QoI-Aware Data Collection for Mobile Users in Wireless Sensor Networks

Gang XU
Abstract

QoI-Aware Data Collection for Mobile Users in Wireless Sensor Networks

Gang XU

Ubiquitous data collection enables mobile users to collect data from the surrounding wireless sensors along their walks. However, the limited contact time and the wireless capacity constrain the amount of data that can be collected by the mobile users. Quality of Service (QoS) becomes very important for mobile users to collect sensing data that can maximize their information value. To the best of our knowledge, we are the first to propose a distributed algorithm that can support QoS ubiquitous data collection for multiple mobile users. Our distributed algorithm constructs the data collection trees adaptively to the dynamic moving speeds and the available capacity of the mobile users. It allocates capacity for receiving high priority data to maximize the information value with low communication overheads. Our algorithm supports smooth data collection for multiple mobile users with independent movements. We provide analysis and extensive simulations to evaluate the information value, energy efficiency and scalability of our distributed solution. The results showed that our distributed algorithm can improve information value up to 50% and reduce energy consumption to half compared with the existing approach. Our algorithm also scales perfectly well with increasing number of mobile users and moving speeds.
Acknowledgements

I would like give my deepest appreciations to Dr. Edith Ngai and Dr. Per Gunningberg, they have broaden me the view of WSN area as well as given their full supports on my proposals. They guide me always on the track and focus on what I need to do. Their hard-working and academically rigorous taught me how to overcome all the academical difficulties. Special thanks to Dr. Edith Ngai, who my supervisor, always patiently repeatedly discuss with me even I made a lot of stupid mistakes. She also helped me a lot on the algorithm analysis and writing in this thesis. I cannot accomplish my thesis without her marvelous supports.

This thesis has also had very massive helps from Laura. Because your special insight of OMNeT++ make my work possible.

Thanks to all the colleges and friends, my research life became interesting and colorful. Special thanks to excellent scholars: Fredrik, Frederik, Martin, Liam and all other members in CoRe group, your great encouragement on my exploring WSN as well as my thesis are with me all the time. Especially Maria and Dena, the talks with you were so nice, and without Maria’s enlightenment, I could not start my coding.

My fiancee Jing Liu is always with me even in the late mid night. The one year life will become miserable if out of her kindness and tolerant.

Finally, this thesis is dedicated to my most beloved parents.
1. Introduction

Wireless sensors networks (WSNs) have been widely deployed for environmental monitoring to build a sustainable society. They enable better understanding on the climate change, pollution, habitats from natural environment to water system, road traffic, and energy consumption in urban lives [1, 2, 3]. With the advancement of mobile devices, ubiquitous data collection is a set of mechanism to allows mobile users to collect data from their nearby sensors using their mobile devices, such as PDAs or mobile phones [4, 5, 6]. This architecture increases the flexibility of sensor network deployment and offers a cost-effective solution for sensor data collection. Mobile users can support data collection in sensing applications such as environmental monitoring, social network, healthcare, transportation, safety, etc. For example, PEIR (Personal Environmental Impact Report) is a mobile sensing application developed to calculate personalized estimates of environmental impact and exposure [7]. Mobile phones, mobile users, can collect sensing data of more variety by communicating with wireless sensors in their surroundings. For instance, in the GreenOrbs project [8], forest rangers can use their PDAs to collect data with the wireless sensors deployed in the forest to collect scientific data, such as temperature, humidity, concentration of carbon dioxide, etc.

1.1 Ubiquitous Data Collection

Different from traditional WSNs, ubiquitous sensor data collection does not rely on a stationary sink to collect sensing data from the whole network. Instead, mobile users collect sensing data from their surrounding sensors pervasively using their mobile devices. The uncontrollable mobility of the mobile users and the limited wireless communication range pose new challenges in ubiquitous sensor data collection. In particular, the contact time between the mobiles and the sensors can be very short given the continuous and potentially fast movement of mobile users. Quality of Service (QoS) becomes very important to maximize the information gain out of the limited contact time and wireless capacity. We define information utility to measure the total value of information obtained by the mobile users from collected sensing data. In general, sensing data of unusual events are considered more important and valuable to the users compared with routine data, so that they are weighted
with higher priority for collection. Intuitively, the users will obtain higher information utility if they collect more important (or highly weighted) sensing data. QoS in ubiquitous data collection is very challenging as it has to handle the dynamic topology and capacity of the data collection due to the mobility of mobile users. Energy efficiency is another major concern for both the mobile devices and the wireless sensors. The mobile users may want to save batteries of their phones for making phone calls or running other mobile applications. The battery lifetime is even more critical for wireless sensors as they are usually non-rechargeable after deployment.

