Virtual Coordinate Based Co-moving Detection in Wireless Sensor Networks

Wei-Hua Chen[†], Guey-Yun Chang[†], and Jen-Feng Huang[‡]
[†]Dept. of Computer Science & Information Engineering, National Central University, R.O.C.
[‡]Dept. of Computer Science & Information Engineering, National Taiwan University, R.O.C.
E-mail:gychang@csie.ncu.edu.tw

***Abstract*—In mobile wireless sensor networks, it is important to understand the *moving pattern* of sensor nodes. There are a lot of application that need sensors nearby to co-work, such as data centric applications. Thus, knowing the *co-moving* nodes will be helpful to these applications. In this paper, we propose a co-moving sensor nodes detection algorithm based on virtual coordinate system. We use the geometric property of *independent set* to build our virtual coordinate system. The computation cost of building a virtual coordinate in this way is low and the result is relative accurate − just a coordinate rotation will it matches true coordinates. With the aids of virtual coordinate system, we can find out co-moving sensor node pairs efficiently and no precise localization scheme needed (for each sensor) in our works.**

***Index Terms*—Co-moving detection, Wireless sensor networks, Object tracking, Virtual coordinate**

I. Introduction

In mobile wireless sensor networks, sensor nodes might move arbitrarily or follow some *moving pattern*. Nearby sensors move within a distance and toward the similar direction [1], called *co-moving sensors*, can work together to achieve some goals that sensors cannot achieve alone. *Co-moving detection* can applied to social tracking [2]–[4], wild animal tracking and data centric applications. For hand-held computing devices are growing in these years like PDAs and smart phones, it's possible to acquire social interaction of people by the aid of these devices. For wild animals tracking, if wild animals are equipped with wireless sensors, their moving pattern can be traced. Understanding the co-moving relationship can help us discover social activities or family structures of wild animals. In data centric sensor networks, queries for the specific type of data can be sent directly to the corresponding *storage nodes* instead of flooding it to the whole network. Therefore, sensing nodes must co-move with data storage nodes so that the sensed data can be stored.

Co-moving detection is related to *object tracking*. There are works about object tracking in wireless sensor networks [5]–[7]. However, co-moving detection is not equivalent to object tracking. In tracking problem, the targets' moving tracks are recorded or there are some sources moving tower the targets. In some cases, there are a set of sources tracking a set of target. If we know the co-moving status of targets to be traced, we can classify targets into groups by their *moving direction*. Therefore, the mobile sources can focus on a group of targets rather than a single target. This will reduce track length for sources. Finding co-moving sensor nodes is obviously an important issue.

Intuitively, we can find co-moving sensors by analyzing their location information. By recording the location of each sensor at continuous time point, we can get a trace for each sensor. However, the cost of acquiring location information for each node is high. It requires that each node should be equipped with GPS devices or some accurate localization algorithms. The process might be time consuming, too. In mobile environment, the major problem of using location information in co-moving issue is that every sensor might be moving. Maintaining the latest location of one or two hop neighbors is a power consuming task. The communication cost and computing cost will be high for nodes are moving all the time. Therefore, it's not a good approach for finding co-moving nodes.

Some researches use the analysis of signals received by base stations (or access points) to discover relationship between sensors In an environment without any obstacles, we can use the signal strength to for localization and use the localization information to discover co-moving relationship. In an indoor environment with obstacles, there are researches using the signal pattern caused by obstacles for co-moving detection, like DECODE [1]. DECODE uses the correlation coefficient of signals from two mobile nodes to detect co-moving.

