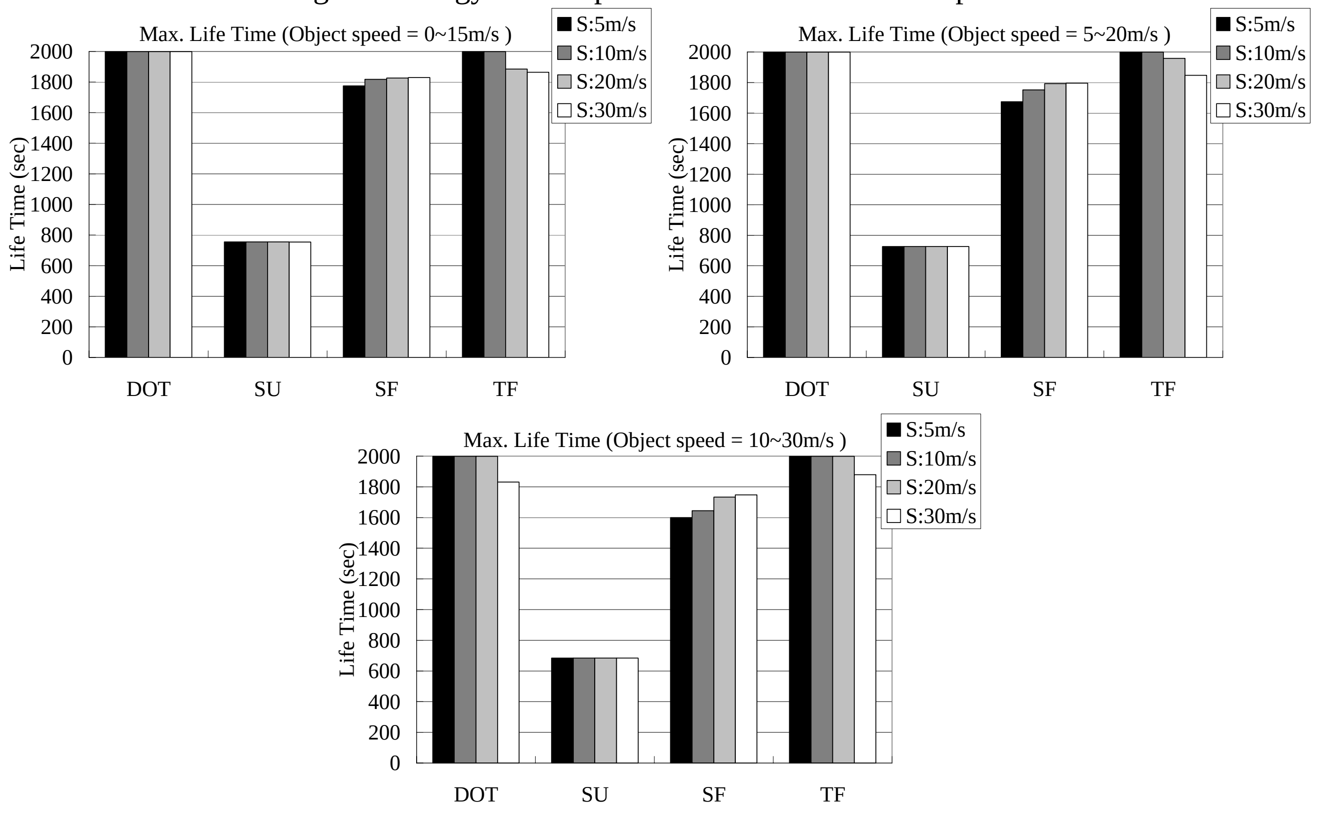


Fig. 16: The maximum life time for different object moving scenarios.

The simulation results of relative distances are illustrated in Fig. 18. The results of average distance are gotten from 0~670 seconds. The SU and SF have the same performance in relative distance because of the source chases the object along the shortest path. When the source receives the object location updating, it changes its moving direction immediately. The update ratio is 2 seconds. Therefore, they have better performance than DOT and TF in slow source speed. The DOT has better performance with high source speed than others. It is because that the source can obtain the object location information from the beacon node immediately. The source need not wait to obtain the present object position by querying. Therefore, the source with DOT can catch the target faster than that with others. Additionally, after the source met the object, the source can obtain the latest news from beacon node immediately when the object changes its moving direction. This also increases the tracking efficiency of DOT. The TF has longer distance than others. The source with TF queries the present object position when it reached the obtained location. Hence, the source cannot get the latest news of object position when the object changes its moving direction.