Although data collection in WSNs with mobile elements have been studied, the previous work tackled the problem mostly in a controllable environment. Most of them assume that the mobile elements move along pre-defined paths and stop at rendezvous points to collect data. However, the mobility of mobile users are individual and uncontrollable in real life. Mobile users may move continuously without any stops, which implies that data collection has to be done along their walks. The data collection trees need to be constructed and migrated quickly and adaptively to the mobility of users. Global data collection trees have been widely adopted by the mobile elements (MEs) in existing work [4, 9, 10, 11]. The idea is to broadcast HELLO messages to the nodes in the network for tree construction. However, it is not easy to determine how far the messages should be broadcasted. A simple way to limit the size of the data collection tree is by predefining a maximum hopcount $h$ that the HELLO message is broadcast. However, this simple solution does not consider the dynamic moving speed of the mobile user, so the predefined maximum hopcount may be over-estimated or under-estimated. To handle this problem, we propose a distributed approach to construct data collection tree and dynamically estimates its capacity according to the moving speed of the mobile user.

1.2 OMNET++ and MiXim

OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators [12]. It is open-source software with supports to networks including on-chip network, wireless communication network, queuing network and so on. A user-friendly Integrated Development Environment (IDE) based on Eclipse also has been provided for easily and rapidly developing simulation cases. In addition to IDE features, OMNeT++ provides a graphic user interface (GUI) based on TCL to investigate the on-going network simulations. The GUI consists individual nodes, gate and links. The developer can easily access a node and look into different layers in order to find out packets behaviours, memory leak, layer cooperation, etc. Furthermore, OMNeT++ has a very wide which community support continuously providing updates and new frameworks, e.g. MiXim for wireless sensor network, INET Framework for Internet protocols, etc.
Among several wireless sensor network and mobile network simulation libraries, MiXim is one of the best and long-term maintained framework. It offers detailed models of radio wave propagation, interference estimation, radio transceiver power consumption and wireless MAC protocols (e.g. 802.15.4). In the meantime, it also supports different mobility models including random-way mobility, mass mobility, liner mobility and so on, which very well simulates human movements. With the help of MiXim, developer can simply focus on proposed algorithm without concerning other layers’ protocols. More importantly, MiXim is actually the only framework that fully supports non-beacon based IEEE 802.15.4 protocol in the multi-hops WSN collection tree networks. The comfortable initial training, the modularity, the possibility of programming in an object-oriented language (C++), are among the reasons that led us to prefer the OMNeT++ platform, and MiXim, over other available network simulators like the well-known ns2 and SensorSim [13].

1.3 EQRoute
The contributions of this work are as follows. First, we propose a distributed algorithm, called EQRoute, to provide QoS ubiquitous data collection using mobile users. To the best of our knowledge, we are the first to propose a distributed QoS data collection solution for multiple mobile users with uncontrollable, unpredictable and non-stop mobility. Second, we estimate the available capacity (i.e. the amount of data a mobile user can collect in a certain period of time dynamically according to their moving speeds) and provide energy-efficient and smooth data transmission. Third, the proposed distributed algorithm can support multiple mobile users to construct balanced data collection trees to maximize the information utility. Finally, we analyse the optimality of EQRoute and evaluate its performances by extensive simulations to examine the information utility, energy consumption, communication overhead, and scalability with multiple mobile users and variable speeds. Compared with the most advanced existing approach, EQRoute greatly improves the information utility and reduce half of the energy consumption. The results also validate its scalability with multiple mobile users and high moving speeds.