In our works, we proposed a *virtual coordinate based co-moving detection* algorithm (VCD). Since we concentrate on co-moving detection, we need only information that can help us understand the moving pattern of sensors. There is no need to acquire the actual location of each node, therefore we don't need GPS devices for each node nor the accurate localization schemes which are not cost efficient. In this paper, we exploit the geometric property of *independent set* to establish our *virtual coordinates system*. Nodes in the independent appear to be in some *regular pattern*. The pattern is helpful to our coordinates assigning scheme. If a sensor node finds a *maximum independent set* in its neighborhood, nodes in the set will cover most of its neighborhood. Since nodes in the independent set are not neighbors of each other, the theoretical maximum independent set of size 5 [8] will form a pentagon in the neighborhood. Therefore the regular pattern of the nodes' distribution can be used for virtual coordinate assigning. We assign each node a level from center of the coordinate and a location constant, therefore the system is like a *polar coordinate system*. Moving nodes can obtain location constants from its neighbor when they are moving, a well defined detecting rule using these information can have an efficient co-moving detection.

II. Preliminaries

A. System model

We assume that every sensor has a *unique* node id. Sensors are uniformly distributed in the network. Node N's communication range is called *neighborhood* of N and nodes in N's neighborhood area neighbors of N. Also, we assume the connection of nodes are undirected, that is, in a *connectivity graph*, an undirected link connection two nodes denotes to these two nodes can communicate with each other. Every node broadcasts hello message (i.e., beacon) with node id periodically. When a hello message arrives at a sensor node, the node record node id marked in hello message into its neighbor table.

All sensor nodes are not equipped with GPS system and sensors can be static or have ability to move. They are aware of its mobility status. We classify sensor nodes by their mobility status: static and mobile node. *Static nodes* denote nodes which are static. Some static nodes will be selected as *infrastructure nodes* (INs) in a coordinate system. *Mobile nodes* (MNs) denote nodes that are currently moving. *Initiator* is a subset of infrastructure nodes which establish a virtual coordinate system. Additionally, our work do not need know the mobility of sensors, hence, our co-moving detection algorithm can be applied to many different purpose.

B. Problem statement

The goal of co-moving detection problem is to find (h, p) $co\text{-}moving$ for each sensor. Also, (h, p) $co\text{-}moving$ define as below:

Definition 1: Define two sensors to be (h, p) $co\text{-}moving$ if they keep h hops and remain this relationship for p timeslots.

A group of the (h, p) co-moving sensors can communicate with each other within h hops during p timeslots. Also, a sensor's moving direction in the (h, p) co-moving sensors can the moving direction of the group (of the (h, p) co-moving sensors). Note that (h, p) co-moving sensors' direction can be used to find the mobility of a (social) group. Thus, if (h, p) co-moving sensors, we can be applied to many different applications which are mentioned in Section I.

III. Virtual coordinate based co-moving detection algorithm

A. Overview

In this section, we propose a *virtual coordinate based co-moving detection* algorithm (VCD) to find (h, p) co-moving. Our co-moving detection algorithm can be divided into two phases: *coordinate assignment phase* and *co-moving detection phase*. In coordinate assignment phase, initiators (referred to Section II-A) establish their virtual coordinate systems; nodes in this system will be assigned a *tuple of coordinate* (detailed later). Every node has a probability p_i to become an initiator. Before the initiator start establishing a coordinate system, it checks that there are no other initiators in its 2 hop neighbor. Static nodes selected as infrastructure nodes (INs for short) by initiators or previous level INs. Note that there are several initiators in a network, each establish its own coordinate system. If a node is located at the overlap area of two or more coordinate system established by different initiators, the node will have more than one coordinates, one for a initiator. In co-moving detection phase, mobile node (MNs for short) calculate its moving direction using coordinate information from INs. A MN can find some co-moving nodes by searching for MNs with similar moving direction a few hops away. Details of each procedure are described in III-B and III-C, respectively.

B. Coordinate assignment phase

In this phase, the initiator starts the establishment of coordinate system. Some static nodes are selected as INs of coordinate system and are assigned a tuple of coordinate.