By the experimental results, DOT protocol has better performance of energy consumption, packet overhead and tracking than the flooding-based query methods. This protocol can reduce the power consumption when the source wants to query the target position. The source also need not issue periodical broadcast query to obtain the object position. It can reduce the packet overhead. While the source catches the target and it wants to follow the target, the proposed protocol can save the power efficiently. Moreover, a shortening method for *face-track* is proposed in this protocol. The source can follows short path to chase target and avoid loop path.

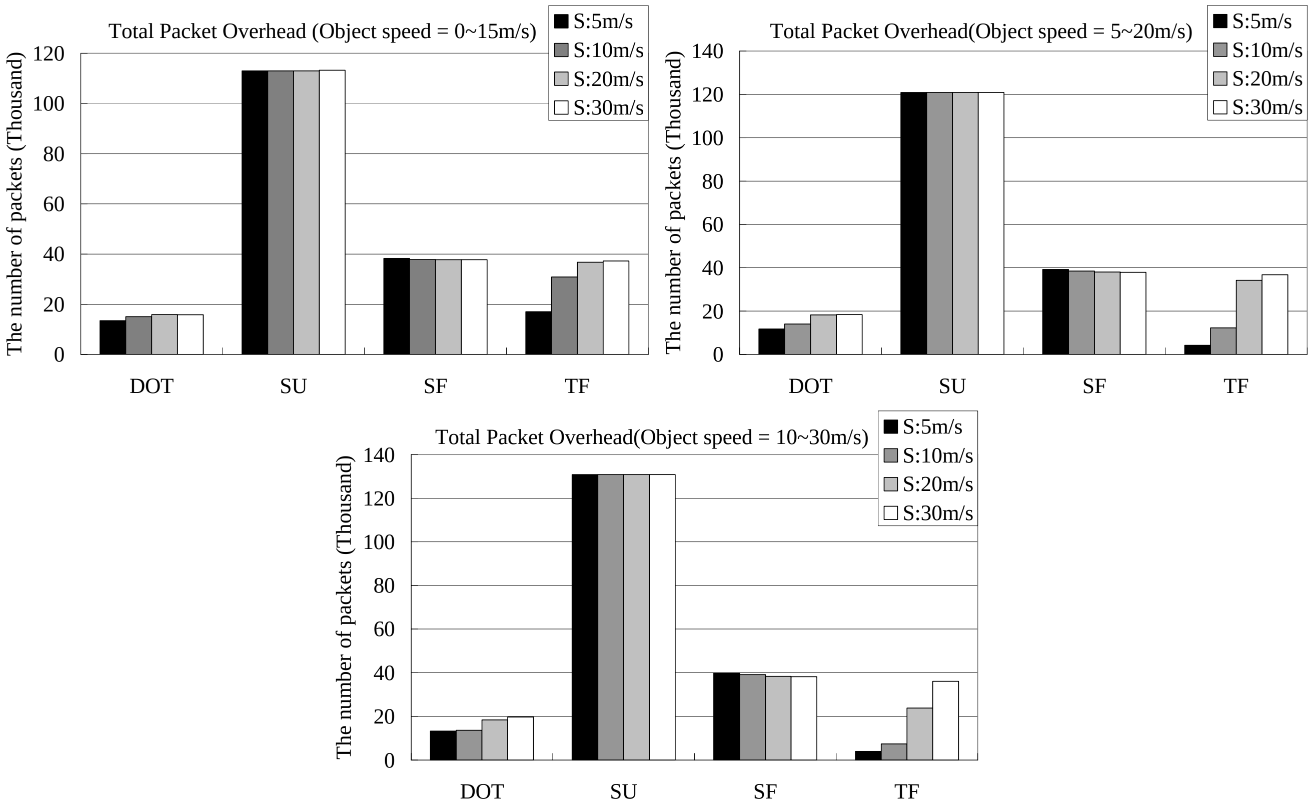


Fig. 17: The total packet overhead for different object moving scenarios.