The rest of this thesis is organized as follows. The related work is presented in chapter 2. The problem define, challenges described, and problem formulation with discussion on a centralized approach are in chapter 3. We present our distributed ubiquitous data collection algorithm in chapter 4. We demonstrate our QoS and energy efficiency ubiquitous data collection protocol design in chapter 5. Finally, We provide analysis and extensive simulations in chapter 6 and 7 and conclude this thesis in chapter 8.
2. Related Work

2.1 Collection Tree Protocol

Data collection is the primitive function in wireless sensor network, a typical collection protocol provides for the construction and maintenance or one or more routing tree having each so-called sink as their root[14]. The sink can therefore upload the data to the internet or other database for further analysing. It is essential to meet the following requirements for a data collection protocol: a) it can estimate the link quality, b) it must have some mechanism to detect and remove routing loops, c) it should have ability to detect packets loss and duplicate packets.

The Collection Tree Protocol (CTP) is a link detection based data collection protocol. It provides several mechanism to route the packets from sensor nodes to (one or more) sink nodes with best effort, which means, it implements a set of strategies to increase the data delivery ratio[2009collection]. To achieve reliability, robustness, efficiency, and hardware independence in CTP, standard implementation has been proposed by [16] and evaluated by [17][15]. It consists of three major subcomponents: link estimator, route engine, and forward engine. [15] further indicated out the link dynamics and transient loops are the essential reasons cause only 2-68% of delivery ratio.

2.2 QoS and Energy-Efficiency Aggregation Tree with Mobile Sinks

Mobile sinks and mobile relays have been suggested for improving the performance of data collection in wireless sensor networks [18, 19]. Shah et al. [10] presented an architecture using moving entities, called data mules, to collect sensing data. Gu et al. [20] proposed a partitioning-based algorithm to schedule the movement of mobile elements, which minimizes the required moving speed and eliminates buffer overflow. Bisniket et al. [21] studied the problem of providing quality coverage using mobile sensors and analysed the effect of controlled mobility on the fraction of events captured. Xu et al. [22] further studied delay tolerant event collection in sensor networks with mobile sinks which considers the spatial-temporal correlation of events in the sensing field. He et al. [23] analysed the performance of data collection theoretically to evaluate services disciplines of mobile elements through a queueing model. Nevertheless, the above works focus on controlling the movement of mobile
sinks for data collection, which is different from the mobile phone users with independent and uncontrollable mobility in our work.

Some studies have been conducted focusing on mobile elements without any fixed trajectory. Kusy et al. [24] presented an algorithm to predict the mobility pattern of the mobile sinks from the training data. They computed and maintained the mobility graph of the mobile sinks to improve routing reliability in data collection. Similarly, Lee et al. [25] presented a routing scheme that exploits the mobility pattern of the mobile sinks to minimize energy consumption and network congestion. The above works mainly focus on predicting the movement of mobile elements to improve routing efficiency. Recently, ubiquitous data collection with mobile users has been studied for mobile users to collect data from wireless sensors networks. Li et al. [4] proposed a ubiquitous data collection scheme that can efficiently form a new data collection tree by locally modifying the previously constructed data collection tree. However, no work has been done for maximizing the information utility in ubiquitous data collection. In addition, collaboration among multiple mobile users for ubiquitous data collection, e.g. balance the data, adapt different user’s location and speed, etc. has not been fully explored.

QoS and energy-efficient data collection and routing have been widely studied for wireless sensor networks. Akkay et al. [26] presented an energy-aware QoS routing protocol for sensor networks that finds QoS paths for real-time data with certain end-to-end delay requirements. SPEED uses information about a node’s one-hop neighbourhood and geographic forwarding to find paths, while enforcing a uniform delivery velocity to bound the end-to-end packet delay [27]. Felemban et al. [28] further proposed another packet delivery mechanism called Multi-Path and Multi-SPEED Routing Protocol (MM-SPEED) for probabilistic QoS guarantee in wireless sensor networks. Gelenbe et al. [29] proposed an adaptive routing protocol that can detect the presence of novel events in a distributed manner hereby provides better delay and better quality of service (QoS) for the high priority traffic. Recently, Bai et al. [30] proposed Delay-bounded Energy-constrained Adaptive Routing (DEAR) that jointly studies the adaptive multipath routing, differential delay, and energy consumption problems in WSNs. However, the above works only consider wireless sensor networks with static settings, which do not address data collection by mobile users with dynamic movements.