1) Independent set discovery: Here, we find an independent set as INs since this is a compromise of accuracy and efficiency. Some methods for establishing virtual coordinate systems requires the information of boundary nodes of the whole network [8]–[10]. In [10], the system sees boundary nodes as a circle, and then assigns coordinates starting from nodes on the circle. This is reasonable since the locations of boundary nodes form a shape similar to a circle. However, finding boundary nodes of a network can be a time consuming job. Instead, we focus on finding nodes in some regular pattern in a node's neighborhood. If infrastructure nodes are too crowded, the accuracy of the coordinate system is low. An independent set in a node's neighborhood will be relatively far from each other. We exploit this feature to establish our coordinate system. If a virtual coordinate system is well-defined so that the coordinate can represent related relation in real location, such coordinate can be used for co-moving detection. Following, we detail our steps.

A initiator will find an independent and set them as INs. In an ideal case, an initiator can find an independent set of maximum size 5 in its neighborhood. The node in such the set will form a pentagon similar to a regular pentagon. Also, a maximum independent set means that each node not in the set must has at least a neighbor in the set. However, the maximum independent set problem is a NP-hard problem, so we have an approximation scheme to find a maximum independent set. When a node decides to become a initiator, it will broadcast a *neighbor information collecting* (NIC) packet to all its neighbors with initiator's id recorded. When a static node in initiator's neighborhood receive the NIC packet, it will send to the initiator a *neighbor information report* (NIR) packet recording its' *neighbor table*. After all nodes in the neighborhood of the initiator have reported their neighbor information to initiator, initiator will start to find an independent set from all its neighbors. Ideally, initiator can find a maximum independent set of size 5 as shown in Fig. 1.

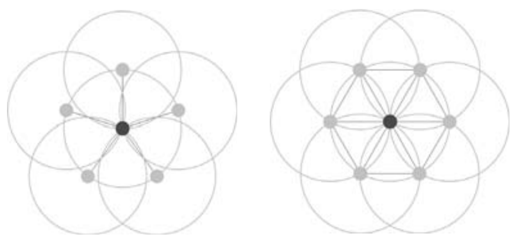

Fig. 1. Explanation of reason for finding independent set of size 5. (a). Black node denotes to the initiator. Gray nodes denote to a maximum independent set of initiator's neighbor. (b). 6 nodes in initiator's neighborhood cannot form an independent set. A regular hexagon is a common case.

Since initiator has all its neighbors' neighbor table, it can sort its neighbors by their number of common neighbors with initiator. Initiator picks the node with smallest number of common neighbors as a IN I_1. Then initiator eliminates all neighbors of I_1 from list, also adjusts the number of neighbors of each remaining nodes since neighbors of I_1 is no longer exists in the list. After the adjusting, initiator sorts the remaining node again and picks one node with current smallest number of common neighbors with initiator. This process repeats until all neighbors of initiator are eliminated from list. The nodes picked are not neighbor of each other and are *level* 1 INs, referred to Fig. 2.

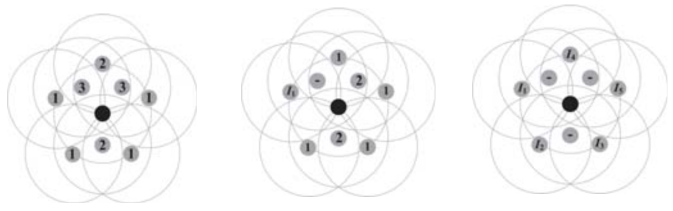

Fig. 2. An example of finding independent set. Black node denotes to the initiator and gray nodes denote to static nodes. Numbers of common neighbors with initiator are marked on the node. I_i denote to INs selected by initiator, in the order of i time to be selected. (a) The original status. (b) Initiator selects the I_1 as an IN. The node and all its neighbor will be marked with "-" and be eliminated from initiator's list. Initiator then adjusts other nodes' numbers of common neighbors with initiator. This process repeats until all initiator's neighbor are marked with "-". (c) At the end of the process, I_1 to I_5 are selected by initiator as level 1 INs.

