Front. Comput. Sci., 2016, 10(6): 1000–1011
DOI 10.1007/s11704-015-4571-6

RESEARCH ARTICLE

DPHK: real-time distributed predicted data collecting based on activity pattern knowledge mined from trajectories in smart environments

Chengliang WANG (✉)[1,2], Yayun PENG[2], Debraj DE[3], Wen-Zhan SONG[3]

1 Key Laboratory of Dependable Service Computing in Cyber Physical Society, Chongqing University, Chongqing 400044, China
2 College of Computer Science, Chongqing University, Chongqing 400044, China
3 Department of Computer Science, Georgia State University, Atlanta GK 30303, USA

Abstract In this paper, we have proposed and designed DPHK (data prediction based on HMM according to activity pattern knowledge mined from trajectories), a real-time distributed predicted data collection system to solve the congestion and data loss caused by too many connections to sink node in indoor smart environment scenarios (like Smart Home, Smart Wireless Healthcare and so on). DPHK predicts and sends predicted data at one time instead of sending the triggered data of these sensor nodes which people is going to pass in several times. Firstly, our system learns the knowledge of transition probability among sensor nodes from the historical binary motion data through data mining. Secondly, it stores the corresponding knowledge in each sensor node based on a special storage mechanism. Thirdly, each sensor node applies HMM (hidden Markov model) algorithm to predict the sensor node locations people will arrive at according to the received message. At last, these sensor nodes send their triggered data and the predicted data to the sink node. The significances of DPHK are as follows: (a) the procedure of DPHK is distributed; (b) it effectively reduces the connection between sensor nodes and sink node. The time complexities of the proposed algorithms are analyzed and the performance is evaluated by some designed experiments in a smart environment.

Received December 16, 2014; accepted September 6, 2015

E-mail: wangcl@cqu.edu.cn

Keywords trajectory prediction, sensor data mining, wireless sensor networks, smart environments, hidden Markov model

1 Introduction

Sensing, data collection and data processing are three basic functions in wireless sensor network (WSN). Particularly, the data collection plays an important role in building a bridge between the processing center and the physical environment sensed by sensor nodes. In the progress of data collection, when hundreds or even thousands of sensor nodes periodically report their sensed data to the sink node, high latency or data loss may take place for the sink node cannot afford to process so much data in one time. For example, In Fig. 1(a), sensor nodes s_3, s_9, s_{10}, s_{13}, s_{14}, s_{15}, s_{24}, s_{27} and s_{29} are triggered, and they will send their data to the sink node at the same time. If data of s_3 is lost or data of s_{10} and s_9 is delayed being processed, it will be hard for applications to know UserOne is under s_3 or the applications may get that UserTwo walks from s_{10} to s_9 after a period. Thus, how to efficiently aggregate the sensed data from scattered sensor nodes with low communication traffic around the sink node is one of the most critical challenges in the applications of resource-limited sensor networks.

In recent years, much research effort has been devoted to efficient data gathering in WSNs and various types of data gathering mechanisms have been proposed for large-scale sensor networks. Most of them focused on data collection based on mobile data gathering schemes [1,2], efficient relay routing [3], multi-channel scheduling [4] and multiple sinks schemes [5]. However, in indoor applications or services like Smart Office and Smart Wireless Healthcare, the methods above have some limits: i) mobile data gathering schemes is not fit in indoor services for the carrier of mobile sink node is not convenient to work in indoor environment; ii) in efficient relay routing, the data may be stored in some intermediate sensor nodes for a while and the result is that this method is not real-time enough; iii) multi-channel scheduling and multiple sinks schemes may still face the problem in their basic element-one channel or one sink when the data needed to be sent is too much. So in this paper, in order to reduce the communication traffic load of the sink node in a smart environment in which applications can conduct activity recognition, we propose a real-time and distributed method-DPHK (data prediction based on HMM according to activity pattern knowledge mined from trajectories) which is inspired by our former research [6] and the study of human activity prediction [7–9]. In addition, our designed system DPHK can also be applied jointly with the above methods to make the effect much better. Figure 1(a) shows the testbed environment used in this work.

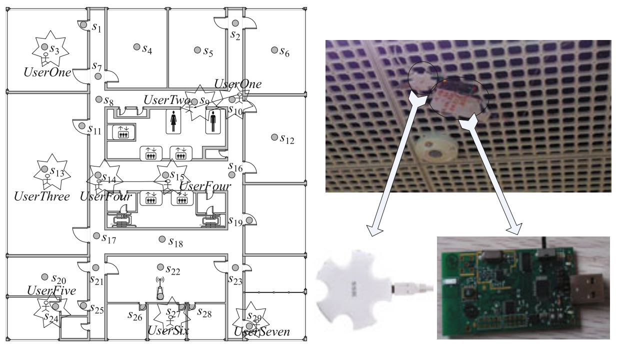


Fig. 1 (a) Layout of the smart environment of our study; (b) the USB hub which powers several sensor nodes and the sensor nodes used in our work

In our proposed method, we have two assumptions:

- People will not walk back and forth in the environment during one trajectory for the scenario takes place not very frequently in daily life and routine office. While if people walk back, we also think this is a new trajectory for the reverse often takes place in the end of an activity. For example, going to bathroom can be divided into going to the bathroom and coming back from the bathroom. People will walk back after going to the bathroom and before coming back from the bathroom;
- People could walk in the environment in different speed but everyone's walking speed is treated as a constant or slow-changing variable [6]. What's more, people may change their walking speed when something sporadic or urgent happens, however, they may still have a constant or slow-changing but new speed before the event ends or they burn out.

Our motivation of this paper is to solve two main problems: (i) how to reduce the incoming data traffic load of sink node, (ii) how to make distributed prediction in real-time. The main contributions of our work are as follows:

- Resigning algorithm for using HMM to make multihop prediction about the incoming trajectory based on the multihop historical knowledge and received message;
- Designing algorithm with distributed to make prediction with the localized computing and knowledge of each sensor node;
- Conducting performance evaluation experiments to test the feasibility and effect of DPHK.

The rest of the paper is organized as follows. The existing works in the literature is discussed in Section 2. The model definition and system design for Smart environment is presented in Section 3. In Section 4, the core of DPHK which is called PSM is discussed in detail. Then in Section 5, the performance of our proposed framework is evaluated with real data set collected from a smart workplace environment. Finally, in Section 6, we conclude this work and discuss the future research directions.