2.3 Energy Consumption Model

Whilst some researches [31][32] studied rechargeable energy consumption in WSN, as mentioned in Chapter 1, the energy is still the key to sensors, for instance a 1 cm$^3$ battery can provide approximately up to 2KJouls energy, which means that a node has to dissipate an average power only up to 63.4 µW in order to achieve a 1 year lifetime of continuous operation [33].
energy consumption model normally divided into two parts, one is commu-
nication regarding to transmission consumption, the other is electronic cir-
cuit consumption. Most of the current research focus on the first part based
on assumption the second part is constant. [34] proposed the estimation of
communication subsystem as $E_T(d) = E_{T\_elec} + \varepsilon_{fs} \times d^\alpha$ and $E_R = E_{R\_elec}$. Where the path loss exponent $\alpha$ includes 2 and 4, energy used by transmit,
receive, and chip are defined as $E_{T\_elec} = E_{R\_elec} = E_{elec} = 50\text{nJ/\text{bit}}$, and
$\varepsilon_{fs} = 10\text{pJ/\text{bit}/m}^2$. [35] further investigated the two typical reliable trans-
mision modes: End-to-End Retransmission (EER) and Hop-to-Hop Retrans-
mission (HHR) to propose a Total Transceiving Power Model. Based on [35],
[36] provided an accurate energy consumption model for the both modes with
three common wireless MAC protocols: CSMA, MACA and 802.11. Later
on, [37] indicated that Power Amplifier (PA) also plays very important role in
energy consumption: $P_{r}(d) = P_{r0} + \frac{P_{Rx} \times A \times d^\alpha}{\eta}, P_R = P_{R0},$ where $P_{Rx} \times A \times d^\alpha$
is the power delivered to the antenna by the PA of the transmission node, in
which $P_{Rx}$ is the power received by the antenna of receiving node and deliv-
ered to the low noise amplifier (LNA); A is determined by characteristics of
the transmitting and receiving antennas; path loss exponent is given by $\alpha$. The
parameter $\eta$ is the drain efficiency of PA. $P_{T0}$ and $P_{R0}$ are constant, and denote
the power consumption in the circuit.

In this thesis, to simplify our scenario by presenting core feature of EQRoute,
we adapt the simplest energy consumption model $E_{Tx} = E_{Rx} = e_d \times b \times d^\alpha$
[38], where $b$ is the transmitted data site in unit of bit, $e_d$ is the energy dis-
ipated per bit per $m^2$, $d$ is the distance from transmitter to receiver and $\alpha$ is
a constant no smaller than 2. This model presents the energy consumption
a positive relation to the distance, which we further simplify this model by
assuming the average distance between any two nodes is $d$, then the energy
model relates to the hop counts. Therefore, we use hop counts to calculate the
energy consumption.
3. QoS and Energy Efficiency Ubiquitous Data Collection

We first consider a motivational toy example shown in Figure 3.1, where a mobile user is walking in a sensing field to collect data such as temperature, soil moisture, and light intensity, etc. The wireless sensors take sensor measurements and store the data in their buffers. The data can be picked up by the mobile users when they pass the sensors. We define information value as the importance of various observations carried by the sensing data [39]. The information value of a data packet, $w$, can be measured by its temporal variation to the normal range of the environmental parameter being sensed. For instance, the information value is high if the data carry important information of unusual events, such as abnormal temperature, high pollen level, high concentration of $CO_2$, etc. We consider the information value obtained by the mobile users to simply be the sum of the information value collected from each individual data packets. We do not consider any correlations between different data packets. Intuitively, the mobile users want to maximize the information value of the collected data from the wireless sensors. For simplicity, we normalized the information value of the data packets, i.e. $0 \leq w \leq 1$. In this example, the mobile user may want to collect data with $w = 0.8$ rather than those with $w = 0.1$ (see Figure 3.1) if he does not have enough time and capacity to collect all the data.