The reason for picking the nodes with small number of neighbors is that by observation such nodes will locate at the boundary of initiator's neighborhood, therefore we can find an independent set with larger size since the overlapped area of the node and initiator are relatively small. If initiator finds an independent set of size 5, the distance of nodes in the set is far and nodes must appear in a form similar to a regular pentagon. It's reasonable to select these nodes as INs in a virtual coordinate system. Note that the approximate scheme's time complexity is $O(n^2)$ for n denotes to number of neighbors of a node. It's relatively low compared with a NP-hard problem.

2) Location constant assignment: Since initiator has all neighbor tables of its neighbors, it can know the clockwise sequence of INs by observing their common neighbors. Here the initiator picks one IN randomly and assigns it a location constant 1, then increases the location constant by 2, i.e., 5, and assigns it to the next node in counter-clockwise order until all INs are assigned a location constant. An example is shown in Figure 3. If a initiator fails to find an independent set of size 5, it will then pick nodes with the smallest common neighbor with nodes in the independent set. Total number of nodes that are picked as level 1 INs is always 5 therefore INs can form a shape similar to a regular pentagon.

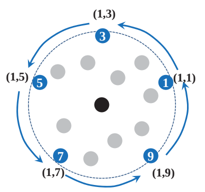

Fig. 3. Assign coordinates to INs. The black node denotes to initiator, nodes with number one it are INs, other gray nodes are static nodes in initiator's neighborhood.

Note that the process of finding level 2 INs to level 7 INs is similar to finding level 1 INs, the only different is the number of IN to be selected. For example of level 2 INs, Level 1 INs like the initiator. Level 1 INs collect neighbor table information of their neighbor. Ideally, a level 1 IN can find an independent set of size 5 in its neighborhood including its previous level IN (i.e., initiator). The static nodes set as INs should spread out in our virtual coordinate system, so level 2 INs selected by level 1 INs cannot locate in the region of initiator's (level 0) neighborhood. Also, the distance between a *next* level IN and initiator must be greater than distance between *current* level IN and initiator. Therefore, in an ideal case, only 2 nodes in the independent set are qualified to be INs. For example, referred to Fig. 4, in an ideal case I_2 can find an independent set of size 5: the initiator and other 4 gray nodes connecting to I_2, but only 2 located at the top of the figure are father to initiator than I_2, thus only those 2 are qualified to be next level INs.

From level 1 to 6, each IN selects 1 or 2 nodes according to rules described above as their next level INs. When a node is selected as I node, it inherits the location constant from its' previous level I node. If a node is selected by 2 previous level INs, the location constant will be the average value of those 2 location constants set by different INs. That will be an even number. If a nodes is selected by IN with even location constant and IN with odd location constant, the node will still remain its location constant a even one. For example, if a node is selected by nodes with location constant 1 and 2, the node will pick 2 as its location constant. This is because the number of nodes with location constant in even number is small. This action will have a balance effect to number of nodes with different location constant.

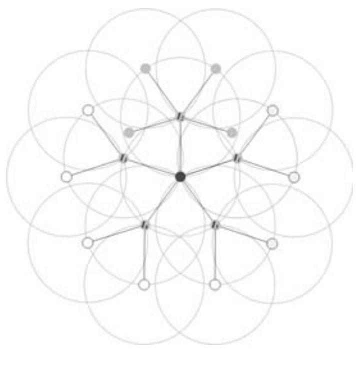

Fig. 4. An example of level 2.

After all enough level (at most 7) of INs are selected and assigned with location constants; each node in this coordinate system now has at least 1 tuple of virtual coordinate, (η, ζ), where level η denotes to radial coordinate, and location constant ζ denotes to angular coordinate. Note that initiator chooses 5 level 1 INs at 5 different directions and all following INs have location constant equal to their previous level INs, the location constants in this coordinate system can represent a specific direction from initiator. Direction constants also represent regions in this coordinate system. INs with the same location constant form an area, so there are at most 10 directions and region in this virtual coordinate system. Note that there might be several initiators establishing virtual coordinate system at the same time. The VCD algorithm allows coordinate systems to overlap, that is, an infrastructure node (IN) can belong to one or more coordinate systems.