2 Related work

In this section, we review some recent works on data collection in WSNs. Some researchers improve the data collection rate in physical means such as increasing the number of sink node. In Ref. [1], the authors proposed a data gathering cost minimization (DaGCM) framework. In this method, the mobile collector with multiple antennas completes the mobile data gathering, which is constrained by flow conservation, energy consumption, link capacity, compatibility among sensor nodes and the bound on total sojourn time of the mobile collector at all anchor points. In Ref. [4], the authors studied the data collection rate under the many-to-one communication paradigm known as convergecast. They combine transmission power control with TDMA scheduling, and use multiple frequency channels to enable more concurrent transmissions. They also construct network topologies with specific properties that help in further enhancing the data collection rate. A method using multiple sinks was proposed in Ref. [5]. In this method, the authors presented an algorithm to select a subset of sensor nodes to represent the whole multi-sink sensor network based on the spatial correlated sensing readings. In this algorithm, only these representatives named sources need to upload their data to the chosen sinks. However, these approaches only reduce the scale of the problem a sink node faces to a certain extent which means the problem will also have a serious impact on these approaches when the scale of the problem is too great.

Other research works are concentrated on data processing, optimization and compression to reduce the communication traffic in the WSN. A communication traffic reduction method by delivering predicted sensor data was proposed in Ref. [2]. In this method, at the base station, the system predicts the sensor data value in each area in the next round by using stored sensor data in the sensor data base. The mobile sink broadcasts predicted sensor data to each sensor node in its route. Only sensor nodes whose sensing data exceeds the admissible error margin from the predicted sensor data transmit their data. In Refs. [10,11], the authors proposed a packet merging approach. In their method, the sensor node combines several smaller records into a few large ones in specific time and forwards the larger packets, thus reduces the number of packets and there by the communication cost. CONCH, short for constraint chaining was proposed in Ref. [12]. In CONCH, a monitored node triggers a report if its value changes and a monitored edge which stands for the relationships between neighboring values triggers a report if the difference between its nodes' values changes. These methods indeed do a good job in reducing communication traffic, but they cannot often offer real-time data for these reasons: i) the sensed data may be stored in some sensor nodes which do the packet merging for a period of time; ii) the mobile sink will not send the data to the process center until it finishes collecting the sensed data on its route; iii) the sensor nodes do not forward the sensed data as the sensed data does not change much.

While in an indoor smart environment, the feasibility of activity recognition indicates that human activity is regular. In Refs. [13,14], the authors presented a distributed model for recognizing user activity sequences. In the model, they studied the temporal and spatial relationships in the activation sequence and took use of the regular relationships to recognize activity. In view of the above-mentioned relationships and the thoughts in Ref. [7], the authors proposed an activity-based continuous-time Markov model to define and predict the human movement patterns [15]. The main work of the method is to define the user's activity patterns, based on the monitoring of user's past activity provided in a specified time interval, to make a prediction of next activity. The transition between activities is modeled as a Markov chain to predict $(n + 1)$th day location using n days information. However, these methods are not adopted in data collection application.

3 Methodology

In this section, we mainly introduce the knowledge model that is stored in sensor nodes used in our framework. We also briefly describe the problem of study.

3.1 Model definition

Here we model the network of our smart environment and define the knowledge structure that is stored in sensor nodes.

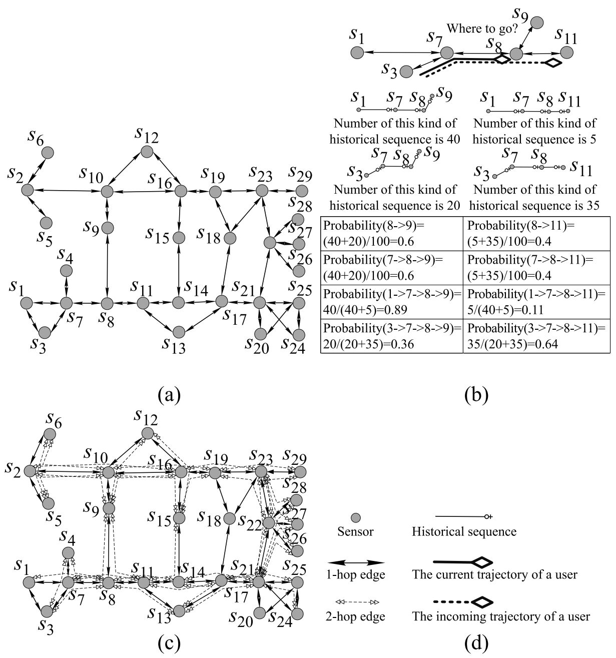
3.1.1 Wireless sensor network model

Let $G = (V, E)$ be the undirected topology graph of the network where $V = \{s_1, s_2, \ldots, s_i, \ldots, s_N\}$ is the set of sensor nodes which are installed across the Smart Environment physical space and $E = \{e_1, e_2, \ldots, e_j, \ldots, e_M\}$ is the edge set of G. An edge $e_j = (s_x, s_y)$ exists when s_x and s_y can physically be reached directly from each other and they can also communicate with each other. N is the number of sensor nodes and M is the number of edges. (G of our environment is shown in Fig. 2(a)).

Table 1 List of parameters and their description

Parameter	Definition
D_i	the number of e connected with s_i
AP	two sensors connected by an edge are APs of each other
MDS	multiple direction sensor, a sensor whose D is greater than 2
UDS	unique direction sensor, a sensor whose D is not greater than 2
θ	the threshold is used to judge whether the predicted result is acceptable
ME	message used to tell sensor where a user ever passed
PSID	previous sensor ID in ME
CSID	current sensor ID in ME
q(st)	state at step st
CTS	current triggered sensor in a step ST
PTS	sensor predicted to be triggered by CTS
RTS	really triggered sensor in step (ST + 1) or (ST + 2)

While thinking about the situation in Fig. 2(b): a user reaches s_8 passing by s_3, s_7. If s_8 doesn't know the user comes from s_3, it may choose s_9 as the next walking location; otherwise it may choose s_{11}. And this means a sensor node can make a prediction more accurately if it knows more historical information about where a user ever passed. However, taking the storage capacity and computing power of a sensor node into consideration, here we consider 2-hop activity transition relationship among the nodes which means a sensor node knows only previous two positions where a user ever passes. And we assume user won't walk back during one trajectory, for UDS like s_9 in Fig. 2(a), the transition probability of a user walking to s_{10} is 100% when the user comes from s_8. So 2-hop activity transitions which need sensor node to make a choice only occur in MDS like s_{11}, s_{16}, also the crossing of two P-R *path*s.