*Figure 3.1.* Mobile user walking in a sensing field to collect data using his mobile phone. The numbers in the figure denote the information value $w$ of the buffered sensing data. The data with higher information value are given higher priority for collection.
3.1 Challenges and Design Goals

It is challenging to collect data from the wireless sensors when the mobile user is moving with uncontrollable and unpredictable mobility. This is because communication opportunities occur only when the mobile user approaches the sensors. Due to the limited wireless communication range, i.e. IEEE 802.15.4 or bluetooth, the mobile user may have a short contact time to collect sensing data, especially when he is moving fast and without any stops. Since the sensors and the mobile device communicate over the same wireless channel, they have to share the limited wireless capacity with their neighbouring nodes. Bottleneck may occur particularly at the mobile node, since it is the root of the data collection tree that receives and processes maximum amount of traffic. Due to the above reasons, quality of service (QoS) is very important for the users to maximize the total information value from the collected data.

We highlight the design goals of our QoS ubiquitous data collection scheme here:

- It provides quality of service in ubiquitous data collection to maximize the information value of the collected data by giving priority to collect important data with high $w$.
- It achieves energy efficiency for the sensor nodes by reducing their communication costs (i.e. hopcounts) in forwarding data to the mobile user.
- It supports smooth data transmission adaptively support different moving speed of the mobile user.
- It should be scalable to support multiple mobile users in a sensing field.

In our example, the sensor nodes generate sensing data periodically and cache them in their buffers. The mobile devices collect data from their surrounding sensors. The collected data is uploaded by the mobile devices to the server when a Internet connection becomes available.

We focus on data collection from the wireless sensors to the mobile users. Our goal is to maximize the information value and reduce the communication overhead in ubiquitous data collection. The mobile user is similar to a mobile element (ME) with uncontrollable mobility, limited communication range, and variable moving speed. Each sensor can communicate with the MEs and sensors that are within its wireless communication range. Multihop routing is supported to deliver data from the wireless sensors to the mobile users.

3.2 Problem Formulation

We introduce the following notations:

- Each mobile user $j$ creates a routing tree $T_j$.
- Each data packet $d_i$ has an information value $w_i$. It is equal to $w_H$ for high priority data and $w_L$ for low priority data.
- We denote $\mu_j$ as the maximum service rate that $j$ can receive and process data from $T_j$. 
• We define capacity $C_j$ as the amount of data (in number of packets) that can be collected by $j$ in a timeslot $t$, where $C_j = \mu_j t$. Note that this capacity is shared among the neighboring nodes of $j$ and their subtrees.

• We use hopcount $c_{ij}$ to measure the communication cost for delivering $d_i$ to $j$.

Variable:
$x_{ij} \in [0, 1]$: indicates whether packet $d_i$ is sent using tree $T_j$.

Objective:

$$\max U = \sum_{i,j} \frac{w_i}{c_{ij}} x_{ij}$$  \hspace{1cm} (3.1)

Constraints:

$$\sum_j x_{ij} \leq 1, \forall i$$  \hspace{1cm} (3.2)

$$0 \leq x_{ij} \leq 1, \forall i$$  \hspace{1cm} (3.3)

$$\sum_i x_{ij} \leq C_j, \forall j$$  \hspace{1cm} (3.4)

We measure the information gain per communication cost of a data packet $d_i$ by $u_i = w_i / c_{ij}$, which is the information value of the data divided by its communication cost in hopcount. Our objective function is to maximize the sum of $u_i$ from all the collected data, denoted by $U$. This allows us to maximize the information value from the collected data, while achieving a good balance between the information value and the communication cost. Constraint 3.2 ensures that each data packet $d_i$ is sent to only one mobile user. Constraint 3.3 allows fractional data to be sent in a packet. Constraint 3.4 ensures that the total packets received by mobile user $j$ does not exceed its capacity $C_j$ in a given timeslot.