IV. Conclusions

This work proposes a dynamical object tracking (DOT) protocol for sensor networks. The previous researches are almost force on how to track object accurately and they do not consider the query for mobile source. Additionally, they need not report the tracking information to user. The work is to focus on mobile user how to query target tracks and obtain the target position effectively. This protocol can be applied in tiny robot (like a bee) to chase an enemy or a wild animal. The scenario is that a source wants to chase a mobile target and it employs the sensor networks to track the target. A group of sensors forms an *envelopment-net* to besiege and to detect the target. When the target moves, the envelopment-net follows the target and a set of ingress nodes is kept. The sequence of ingress nodes is the target traces (*face-track*).

The ingress (beacon) node can pilot the source to the target position. The source can obtain the object information from the ingress node directly so this protocol can decrease the frequency of querying. The broadcast query would consume energy and increase packet overhead. Additionally, a shortening method for *face-track* is proposed to decrease the moving distance of source. It can improve the efficiency for chasing. The source can follow short path to chase the target. This method can also avoid track loop problem. By the experimental results, DOT protocol has better performance than the other flooding-based query methods.

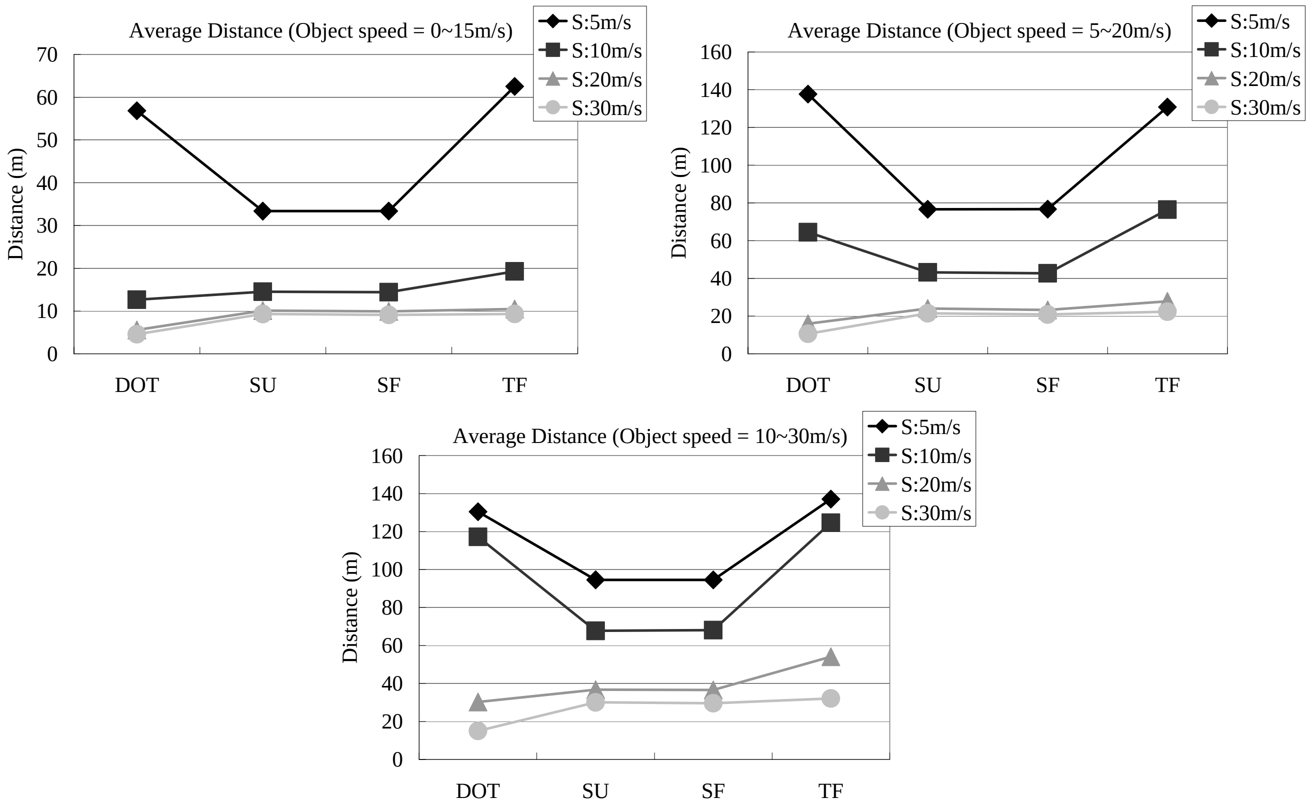


Fig. 18: The relative distances for different object moving scenarios.

REFERENCES

[1] M. d. Berg, M. van Kerveld, M. Overmars, and O. Schwarzkopf, Computational Geometry, Springer, 1998.

[2] K. Bhaskar, B. W. Stephen, and B. Ramon, “Phase Transition Phenomena in Wireless Ad-Hoc Networks,” in *Proceedings of IEEE GLOBECOM*, Volume 5, pp. 2921-2925, San Antonio, Texas, November 2001.

[3] R. R. Brooks and A. M. Sayeed, “Distributed Target Classification and Tracking in Sensor Networks,” in *Proceedings of the IEEE*, Volume 91, Number 8, pp. 1163-1171, August 2003.