Where to go?
Number of this kind of historical sequence is 40
Number of this kind of historical sequence is 5
Number of this kind of historical sequence is 20
Number of this kind of historical sequence is 35
(a)
(b)
Sensor
Historical sequence
1-hop edge
The current trajectory of a user
2-hop edge
The incoming trajectory of a user
(c)
(d)

Probability(8->9)=(40+20)/100=0.6	Probability(8->11)=(5+35)/100=0.4
Probability(7->8->9)=(40+20)/100=0.6	Probability(7->8->11)=(5+35)/100=0.4
Probability(1->7->8->9)=40/(40+5)=0.89	Probability(1->7->8->11)=5/(40+5)=0.11
Probability(3->7->8->9)=20/(20+35)=0.36	Probability(3->7->8->11)=35/(20+35)=0.64

Fig. 2 (a) Basic graph G constructed for the smart environment layout; (b) Explanation of why we adopt EG; (c) Practical EG of G. Solid lines indicate activity transition between nodes 1-hop away, while dashed lines indicate activity transition between nodes 2-hop away. Nodes 1-hop to each other can be physically reachable without triggering any other node

Definition 1 (**P-R path**) If there is a path connecting with MDS s_i and MDS s_j and on the path no other MDS exists or only *UDS* exists, we say s_i and s_j are Path-Reachable, P-R for short. This path is called a P-R path. For example, $s_8 \rightarrow s_9 \rightarrow s_{10}$ is a P-R path, s_8 and s_{10} are MDSs and s_9 is a *UDS*.

The notion of activity transitional relationship among the nodes or states is presented and used in this work in the form of an Extended Graph or EG.

EG (extended graph) In EG = (V, E, E', A, A'), weighted 2-hop edge $e' \in E'$ denotes a pair of sensor nodes that can physically be reached from each other through two P-R paths. The weights a and a' of edge e and e' respectively denote direct 1-hop and indirect 2-hop activity transition between the nodes. If $a' = 0$, the corresponding e' doesn't exist. (Figure 2(c) shows EG of G.)

3.1.2 Data structure

In order to generate predicted data, a sensor node needs to know where a user may go next and when the user will reach the next position. What's more, when a sensor node has predicted where a user will go, as the P-R path through this sensor node is one-way traffic, it can compute the predicted data of the last UDS(s) in the same P-R path without new prediction of the next location. For example, when a user walk to s_9 from s_8, his next destination must be s_{10} in one trajectory. 2-hop prediction only happens at the crossing of two P-R paths, and then the special storage structure of distance information and access probability information should be defined.

Definition 2 (**P-RD**) P-R Distance between two sensor nodes, short for P-RD, is defined like this:

P-RD=m, one can reach the other through m P-R paths

ADLL (access distance linked list) To every sensor node, it stores all the distance information of other sensor nodes which are Ω *P-RD* from it and have the Ω-hop transition relationship with it. Ω is variable which is determined by the actual demand and the storage capability of the sensor node.

ADP (access direction probability) To every sensor node, it stores all the 1-hop and 2-hop activity transition probability between it and sensor nodes in its ADLL.

In our model, $\Omega = 2$ in view of the storage capability and computing power of the sensor node. Figure 3 shows the theoretic structure of ADLL of s_{21} in Fig. 2(c). We can see that the two s_{24}s have different distances for they have different P-RDs: one is the distance of s_{21} to s_{24}, the other is the distance of s_{21} to s_{25} then to s_{24}.

3.2 Problem description

When some trajectories are generated in the smart environment, all the sensor nodes triggered will send their sensed data to the sink node. We define the times the sink node receives sensed information from other sensor nodes in one time as the pressure of receiving data on sink node, short for PRDOS. And the greater the PRDOS is, the worse the data loss is or the higher the latency is.

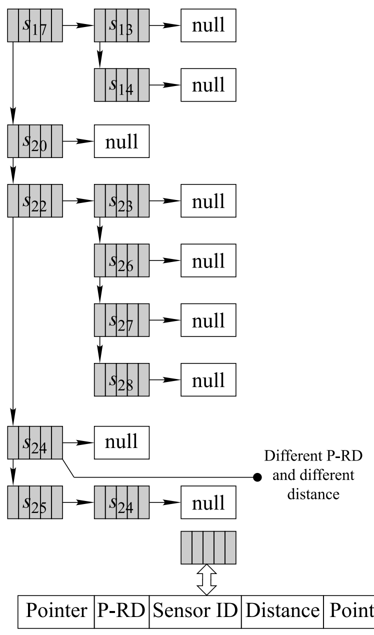


Fig. 3 The structure of ADLL of s_{21} in Fig. 2(c)

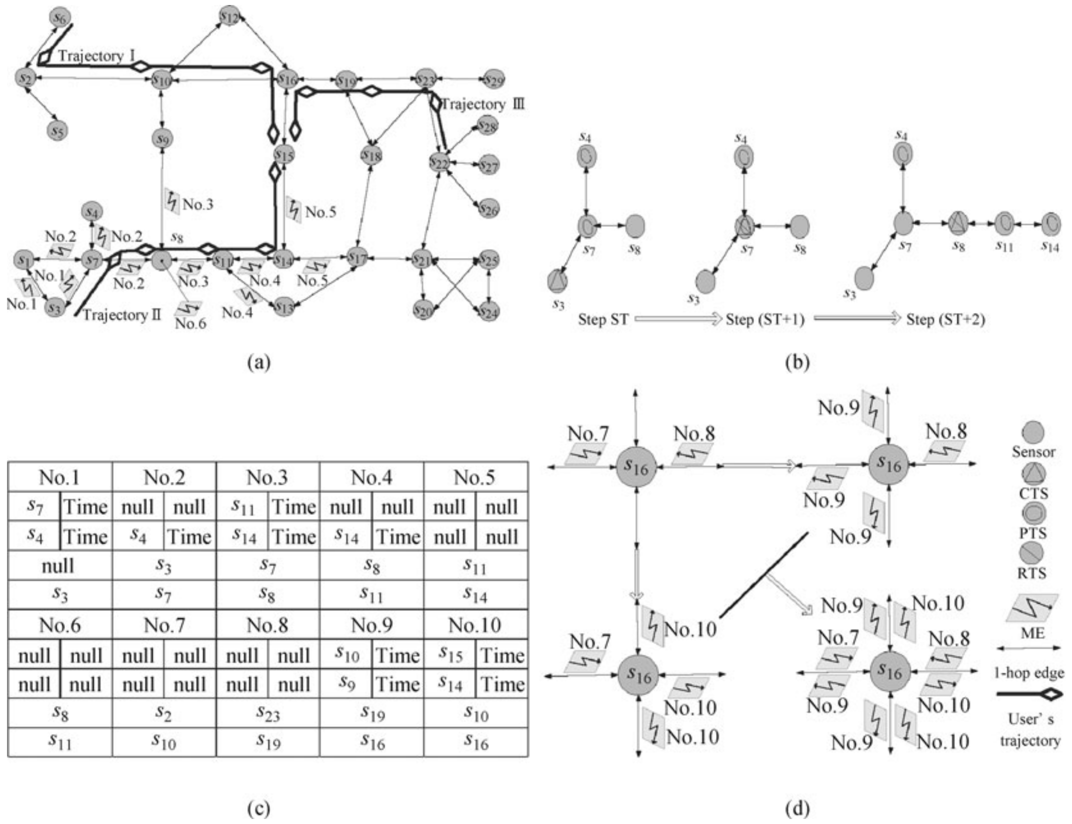
Research purpose In order to reduce PRDOS, each sensor node can send predicted results of the next Ω sensor nodes a user will pass instead of the real triggered data of these sensor nodes through received ME based on its ADP and ADLL.