C. Co-moving detection phase

A coordinate system enters co-moving detection phase when the process of establishing the virtual coordinate system is over, that is, the level of this system reaches desired level and all INs are found and are assigned a coordinate. When an static node is set to be a IN, it will start to broadcast its *location advertisement message* (LAM) periodically. All the INs in the same level will get the message that set them to I node almost at the same time, and they starts to broadcast their LAM right after they get the message. To avoid collision of LAM, the node waits for a constant time before sending hello messages, where the waiting time is node id $\times$ c ms, for c is a constant.

When a mobile node goes across the coordinate system, it will record the coordinate recorded in LAM sent by INs into its *LAM list*. The LAM list is a list of 4-tuple element, (*arrival time*, *IN id*, *level*, *location*). A mobile node calculates its moving direction according to the LAM list while it has received enough LAM. The mobile node will make pairs of every elements in the LAM arrive in the most recent period and elements arrive in previous period. That is, if a mobile receives 3 LAMs from 3 different INs at the current period and it receive 2 at the previous period, then the mobile node have to perform $3 \times 2 = 6$ calculations to know the *potential moving direction*. Note that if a LAMs of the same region appears in both 2 timeslots, mobile nodes will ignore the pair since we can not imply any moving direction from such information, as shown in Figure 5.

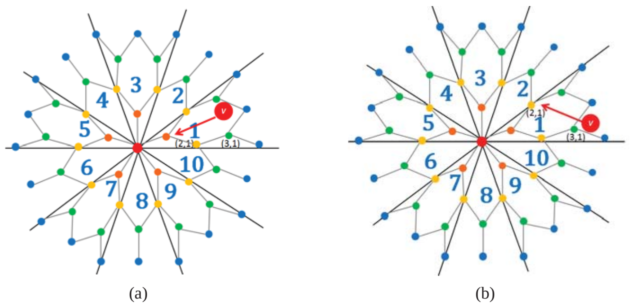

Fig. 5. Explanation of the elimination of pairs consists of LAM from the same region.

We might imply that a node is moving toward initiator if it gets LAMs from different period, same region and decreasing levels. However, nodes moving in a region also get LAMs like that, too. In the example shown in Figure 5(b), node v is moving in direction between 4 and 5. But if we use these 2 LAMs to calculate its moving direction and might get a location of 6. Therefore, we eliminate the LAMs from the same region to calculate moving direction. In the example mentioned before, mobile can have 6 results for the calculation of moving direction. Then the mobile node takes the average value of the results as its current moving direction. (The formula of moving directions are detailed later.)

When a mobile node MN_1 wants to find co-moving nodes, it will have a local flooding of a *co-moving request* (CREQ) packet to its h hops moving neighbors. The CREQ packet records the mobile node's node id and moving direction. When a moving neighbor MN_2 get CREQ and founds that the difference between its moving direction and MN_1's moving direction is smaller than d, it considers itself as a co-moving node candidate of MN_1 and sends a *co-moving reply* (CREP) packet back to MN_1 for p timeslots, one CREP for a period. MN_1 than calculates *co-moving score* (Co-S) of for each co-moving node candidate. The Co-S for a CREP that MN_1 receives is defined as follow:

$$Co\text{-}S = (d_t - d) \times (h_t - h), \tag{1}$$

where d_t denotes the maximum acceptable difference of moving directions of MN_1 and MN_2, d is the difference between the moving direction of MN_1 and MN_2, h_t denotes to the maximum acceptable hop count between MN_1 and MN_2, h denotes to the hop count between MN_1 and MN_2. Note that the moving direction comes from location constant, since the location constant is in a cyclic order, moving direction 1 and 10 are adjacent, the difference between moving direction 1 and 10 will be considered as 1. The *total co-moving score* (TCS) for a co-moving node candidate is the accumulation of CSs calculated from CREP of a mobile node that MN_1 receives for p timeslots. The $(p+1)$ timeslots after MN_1 floods CREQ,

MN_1 sorts its co-moving node candidates by their TCS. The MN_1 then select N nodes with highest TCSs as its co-moving nodes, and the value of N depends on applications.