3.3 Optimal Algorithm

We suggest a centralized data collection algorithm for mobile user to maximize $U$ from the collected data (see Algorithm 1). The mobile user first floods the network to gather the information value, size and communication cost of the data from the sensors. Then, it assigns capacity to the sensors by selecting data with the maximum information value per communication cost, i.e. $w_i / c_{ij}$. However, this algorithm works only in a centralized manner. The mobile user has to wait for the information from all the sensors before allocating the capacity to individuals. As discussed before, the mobile user has to flood the whole networks or broadcast to $h$ hops, but $h$ is not easy to decide.
Algorithm 1 Centralized Capacity Allocation

1: $C_j$: capacity of mobile $j$;
2: $w_i$: information value of data $d_i$;
3: $c_{ij}$: hopcount for $d_i$ to reach mobile $j$;
4: Mobile user $j$ broadcasts to all sensors in the network;
5: Each sensor replies to $j$ with $w_i$ and $c_{ij}$ of its data;
6: while $C_j > 0$ do
7: Choose the data $d_i$ with maximum $w_i/c_{ij}$;
8: if $C_j \geq 1$ then
9: $x_{ij} = 1$;
10: else
11: $x_{ij} = C_j$;
12: end if
13: $C_j = C_j - x_{ij}$;
14: end while

Theorem 1. The above greedy algorithm gives an optimal solution for capacity allocation to sensors.

Proof. The algorithm allocates the capacity $C_j$ for receiving the sensing data. We assume that the capacity is limited, so that it is not enough to collect all the data in the network. This implies that there exists a $q$, such that $1 = x_1 = \ldots = x_{q-1} > x_q \geq x_{q+1} = \ldots = 0$, where $x_{n+1} = 0$. We show the optimality of this solution by comparing to any other feasible solution $y_1, \ldots, y_n$ of this problem. Since $w_i/c_{ij}$ are positive for all $i$, this solution can only be optimal if $\sum_i y_i = C_j$.

Let $k$ be the smallest index such that $y_k < 1$, and let $l$ be the smallest index with $k < l$ such that $y_l > 0$. Note that such an $l$ exists, unless the solution $y_1, \ldots, y_n$ is equal to the solution $x_1, \ldots, x_n$ obtained by the above greedy algorithm. We will now increase $y_k$ and decrease $y_l$, while keeping all other values equal, to obtain a new solution. Let $\epsilon = \min\{1 - y_k, y_l\} > 0$. Increase $y_k$ by $\epsilon$ and decrease $y_l$ by $\epsilon$. It is easy to find that this move yields a feasible solution with value not smaller than the value of the solution $y_1, \ldots, y_n$. Moreover, either $y_k$ has become equal to 1, or $y_l$ has become equal to 0. Repetition of this argument eventually yields the solution $x_1, \ldots, x_n$ obtained by the greedy algorithm. □

However, the energy consumption is very high in this centralized approach. The mobile user has to broadcast to all the nodes in the network and get back their replies. The communication overhead is in the order of $O(N)$, where $N$ is the number of nodes in the network. Plus there is a delay before sending can take place.
4. Distributed Algorithm

We propose a distributed and energy-efficient QoS ubiquitous data collection algorithm, called EQRoute, in this section. The main idea of our approach is to utilize the estimated available capacity $C_j$ and sensor demands to automatically determine the maximum layer for constructing the data collection tree. This distributed design also supports collaborative data collection with multiple mobile users. We present our design with three components: a) Data collection tree construction, b) Data collection tree migration, and c) Multiple mobile users in the followings.

4.1 Data Collection Tree Construction

We consider that each sensor holds high priority and low priority data with probabilities $p_H$ and $p_L$, where $p_H + p_L = 1$. Their information values are $w_H$ and $w_L$ respectively, where $w_H > w_L$. Similar to most of the studies, a data collection tree is constructed with a HELLO message from the mobile user. However, unlike existing approaches, we do not flood the whole network or broadcast to a predefined hopcount. Instead, each node decides whether to extend the tree for the next layer by checking its remaining capacity in a distributed manner.