3.3 Illustrating example

Figure 4 illustrates how proposed DPHK solves the above-mentioned problem. In Trajectory II, all the triggered sensor nodes should send their sensed data to sink node before using DPHK and the total times is 6 times. While after using DPHK, take s_8 as an example, it is not a PTS of s_3 but a RTS and this means s_3 has made a wrong prediction, so s_8 needs to make a new prediction. It applies HMM to predict s_{11} and s_{14} as PTS according to ME-No.2 and sends its sensed data and the predicted results of s_{11} and s_{14} to the sink node. Then when s_{11} becomes RTS, it compares its sensed data with the predicted result received from s_8 to decide whether it needs to send a modified data to the sink node and makes a new prediction. And in our system, the total times of Trajectory II is 3 times.

Table 2 The structure of ME

The structure of ME in DPHK-TT		The structure of ME in DPHK-OT
Predicted sensor ID 1	Predicted triggered time 1	Predicted sensor ID 1
…	…	…
Predicted sensor ID X	Predicted triggered time X	Predicted sensor ID X
PSID		PSID
CSID		CSID



No.1		No.2		No.3		No.4		No.5	
s_7	Time	null	null	s_{11}	Time	null	null	null	null
s_4	Time	s_4	Time	s_{14}	Time	s_{14}	Time	null	null
null		s_3		s_7		s_8		s_{11}	
s_3		s_7		s_8		s_{11}		s_{14}	
No.6		No.7		No.8		No.9		No.10	
null	null	null	null	null	null	s_{10}	Time	s_{15}	Time
null	null	null	null	null	null	s_9	Time	s_{14}	Time
s_8		s_2		s_{23}		s_{19}		s_{10}	
s_{11}		s_{10}		s_{19}		s_{16}		s_{16}	

Fig. 4 An illustrating example of DPHK. (a) The situation of three users walking in the environment; (b) example of CTS, PTS and RTS; (c) passed ME of the situation; (d) process mode of DPHK when cross happens

Figure 4(d) shows how DPHK deals with cross in s_{16}: as long as the MEs used to make prediction are different, the predicted results will be different. s_{16} predicts as ME-No.9 according to ME-No.8 and as ME-No.10 according to ME-No.7. What's more, although two users walk together and have the same speed which means the MEs and received time of MEs are the same, DPHK can process this overlap like only one user walking in the environment and this way doesn't go against our research purpose.

DPHK is divided into DPHK-TT (predicted result with trajectory and time) and DPHK-OT (predicted result with only trajectory) depending on whether the predicted result contain time information.

4 Predict system model based on HMM

This subsection describes the work in solid lines in Fig. 5 and the algorithm is shown in Algorithm 1. PSM is the kernel of DPHK and it is in charge of applying the activity pattern knowledge mined in historical information to predict where a user will go to and when a user will arrive at. PSM of DPHK-TT consists of message passing, prediction of walking direction and prediction of triggering time and PSM of DPHK-OT consists of message passing, prediction of walking direction.

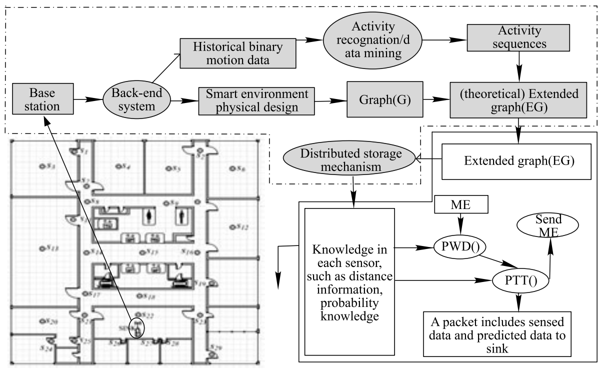

Base station
Back-end system
Historical binary motion data
Activity recognation/data mining
Activity sequences
Smart environment physical design
Graph(G)
(theoretical) Extended graph(EG)
Distributed storage mechanism
Extended graph(EG)
ME
Send ME
Knowledge in each sensor, such as distance information, probability knowledge
PWD()
PTT()
A packet includes sensed data and predicted data to sink

Fig. 5 DPHK system: path prediction of user from sensor network

4.1 System procedure

The operational architecture of proposed DPHK is shown in Fig. 5. Firstly, the base station applies activity recognition and data mining to analyze the historical binary motion data with the information of the physical design of the smart environment and gets the activity sequences which can be used to recognize activity. Secondly, the system gets the network model called extended graph or EG from the sensor deployment and the activity sequences. Next each sensor node stores its own knowledge according to some special rules and this process is also called Distributed Storage. Finally, each sensor node applies PSM to predict in the smart environment.

Algorithm 1 Pseudo code for PSM

```
Input: message ME (ME$_r$) received from other sensor node, access distance linked list of CTS-CTS.ADLL, access direction probability of CTS-CTS.ADP;
Output: message ME (ME$_s$) sent to APs of CTS, predicted results of CTS;
1: if CTS is predicted in ME$_r$ and |the trigger time of CTS-the predicted time of CTS| $\leqslant \theta$//The predicted data of CTS can be accepted then
2:     CTS sends ME$_s$ to its adjacent nodes;
3: else
4:     PWD();//CTS makes new prediction about the next two locations people will go to
5:     PTT()(needed only in DPHK-TT);//CTS computes when people will arrive at the predicted locations
6:     CTS sends a modified command about the wrong prediction, its real sensed data and its predicted results to the sink node;//CTS needs to modify or cancel the wrong predicted data made by its former sensor node
7:     CTS sends ME$_s$;//CTS sends message to its adjacent nodes about the prediction
8: end if
```

What's more, all the procedures in dashed lines is studied by former researchers, we can directly achieve them.