Below, we detail the formula of a mobile node's moving directions. In Fig. 6, the coordinate system is divided into 10 regions. The location constants 1 to 10 also represent 10 directions in this coordinate system originated from the initiator. When a mobile node selects a pair of LAM from two continuous timeslots in LAM list, it will compare levels and location constants marked in 2 LAMs. It uses level and location constant marked in the previous LAM as its current level and current location, whereas level and location constant marked in later LAM as its new level and new region. Following, we explain the process of moving direction calculation in an example shown in Fig. 6. If the coordinate recorded in LAM arrives in previous period is (4,1) and coordinate recorded in current period LAM is (4,2), that means the node goes across region 1 to 2 and remain a same distance with the initiator. In this case, we can imply that the node is going by a direction similar to direction 4, that is, current region plus 3. If the case is (4,1) and (3,2), the node crosses region 1 and 2 and is moving toward the initiator. Moving direction is similar to direction 5, current location plus 4. We use changes of level and location constant to know where the mobile node is going to. We list all possible combination of current and new values that can be used to calculate moving direction below.

$$moving\ direction = RC \times (PR + 3 + LC), \qquad (2)$$

where RC denotes to change in region, PR denotes to previous region number, LC denotes to change in level. Values of RC and LC are defined as follows:

$$RC = \begin{cases} 1, \text{new regin} > \text{current region or from region 10 to 1} \\ -1, \text{new regin} < \text{current region or from region 1 to 10} \\ 0, \text{region remains the same} \end{cases} \qquad (3)$$

$$RC = \begin{cases} 1, \text{new level} < \text{current level} \\ -1, \text{new level} > \text{current level} \\ 0, \text{region remains the same} \end{cases} \qquad (4)$$

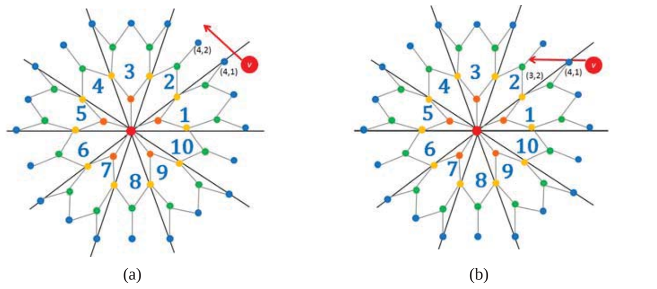

Fig. 6. An example of direction calculation. (a). The mobile node V goes from region 1 to region 2 without level change. (b). V goes from region 1 to region 2 with level decreases by 1.

Note that location constant increasing means the mobile node is going in a counter-clockwise order to initiator. The case includes going from region 10 to 1. Note also that we don't use LAMs of the same direction constant as the references of moving direction because mobile nodes often receive LAM from INs in the same region but different level.

IV. Performance Evaluation

In this section, we show the performance of our VCD algorithm by simulations. We use ns-2.34 to simulate all activities in the networks and use nsg [11] to generate scenes. The Sizes of networks and scenes vary for different simulations. We will describe all the details in the following sections. The Section IV-A shows the detection rate of our VCD algorithm. The Section IV-B shows the relationship of network density and number of packets.