In our algorithm, the capacity of mobile $j$, $C_j$, has to be updated according to its moving speed for energy-efficient and smooth data transmission. It is much easier if we know the coordinates of each sensor and the trajectory of the mobile user. However, we do not make these assumptions, since we want to give more freedom and flexibility for the mobile users to go anywhere. To handle the unpredictable mobility, we introduce $\Delta D$ for the mobile user to estimate its valid moving distance from setting up connection with a new sensor until disconnection. We pick $\Delta D$ as the communication range $R$ in this work to ensure that the tree is updated before mobile user losing connection with its neighbouring nodes. Then, we estimate the available capacity of the mobile user in the time interval $\Delta T = \frac{\Delta D}{v_j}$ as follows

$$C_j = \frac{\mu_j \Delta D}{v_j},$$

(4.1)

where $\mu_j$ is the service rate and $v_j$ is the the moving speed of $j$.

The mobile user $j$ estimates its available capacity according to its moving speed from time to time. It starts the data collection process by running the
Algorithm 2 Distributed Capacity Allocation

1: \( d^H_i \): required capacity for high priority data from node \( i \);
2: \( d^L_i \): required capacity for low priority data from node \( i \);
3: \( f_j \): free capacity of node \( j \); initially \( f_j = C_j \);
4: 
5: **Procedure CapacityAllocation**\((j, f_j)\)
6: Broadcasts Hello message to 1-hop neighbours;
7: Each neighbouring node \( i \) replies with \( d^H_i \) and \( d^L_i \);
   { //Allocate capacity for \( H \) data}
8: for each reply from neighbouring nodes \( i \) do
9:   if \( f_j + d^L_i > 0 \) then
10:      Assign capacity to \( d^H_i \);
11:      \( f_j = f_j - \min(f_j + d^L_i, d^H_i) \);
12:   end if
13: end for
   { //Allocate capacity for \( L \) data}
14: for each \( d^L_i \) do
15:   if \( f_j > 0 \) then
16:      Assign capacity to each \( i \) with \( d^L_i \);
17:      \( f_j = f_j - \min(f_j, d^L_i) \);
18:   end if
19: end for
20: Assign remaining capacity to each \( i \) with \( f_i = f_j/N(j) \);
   { //Extending the tree}
21: for each neighbouring nodes \( i \) do
22:   if \( f_i > 0 \) then
23:      Run **CapacityAllocate**\((i, f_i)\);
24:   end if
25: end for
End Procedure
CapacityAllocation($j, C_j$) algorithm (see Algorithm 2). It first broadcasts to its one-hop neighbours to obtain their capacity requests including the size of high and low priority data, $d^H_i$ and $d^L_i$, respectively. Then, it assigns capacity first to the high priority data according to the received requests. The neighbouring nodes start transmitting the high priority data immediately after the capacity is allocated. Afterwards, the mobile user assigns the remaining capacity $f_i$ to the low priority data of its neighbouring nodes. If there is still remaining capacity, the mobile user will assign the capacity evenly to its $N(j)$ neighbours, i.e. $f_i = f_j/N(j)$. The neighbouring nodes will extend the tree $T_j$ to the next layer by running the CapacityAllocation procedure. Similar to $j$, each node $i$ broadcasts to its one-hop neighbours $m$ to receive replies of $d^H_m$ and $d^L_m$. The capacity allocation process is repeated until there is no remaining capacity left in the data collection tree. Note that the high priority data can preempt the low priority data in the father nodes if there is not enough capacity in the data collection tree.

4.2 Data Tree Migration

As discussed before, the mobile user estimates the new capacity of the data collection tree every $\Delta T$. However, unexpected disconnections may still occur between the mobile user and the sensors due to its changing speed and moving direction. Hence, the mobile user broadcasts a “MobileHere” maintenance message periodically to its neighbouring sensors to notify its existence. In general, the sensors wait passively for the maintenance message. However, they can also check actively for the existence of the mobile user if they do not receive any maintenance messages.

In addition, the MobileHere message can be used for updating the tree structure according to the new location of the mobile user. For example, node $i$ may observe that the mobile user is very close if it can receive the maintenance message directly from the mobile user. Node $i$ can then connect directly to mobile user rather than taking a longer path through a relay node. This scenario can be handled formally by a tree migration process. For better illustration, we divide these migration processes into two types, namely inner-tree migration and tree recovery.