4.2 Message (ME)

ME is used to convey news among sensor nodes. The structure of ME is shown in Table 2, and the number of predicted sensor ID X is determine by Ω. Here are some rules about how to send ME marked as ME$_s$: if the CTS receives a ME from other sensor node marked as ME$_r$:

- The PSID and CSID in ME$_s$ should be changed into the sensor node which sends ME$_r$ to CTS and CTS;
- CTS will not send ME$_s$ to CSID in ME$_r$;
- If CTS is predicted and the predicted result is acceptable, ME$_s$ doesn't include the predicted result of CTS, like ME-No.2 in Fig. 4(c);
- If CTS is predicted but the predicted result is not acceptable, ME$_s$ includes the new predicted results made by CTS and doesn't include the old predicted results, like ME-No.3 in Fig. 4(c).

4.3 Prediction of walking direction (PWD)

This explains the task PWD() in Algorithm 1 and its pseudo code is showed in Algorithm 2. PWD() is used to make prediction of the walking direction of a user. Our strategy is like this: CTS makes prediction according to where a user ever passes. If the PRD of the CTS and the PTS which is a MDS is less than Ω (step 3 in Algorithm 2), CTS will go on predicting, or PTS is a UDS (step 4 in Algorithm 2), CTS only get the next sensor node of PTS in CTS.ADLL as the second location the user will arrive at.

Algorithm 2 Pseudo code for PWD()

Input: message ME (ME_r) received from other sensor node, access distance linked list of CTS-CTS.ADLL, access direction probability of CTS-CTS.ADP;

Output: predicted sensor ID list PSL;

Variables: predicted hop *ph*;

```
1:  AP′=PPWD($ME_r$,CTS.ADP);//prediction of the first location a user
    will arrive at
2:  Put AP′ into PSL;
3:  if $ph < \Omega$ then
4:      if 2-hop activity transition probability of CTS across AP′ exists//
        There are sensor nodes that have 2-hop transmission relation with
        CTS could be predicted
5:          ME=ChangeME();//detailed in 1
6:          AP″=PPWD(ME,CTS.ADP)(prediction of the second location a
            user will arrive at);//prediction of the second location a user will
            arrive at
7:          Put AP″ into PSL;
8:      else
9:          AP″ is the next sensor node of AP′ in CTS.ADLL;//AP′ is a
            UDS
10:     end if
11: end if
12: return PSL;
```

4.3.1 Update ME

This explains the task ChangeME() in Algorithm 2. Because our Ω is two which means our system can make prediction 2-hop away, and our system makes prediction according to the historical information where a user comes from, then when a sensor node predicts the second PTS the user will go to, it should think itself and the first PTS as historical information like it becomes the first PTS and receive a ME from itself. Take s_8 in Fig. 4(a) for example, after it predicts s_{11}, it should update the ME as ME-No.6 and make prediction beforehand instead of s_{11}.

4.3.2 Mixed order HMM modeling

The predict system model is designed as a modified hidden Markov model (HMM) named mixed order HMM or mixed-HMM which is $\lambda = (A, C, \prod)$ and the set of states is S' and it works as PPWD(). PPWD() takes charge of construct the sub set of states and the λ. It also should compute each probability of all the observation sequence, especially, when the CTS is an end point which means people only walk back, it will directly choose the CSID as its PTS.

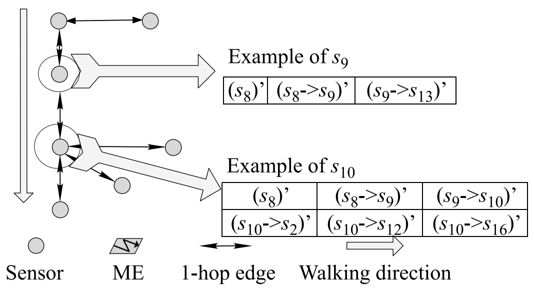

Example of s_9
(s_8)' | $(s_8$->$s_9)$' | $(s_9$->$s_{13})$'
Example of s_{10}
(s_8)' | $(s_8$->$s_9)$' | $(s_9$->$s_{10})$'
$(s_{10}$->$s_2)$' | $(s_{10}$->$s_{12})$' | $(s_{10}$->$s_{16})$'
Sensor
ME
1-hop edge
Walking direction

Fig. 6 Example of choosing state of UDS and MDS

1) **States** This explains the task DecideState() in Algorithm 3. Our Mixed-HMM chooses only the subset of states that contains the nodes active and their neighbor nodes. This reduces the computational complexity without compromising the accuracy (Theorem 1). In the process, if a CTS is not predicted or the predicted result is not acceptable, it got S' as follows:

- If CTS is a MDS, S' is {(PSID)′, (PSID → CSID)′, (CSID → CTS)′, (CTS → every AP of CTS except CSID)′}, like s_{10} in Fig. 6, when s_{10} is to make prediction, it needs more historical information to make a more correct prediction for it has three choices;
- If CTS is a UDS, S' is {(CSID)′, (CSID → CTS)′, (CTS → every AP of CTS except CSID)′}, like s_9 in Fig. 6, for it has only one choice it does not need the states like MDS to increase the computational complexity.

Sub-state selection in Mixed-HMM The sub-state selection in Mixed-HMM does not affect the optimality of HMM model and Forward computation in our application scenario (due to activity transitional relationship among nodes upto 2 hop away).

Theorem 1 In Mixed-HMM the reduced state set results in the same optimal state sequence as that with complete state set [16].

2) **Mixed-HMM model** This explains the task FormHMM() in Algorithm 3. () mainly construct the state transition probability matrix A, emission probability distribution matrix C and initial state distribution $\prod$ of our Mixed-HMM. The three parts are got like followings:

- State transition probability In HMM computation, $A = \{a(j_2, j_1)\}$; when $q(ST-1) = (s_{j_3} \to s_{j_2})'$, $q(ST) = (s_{j_2} \to s_{j_1})'$ and s_{j_1}, s_{j_2} and s_{j_3} are in the same P-R path, then $a(j_2, j_1) = 1$; when $q(ST-1) = (s_{j_3} \to s_{j_2})'$, $q(ST) = (s_{j_2} \to s_{j_1})'$ and s_{j_1}, s_{j_2} and s_{j_3} are in two P-R paths, then $a'(s_{j_3}, s_{j_1})$;
- Emission probability distribution $C = \{c_j(p)\}$. $c_j(p) = P[o_p | q(ST) = (s_j)']$ is the probability that the system outcome at step ST is $o_p \in V$ given the state $q(ST) \in S'$;
- Initial state distribution $\prod=\{\pi_j\}$, where π_j is the probability that at starting step ST_s of HMM time window the state is $(s_j)' \in S'$.

3) **Forward computation** This explains the task Forward() in Algorithm 3 and its pseudo code is given in Algorithm 4. Given the HMM model and the sequence of observations, Forward() computes the probability of the sequence. Take s_{10} in Fig. 6 for example, PPWD() will compute the probabilities of $s_8 \to s_9 \to s_{10} \to s_2$, $s_8 \to s_9 \to s_{10} \to s_{12}$ and $s_8 \to s_9 \to s_{10} \to s_{16}$ using Forward() and choose the maximum one to decide the predicted destination.