A. The detection rate of VCD algorithm

In this section, we show the detection rate of VCD. We define detection rate as the ratio of the number of pairs of (h, p) co-moving nodes detected by VCD and detected by brute force, range of h hops and remain this relationship for p timeslots. In our simulation, h is set to 3 and p is set to 3. There are 300 sensor nodes (including 3 initiators) are uniformly distributed in the network. The Network density D is the average number of neighbors of a node. Following, we simulated different D: 8, 10, 12, 14, 16, 18, where the range of the scene are about 543×543, 485×485, 443×443, 410×410, 383×383, and 361×361 m^2. The communication range of each sensor is set to $50m$. The speed of mobile nodes is 1 m/s. The detection rate is shown in Fig. 7.

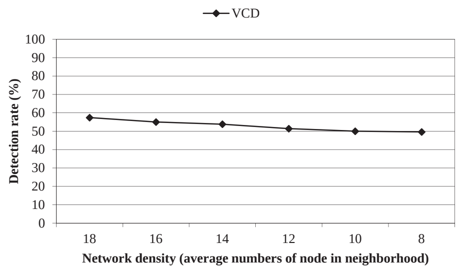

Fig. 7. Detection rate of VCD.

The Fig. 7 shows that higher D comes with higher detection rate. Networks with higher density have higher probability to forward their CREQ and CREP, thus have higher detection rate. Note that although higher density with higher detection rates, the detection rates vary in a small amount. For we restrict the level and size of coordinate system, the number of static nodes selected as infrastructure nodes in different density don't vary a lot. For this reason, nodes in different density network can have almost the same probability to receive LAM packets.

B. Network density and packet amounts

In Fig. 8, we can see that the average packet number per node is in a inverse ratio of network density, at most close to 14, at least close to 8. Since the cost of establishing a coordinate system and the number of LAMs are not so different among different scene as shown if Fig. 9, we can imply that the main cost of VCD is the cost of finding co-moving nodes, that is, the CREQ packets and the CREP packets. When D is large, lots of nodes will receive CREQ from other nodes and forward it when hop count of CREQ is still smaller than h. Co-moving nodes then will have higher probability to be found co-moving, thus the number of CREP also increases in dense network. This is the main reason that networks with higher density will have high packet number. If Initiator is located near the boundary, it cannot establish a full coordinate system at the side of boundary. Also, if there are a lot of nodes co-moving, the CREQ and CREP will be forwarded massively.

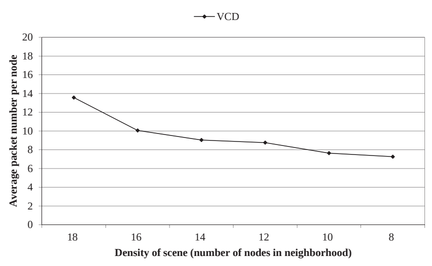

Fig. 8. Relationship of network density and average packet number.

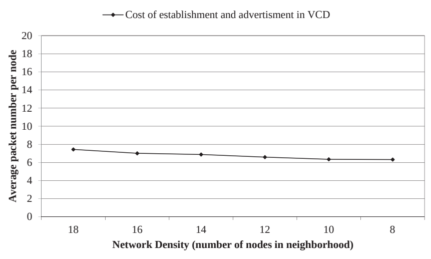

Fig. 9. The cost of establishing VCD and advertisement in VCD.

In Fig. 9, we can know that there are only slightly differences for different networks with different D to establish and maintain VCD. The level of a coordinate system is restricted and infrastructure nodes in a coordinate system will send LAM periodically, the only difference between Ds is the numbers of nodes that reply the neighbor information collect (NIC) packets. In dense networks, there are more neighbors in a nodes neighborhood. When a node sends NIC to its neighbors, a lot of nodes will respond to this node. Thats why dense networks have a little bit higher number of packets for establishing a coordinate system.

V. Conclusions

In wireless sensor networks, knowing the moving pattern is important and is crucial for some applications like social engineering and tracking. Finding co-moving pair of nodes is not an easy task. Some researches use changes of signals to imply the moving pattern of mobile nodes, which is not impractical.