[4] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, “Routing with Gurranteed Delivery in Ad Hoc Wireless Networks,” in *Wireless Networks*, Vol. 7, No. 6, pp. 609-616, 2001.

[5] Y.-S. Chen and S.-Y. Ann, “VE-Mobicast: A Variant-Egg-Based Mobicast Routing Protocol in Wireless Sensor Networks,” in *Proceedings of the 40th IEEE International Conference on Communications* (*IEEE ICC* 2005), Vol. 5, pp. 3020-3024, Seoul, Korea, May 2005.

[6] Y.-S. Chen and Y.-J. Liao, “HVE-mobicast: a hierarchical-variant-egg-based mobicast routing protocol for wireless sensornets,” in *Proceedings of the IEEE Wireless Communications and Networking Conference* (*WCNC*2006), Vol. 2, pp. 697-702, Las Vegas, NV, USA, April 2006.

[7] W.-P. Chen, J. C. Hou, and L. Sha, “Dynamic Clustering for Acoustic Target Tracking in Wireless Sensor Networks,” in *Proceeding of 11th IEEE International Conference on Network Protocols* (*ICNP*'03), pp. 284-294, Atlanta, Georgia, USA, November 2003.

[8] C.-Y. Chong, F. Zhao, S. Mori, and S. Kumar, “Distributed Tracking in Wireless Ad Hoc Sensor Networks,” in *Proceedings of the Sixth International Conference on Information Fusion* (*FUSION* 2003), pp. 431-438, Cairns, Australia, July 2003.

[9] CMU Monarch Group. CMU Monarch extensions to ns.

[10] M. Ding, D. Chen, A. Thaeler, and X. Cheng, “Fault-Tolerant Target Detection in Sensor Networks,” in *Proceeding of the IEEE Wireless Communications and Networking Conference* (*WCNC* 2005), Vol. 4, pp. 2362-2368, New Orleans, LA, USA, March 2005.

[11] I. Downard, “Simulating sensor networks in ns-2,” Naval Research Laboratory, NRL Formal Report 5522-04-10, 2004.

[12] L. J. Guibas, “Sensing, Tracking, and Reasoning with Relations,” *IEEE Signal Processing Magazine*, Volume 19, Number 2, pp. 73-85, March 2002.

[13] S. Goel and T. Imielinski, “Prediction-based Monitoring in Sensor Networks: Taking Lessons from MPEG,” in *ACM SIGCOMM Computer Communication Review*, Volume 31, Number 5, pp. 82-98, October 2001.

[14] Q. Huang, S. Bhattacharya, C. Lu and G. Roman, “FAR: Face-aware routing for mobicast in large-scale sensor networks,” in *ACM Transactions on Sensor Networks*, Vol. 1, No. 2, pp. 240-271, November 2005.