Algorithm 3 Pseudo code for PPWD()

```
Input: message ME, access direction probability of CTS-CTS.ADP;
Output: predicted sensor ID, predicted hop $ph$;
1: S′=DecideState(ME,CTS.ADLL);//getting the set of states will be used in current prediction
2: $\lambda$=FormHMM();//getting all the three parts of HMM
3: for each AP marked as AP$_j$ except CSID of CTS ($1 \leqslant j \leqslant n$) do
4:     if AP$_j$ isn't null then
5:         $p_j$ = Forward($\lambda$, S′, sequence[PSID, CSID, CTS, AP$_j$]); //computing the probability of each observation sequence
6:         choose the AP$_j$ which has the maximum $p_j$ marked as AP′′′;
7:     else
8:         let AP′′′ be CSID;
9:         $ph$=2;//considering the situation that CTS is an end point and it is predicting its first PTS
10:        break;
11:    end if
12:    $ph$+=1;
13: end for
14: return AP′′′;
```

4.4 Prediction of trigger time (PTT)

PTT() in Algorithm 5 is introduced in this section. When one sensor node has got its two PTSs, it will compute the speed a user walks based on the receive time of ME, its trigger time and the distance between it and the sensor node which sends ME to it. Then it will search its ADLL to find the distance information of the PTSs. At last, it calculates the trigger time of PTSs and sends the result to the sink node.

Algorithm 4 Pseudo code for Forward()

```
Input: $\lambda$, $S'$, $sequence[a_1, a_2, a_3, a_4]$;
Output: probability of the $sequence[a_1, a_2, a_3, a_4]$;
Variables: $\alpha_t(x)$ is the middle probability of t recursive step;
1: Initialization: $\alpha_1(x) = \pi(x) \times C_x(a_1)$ ($s_x \in S'$);
2: Recursive step: $\alpha_{t+1}(x) = \sum_{x'=1}^{Z} (\alpha_t(x') \times a(x', x)) \times C_x(a_{t+1})$, where ($1 \leqslant t \leqslant 3$, $s_x \in S'$ and $Z$ is the number of states in $S'$);
3: Termination: $p' = \max[\alpha_4(x)]$, where($s_x \in S'$);
4: Output: $p'$ is probability of the $sequence[a_1, a_2, a_3, a_4]$;
```

Algorithm 5 Pseudo code for PTT()

```
Input: message ME (ME$_r$) received from other sensor node, access distance linked list of CTS-CTS.ADLL, predicted sensor ID list;
Output: trigger time of sensor nodes in predicted sensor ID list;
Variables: user speed US
1: CTS searches CTS.ADLL to get the distance-Dis between CSID and it;
2: the time $T$=(the trigger time of CTS-the receive time of ME$_r$);
3: US=Dis/T;
4: for each sensor node marked as PS$_k$ in predicted sensor ID list($1 \leqslant k \leqslant \Omega$) do
5:     CTS searches CTS.ADLL to get the distance Dis$_k$ between PS$_k$ and it;
6:     CTS computes the trigger time of PS$_k$ with the US and Dis$_k$;
7: end for
```

4.5 Real-time applicability of DPHK

There were some constraints to directly using standard HMM model and Forward algorithm to our real-time application scenario. In our framework, the core is the Forward() in HMM, and for the complexity of Forward() is related to the size of state space, we have designed a model with varying size of system state S' that is the set of motion activated nodes and their 1-hop or 2-hop neighbor nodes in EG (explained earlier). Regarding the maximum degree of a sensor node in EG (say MD), the Forward() algorithm requires $O((MD+3)^2)$ operations and other operations like DecideState() and PTT() need no more than O(MD) steps and ChangeME() can be done in constant time.

5 Performance evaluation and experiments

In this section we first introduce the evaluation index of DPHK, and then we will explain our experimental setup in a real smart environment and in a simulation environment, followed by system performance analysis of DPHK.

5.1 Evaluation index

In order to reduce the PRDOS of a sink node, we have proposed DPHK and given an evaluation index of DPHK here. We define the performance of DPHK-POP (proportion of promotion) like this:

$$\mathrm{POP} = (1 - \frac{\text{data received times after using DPHK}}{\text{data received times before using DPHK}}) \times 100\%$$

5.2 System setup

5.2.1 Experiments in real environment

A network of 29 TelosW [17] static wireless sensor nodes (Fig. 1(b)) is deployed throughout the 32 meter × 40 meter floor (Fig. 1(a)) workplace environment (in workplace of Department of Computer Science, ChongQing University). As earlier shown in the physical layout in Fig. 1(a), the sensor nodes are deployed mainly in the hallways, key positions like entry points or exit points and some rooms and they are all powered by the USB patch cord shown in Fig. 1(b). These nodes are fixed on the ceiling and each of them is equipped with Panasonic AMN-31111 passive infrared motion sensor.

Experiment setup: Firstly, we need design some examples that only are used to test and may not have real meanings of an activity. Table 3 shows our test trajectories.

There are some volunteers (users) involving in and every user generates one trajectory. These users finish three scenarios separately in the conditions of DPHK-TT and DPHK-OT: 1) user walks constantly as the user patterns; 2) user walks inconstantly as the user patters, and it means user will change his speed suddenly and walks in the new speed for a while; 3) user walks constantly in random path. In all of the three scenarios, every user carries a device which can record time after being pressed and the users themselves also need to record the sensor ID when they pass a sensor node.

Table 3 List of designed user patterns

No.	Pattern
pattern 1	1-7-8-9-10-12-16-19-18-17-21-20-25-24-21
pattern 2	2-10-12-16-15-14-11-13-17-18-23-22-21-17-14
pattern 3	28-22-23-19-18-17-13-11-14-15-16-12-10-9-8
pattern 4	25-20-21-22-23-18-17-14-15-16-10-9-8-7-4
pattern 5	15-16-10-9-8-11-14-17-18-19-16-15-14-11-8

5.2.2 Experiments in simulation environment

Here we use NS2 to finish our confirmatory experiment about the effect of the number of sensor nodes have on DPHK. And the main simulation parameters are set as Table 4.