In our work, we propose a co-moving detection algorithm based on the aid of virtual coordinate system. We exploit the geometric property of independent set to establish our virtual coordinate system. By an approximation, we can find a maximum independent set in a efficient way and then assign coordinates to nodes. With the help of these coordinates, a mobile node can derive its moving direction, hence and find other nodes with the similar moving direction. Network density and speed of mobile nodes can only affect the result of detection slightly; therefore we have a robust system for co-moving detection.

References

[1] G. Chandrasekaran, M. A. Ergin, M. Gruteser, R. P. Martin, J. Yang, and Y. Chen, "Decode: Exploiting shadow fading to detect comoving wireless devices," *IEEE Transactions on Mobile Computing*, vol. 8, no. 7, 2009.

[2] M. Musolesi, S. Hailes, and C. Mascolo, "An ad hoc mobility model founded on social network theory," in *ACM/IEEE MSWiM*, 2004.

[3] E. M. Daly and M. Haahr, "Social network analysis for information flow in disconnected delay-tolerant manets," *IEEE Transactions on Mobile Computing*, vol. 8, no. 5, 2009.

[4] W. He, Y. Huangt, K. Nahrstedt, and BoWu, "Message propagation in ad-hoe-based proximity mobile social networks," in *IEEE PERCOM*, 2010.

[5] Z. Zhong, T. Zhu, D. Wang, and T. He, "Tracking with unreliable node sequences," in *IEEE INFOCOM*, 2009.

[6] C.-Y. Lin, W.-C. Peng, and Y.-C. Tseng, "Efficient in-network moving object tracking in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 8, 2006.

[7] H.-W. Tsai, C.-P. Chu, and T.-S. Chen, "Mobile object tracking in wireless sensor networks," *Computer Communications*, vol. 30, no. 8, 2007.

[8] M.J.Tsai, H.Y.Yang, and W. Huang, "Axis-based virtual coordinate assignment protocol and delivery-guaranteed routing protocol in wireless sensor networks," in *IEEE INFOCOM*, 2007.

[9] C. A., C. S., D. S., and U. A., "Gps free coordinate assignment and routing in wireless sensor networks," in *IEEE INFOCOM*, 2005.

[10] B. Leong, B. Liskov, and R. Morris, "Greedy vrtual coordinates for geographc routing," in *IEEE ICNP*, 2007.

[11] "Nsg: http://sites.google.com/site/pengjungwu/nsg."

[12] S. C.-H. Huang, P.-J. Wan, C. T. Vu, Y. Li, and F. Yao, "Nearly constant approximation for data aggregation scheduling in wireless sensor networks," in *IEEE INFOCOM*, 2007.

[13] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "Ght: A geographic hash table for data-centric storage," in *ACM WSNA*, 2002.

[14] T. Watteyne, I. Auge-Blum, M. Dohler, and S. Ubeda, "Dominique barthel, centroid virtual coordinates - a novel near-shortest path routing paradigm," *Computer Networks*, vol. 53, no. 10, 2009.

[15] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *ACM MOBICOM*, 2003.

[16] K. Muthukrishnan, M. Lijding, N. Meratnia, and P. Havinga, "Sensing motion using spectral and spatial analysis of wlan rssi," in *EuroSSC*, 2007.

[17] T. Sohn, A. Varshavsky, A. LaMarca, M. Y. Chen, T. Choudhury, I. Smith, S. Consolvo, J. Hightower, W. G. Griswold, and E. de Lara, "Mobility detection using everyday gsm traces," in *IEEE/IFIP UbiComp*, 2006.

[18] J. Krumm and E. Horvitz, "Locadio: inferring motion and location from wi-fi signal strengths," in *MOBIQUITOUS*, 2010.

[19] G. Chandrasekaran, M. Ergin, M. Gruteser, R. Martin, J. Yang, and Y. Chen, "Decode: Detecting co-moving wireless devices," in *IEEE MASS*, 2008.