[15] Q. Huang, C. Lu, and G.-C. Roman, “Mobicast: Just-in-time multicast for sensor networks under spatiotemporal constraints,” in *Proceeding of the* 2*nd International Workshop on Information Processing in Sensor Networks*, pp. 442-457, Palo Alto, CA, USA, April 2003.

[16] Q. Huang, C. Lu, and G.-C. Roman, “Reliable Mobicast via Face-Aware Routing,” in *Proceedings of the IEEE Conference on Computer Communications* (*INFOCOM*'04), pp. 2108-2118, Hong Kong, China, March 2004.

[17] IEEE Standard Department. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE standard 802.11-1997.

[18] D.B. Johnson and D.A. Maltz, “Dynamic Source Routing in Ad Hoc Wireless Networks”, in *Mobile Computing*, edited by T. Imielinski and H. Korth, Chapter 5, pp. 153-181, Kluwer Pub-lishing Company, 1996.

[19] B. Karp and H. T. Kung, “GPSR: greedy perimeter stateless routing for wireless networks,” in *Proceedings of the* 6th *ACM/IEEE International Conference on Mobile Computing and Networking* (*MobiCom* 2000), 2000, pp. 243-254.

[20] E. Kranakis, H. Singh, and J. Urrutia, “Compass Routing on Geometric Networks,” in *Proceedings of* 11*th Canadian Conference on Computational Geometry*, pp. 51-54, Vancouver, Canada, August 1999.

[21] H. T. Kung and D. Vlah, “Efficient Location Tracking Using Sensor Networks,” in *Proceedings of the IEEE Wireless Communications and Networking Conference* (*WCNC* 2003), New Orleans, Louisiana, USA , March 2003.

[22] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, “Geometric Ad-Hoc Routing: Of Theory and Practice,” in *Proceedings* 22*nd ACM Symposium on the Principles of Distributed Computing* (*PODC* 2003), pp. 63-72, Boston, Massachusetts, July 2003.

[23] C.-Y. Lin and Y.-C. Tseng, “Structures for In-Network Moving Object Tracking in Wireless Sensor Networks,” in *Proceedings of the First International Conference on Broadband Networks* (*BROADNETS*'04), pp. 718-727, San Jose, California, USA, October 2004.

[24] D. Li, K. D. Wong, Y. H. Hu, and A. M. Sayeed, “Detection, Classification, and Tracking of Targets,” *IEEE Signal Processing Magazine*, Volume19, Number 2, pp. 17-30, March 2002.

[25] F. Mondinelli and Z. M. Kovacs-Vajna, “Self-Localizing Sensor Network Architectures,” IEEE Transactions on Instrumentation and Measurement, Volume 53, Number 2, pp. 277-283, April 2004.

[26] Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, and C.-F. Huang, “Location Tracking in a Wireless Sensor Network by Mobile Agents and Its Data Fusion Strategies,” *Computer Journal*, Volume 47, Number 4, pp. 448-460, 2004.

[27] VINT Project, “Network Simulator version 2 (NS-2),” Technical report, http://www.isi.edu/nsnam/ns, June, 2001.

[28] J.W. Jaromczyk and G.T. Toussaint, “Relative Neighborhood Graphs and Their Relatives,” in *Proceedings of the IEEE*, Vol. 80, No. 9, pp. 1502-1517, 1992.

[29] Y. Xu, J. Winter, and W.-C. Lee, “Prediction-based Strategies for Energy Saving in Object Tracking Sensor Networks,” in *Proceedings of the* 2004 *IEEE International Conference on Mobile Data Management* (*MDM*’04), pp. 346-357, Berkeley, California, January 2004.

[30] H. Yang and B. Sikdor, “A Protocol for Tracking Mobile Targets Using Sensor Network, Sensor Network Protocols and Applications,” in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pp. 71-81, Anchorage, Alaska, May 2003.

[31] Y. Zou and K. Chakrabarty, “Target Localization Based on Energy Considerations in Distributed Sensor Networks,” in *Proceedings of the* 1st *IEEE International Workshop on Sensor Network Protocols and Applications*, pp. 51-58, Anchorage, Alaska, May 2003.