Table 4 Simulation parameters values

No.	Area	NoSS	NoS	LoS	Communication radius
#23	(35,35)	100	1	(17,25)	5
#24	(50,50)	100	1	(25,35)	5
#25	(50,50)	300	1	(25,35)	5
#26	(100,100)	300	1	(50,70)	5
#27	(100,100)	500	1	(50,70)	5

Note: NoSS: number of sensor node; NoS: number of sink node; LoS: location of sink node

While we also carefully design some activity patterns without real meanings of each scenario in the simulation experiment to ensure that all the networks of each scenario have similar topology and all sensor nodes are included in the activity patterns. What's more, we set the number of each activity pattern coming out in the experiment before using DPHK and after using DPHK. At last, we set an approximate total number as the closing flag in each experiment, that is to say, when the number that sink node receives data from other sensor nodes reaches the closing flag, the system will quit.

5.3 Results of DPHK experiment

Tables 5 and 6 show the effects of DPHK in single and multiple people environment while Table 7 shows the result of using DPHK in the network with different scales. And we also conduct a special experiment like this: 12 sensor nodes divide into three groups which include 4 sensor nodes separately, and the 4 sensor nodes form a quadrilateral with a 3 meters edge. While we apply DPHK in one group, data merge in another group and do nothing in the last group, then we invite some people just walk constantly in each group clockwise. Additionally, the total distances of the path in each group are the same. The result is shown in Fig. 7.

Table 5 DPHK accuracy results of single user

No.	Scenario	Times	APL	LTA	APRDOS	POP/%
#1	(1)OT	30	15	.995&/	5.02	66.53
#2	(1)OT	30	45	.994&/	15.04	66.58
#3	(2)OT	30	15	.997&/	5.01	66.60
#4	(2)OT	30	45	.992&/	15.07	66.51
#5	(3)OT	30	15	.007&/	14.99	0.07
#6	(3)OT	30	45	.010&/	44.98	0.04
#7	(1)T*T	30	15	.994&.991	5.05	66.33
#8	(1)T*T	30	45	.995&.992	15.27	66.07
#9	(2)T*T	30	15	.995&.575	9.93	33.80
#10	(2)T*T	30	45	.992&.544	31.24	30.58
#11	(3)T*T	30	15	.004&.001	14.99	0.07
#12	(3)T*T	30	45	.007&.001	44.99	0.02

Note: APL: average path length; LTA: location&time accuracy; APRDOS: average PRDOS of sink node

Table 6 DPHK accuracy results of multiple users

No.	Scenario	User num	Times	Length/time	APRDOS	POP/%
#15	(1)T*T	1	30	30	11.04	63.20
#16	(1)T*T	2	30	60	23.37	61.05
#17	(1)T*T	3	30	90	34.87	61.26
#18	(1)T*T	4	30	120	45.29	62.26
#19	(2)T*T	1	30	30	17.36	42.13
#20	(2)T*T	2	30	60	36.74	38.77
#21	(2)T*T	3	30	90	54.83	39.08
#22	(2)T*T	4	30	120	77.26	35.62

Table 7 The results of simulation experiment

No.	Scenario	Data received times (before)	Data received times (after)	POP/%
#23	(1)OT	15 000	5 547	63.02
#24	(1)OT	15 000	5 324	64.51
#25	(1)OT	15 000	5 278	64.81
#26	(1)OT	15 000	5 465	63.57
#27	(1)OT	15 000	5 372	64.19

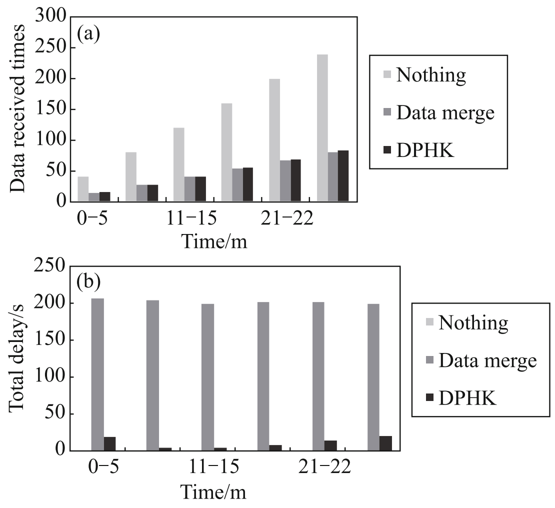
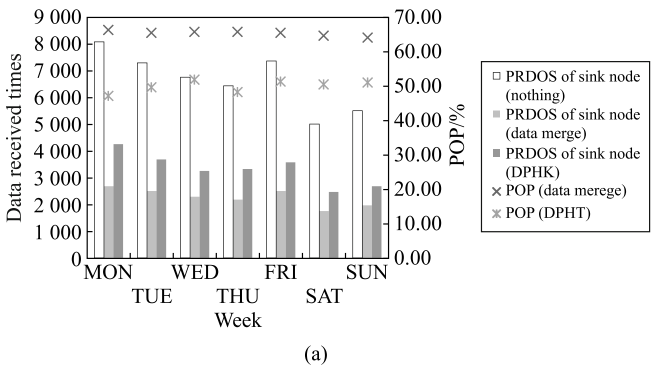


Fig. 7 The comparative results among DPHK, data merge and doing nothing (a) PRDOS of sink node; (b) total delay

What's more, we need design some more examples that can generate ground truth and reflect the typical situation when user walks in a workplace. We first let the system run for a month to learn the knowledge needed for prediction. Next we record the necessary data of three weeks which is generated in the condition of not using the system in the first week, in the condition of using data merge in the second week and in the condition of using the system in the third week. Then we compare the data. Here we should guarantee that all the users will appear in the two weeks only if they have appeared a month ago. Figure 8 shows the comparative result.



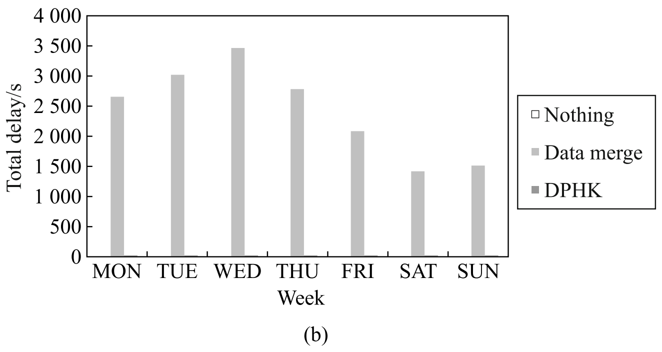


Fig. 8 Comparison of statistical data in three weeks. (a) PRDOS of sink node and POP; (b) total delay

5.4 Performance evaluation

In Table 5:

- From #1 to #6, when taking no account of time of arrival, we could find that scenario (1) and scenario (2) have the same performance and both reduce almost 66% times that sink node receive data from other sensor nodes, and the POP is related to the predicted hop;
- From #7 to #10, we could see that the effect of DPHK decrease substantially for activity without regularity is hard to predict;
- From #5, #6, #11 and #12, the data shows that DPHK almost has no effect for the people walks like a headless chicken, the system really do not know where this user will go to.

In Table 6, as the third scenario really makes no sense, we only get the experiment data of the first two scenarios:

- From #15 to #18, we could find that the number of user has no effect of DPHK;
- Compared #15, #16, #17 and #18 with corresponding #19, #20, #21 and #22, DPHK still work not well in the scenario that people have a irregular speed.

In Table 7:

- Comparing #23 with #24,and #25 with #26, we get that DPHK works well despite the area of testbed;
- Comparing #24 with #25, and #26 with #27, we also find that the number of sensor nodes has no effect of DPHK.

From Figs. 7 and 8, we could get the conclusion that DPHK work well in a daily office environment. What's more, DPHK has almost the same effect with data merge but have a huge advantage on real-time. This is really an important point in some real-time applications.

In one word, whether DPHK could work well is related to whether people act in the environment have regular behaviors, and the performance is determined by the regularity in people's activity and the predicted hop.

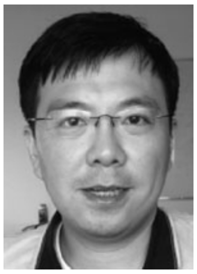
6 Discussion and conclusion

This paper discusses congestion and data loss caused by too many connections to sink node in current Smart Environment and proposes a method to solve the problem by delivering predicted data. Based on DPHK, the data traffic load of sink node to receive data from too many sensor nodes can be significantly reduced. The performance evaluation is demonstrated with results from designed experiments in a smart workplace environment. In the future, our work will focus on: 1) optimizing the DPHK framework for lower complexity and adaptability in different smart environments, 2) improving the DPHK framework with processing overlaps and the situation that when one user reaches a sensor node while this sensor node gets a prediction from a sensor node triggered by another user, 3) perfecting the DPHK framework for dealing with energy consumption in transferring message between sensor nodes and balance the energy among sensor nodes which make prediction, furthermore, designing effective route strategy to send the predicted data.

Acknowledgements The work was supported by the National Natural Science Foundation of China (Grant No. 61672115) and the Fundamental Research Funds for the Central Universities (CDJZR12180006).

References

1. Guo S, Yang Y. A distributed optimal framework for mobile data gathering with concurrent data uploading in wireless sensor networks. In: Proceedings of IEEE INFOCOM. 2012, 1305–1313
2. Seino W, Yoshihisa T, Hara T, Nishio S. A sensor data collection method with a mobile sink for communication traffic reduction by delivering predicted values. In: Proceedings of the 26th IEEE International Conference on Advanced Information Networking and Applications Workshops. 2012, 613–618
3. Kulik L, Tanin E, Umer M. Efficient data collection and selective queries in sensor networks. Lecture Notes in Computer Science, 2008, 4540: 25–44
4. Incel O D, Ghosh A, Krishnamachari B, Chintalapudi K. Fast data collection in tree-based wireless sensor networks. IEEE Transactions on Mobile Computing, 2012, 11(1): 86–99
5. Cheng B, Xu Z, Chen C, Guan X. Spatial correlated data collection in wireless sensor networks with multiple sinks. In: Proceedings of 2011 IEEE Conference on Computer Communications Workshops. 2011, 578–583
6. Wang C, De D, Song W. Trajectory mining from anonymous binary motion sensors in smart environment. Knowledge-Based Systems, 2013, 37: 346–356
7. Hasan M, Rubaiyeat H, Lee Y, Lee S. Mapping of activity recognition as a distributed inference problem in sensor network. In: Proceedings of the 2008 International Conference on Artificial Intelligence. 2008, 280–285
8. Yu G, Yuan J, Liu Z. Predicting human activities using spatiotemporal structure of interest points. In: Proceedings of the 20th ACM International Conference on Multimedi. 2012, 1049–1052
9. Tastan B, Sukthankar G. Leveraging human behavior models to predict paths in indoor environments. Pervasive and Mobile Computing, 2011, 7(3): 319–330
10. Li W, Zhang X, Yang Y, Cai S, Luo Q. Efficient data fusion for wireless sensor networks. In: Proceedings of 2011 IEEE International Workshop on Open-Source Software for Scientific Computation. 2011, 43–46
11. Tsitsipis D, Dima S, Kritikakou A, Panagiotou C, Koubias S. Data merge: a data aggregation technique for wireless sensor networks. In: Proceedings of IEEE International Conference on Emerging Technologies and Factory Automation. 2011, 1–4
12. Silberstein A, Braynard R, Yang J. Constraint chaining: on energy-efficient continuous monitoring in sensor networks. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. 2006, 157–168
13. Marin-Perianu M, Lombriser C, Amft O, Havinga P, Troster G. Distributed activity recognition with fuzzy-enabled wireless sensor networks. Lecture Notes in Computer Science, 2008, 5067: 296–313
14. Amft O, Lombriser C, Stiefmeier T, Troster G. Recognition of user activity sequences using distributed event detection. Lecture Notes in Computer Science. 2007, 4793: 124–141
15. Koodziej J, Xhafa F. Utilization of markov model and nonparametric belief propagation for activity-based indoor mobility prediction in wireless networks. In: Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems. 2011, 513–518
16. De D, Song W, Xu M, Wang C, Cook D, Huo X. Findinghumo: real-time tracking of motion trajectories from anonymous binary sensing in smart environments. In: Proceedings of International Conference on Distributed Computing Systems. 2012, 163–172
17. Lu G, De D, Xu M, SongW, Cao J. Telosw: enabling ultra-low power wake-on sensor network. In: Proceedings of the 7th International Conference on Networked Sensing Systems. 2010, 211–218



Chengliang Wang is a doctor, professor and PhD visiting scholar to Georgia Institute of Technology, USA and senior member of China Computer Science Association, China and member of America Association of Computing Machinery, USA. His research interests include smart environemt theory and application based on Internet of things; the research and application of video monitoring based on image processing; research and development of intelligent computing on the complex system.

Yayun Peng is currently a master student of Department of Computer Science, Chongqing University, China. His current research interests are in the areas of smart environments, data mining, wireless sensor networks.

Debraj De is currently a postdoctoral research associate in Department of Computer Science, Missouri University of Science and Technology, USA. His current research interests are in the areas of smart environments, smart healthcare, machine learning, wearables and sensor networks.

Wen-Zhan Song mainly focuses on cyber–physical systems and computing for geophysical imaging, smart grid and smart health, where decentralized sensing, computing, communication and security play a critical role and need a transformative study. He is a recipient of NSF CAREER Award (2010), Outstanding Research Contribution Award (2012) by GSU Computer Science, Chancellor Research Excellence Award (2010) by WSU Vancouver.