Figure 4.1 demonstrates an example of inner-tree migration. At the beginning, the mobile user is connected to only root node $A$ in the data collection tree. Then, it moves to a new location, where it can communicate directly with some other sensors. When sensor node $B$ receives the tree maintenance message from the mobile user, it knows that the mobile user is nearby. Then, node $B$ becomes the root of its subtree and connects directly to the mobile user. After updating the route, node $B$ notifies its previous relay node $A$ and the mobile user to update the capacity assignment accordingly.
Figure 4.1. Inner-tree migration process

Figure 4.2 shows the tree recovery process. Once the root node $A$ detects a disconnection with the mobile, it sends out a “FindMobile” tree recovery message to its neighbours and tries to recover the connection. Any nodes that do not belong to the subtree of $A$ can help relaying the message to the mobile user. In this example, Node $B$ relays the messages for node $A$, so that $A$ is reconnected to the mobile user. Similar to inner-tree migration, it is necessary to update the capacity accordingly after tree recovery. Otherwise, the mobile user may think that $A$ has finished transmitting, while $A$ is still in the tree. If the tree recovery process fails, node $A$ may join the data collection trees of other mobile users if they have free capacity. To avoid routing loop, we include the root ID of the subtree that is connecting directly to the mobile user in the tree recovery message. Only the nodes with different subtree IDs will response to the tree recovery request.

4.3 Multiple Mobile Users

In the scenario with multiple mobile users, the sensor node has to choose one of the mobile users to report its data. It can compare the performance of different mobile users to select the best route. We suggest a metric, $\rho_j$, to evaluate the performance of each mobile user $j$ considering its available capacity and communication cost as follows.

$$\rho_j = \frac{\mu_{ij} \Delta t_j}{c_{ij}}$$

(4.2)

where $\mu_{ij}$ and $c_{ij}$ are the available service rate and the hopcount for the mobile user to receive data $d_i$, and $\Delta t_j$ is the available transmission timespan. Initially,
(a) A sends tree recovery request

(b) B permits the joining of A

(c) B notifies the mobile user to update the capacity

(d) A updates capacity information to its children

Figure 4.2. Tree recovery process
\( \rho_j \) is set to -1, so that node \( i \) shows interest to any HELLO messages. Node \( i \) computes the performance metric \( \rho_j \) for each newly arrived HELLO message. It changes to a new route only if the new metric \( \rho_j \) is greater than the existing one (see Algorithm 3).

We observe that frequent switching between routes cause energy consumption and reduce the packet delivery rate. To avoid this, node \( i \) will change its route only if the new \( \rho_j \) is better than the current \( \rho_j^* \) with a certain threshold, i.e. \( \frac{\rho_j}{\rho_j^*}>\gamma \), where \( \gamma \) is greater than 1. When the sensor changes its route, it will notify its parent node to request for new capacity. However, it will not stop sending its data through the old route until the new capacity is assigned.

**Algorithm 3** Evaluation of Alternative Routes

1: \( \mu_{ij} \): service rate of \( j \) for receiving data \( d_i \);
2: \( c_{ij} \): hopcount for reporting data \( d_i \) to \( j \);
3: \( \Delta t_j \): timespan can be used for transmitting data to \( j \);
4: 
5: **Procedure SensorEvaluation** \((\mu_{ij}, c_{ij}, \Delta t_j)\)
6: \( \rho_j = \frac{\mu_{ij} \Delta t_j}{c_{ij}} \);
7: **if** \( \frac{\rho_j}{\rho_j^*}>\gamma \) **then**
8: Update the route to report data to \( j \);
9: Send request message for new capacity;
10: \( \rho_j^* = \rho_j \);
11: **end if**
12: **End Procedure**
5. Protocol Design

As a routing protocol, EQRRoute handles control messages from MAC layer, e.g. ack, bit-error, queue-full and so on, and also handles data and control message from application layer. It encapsulates the upper layer packets into network packets to save routing overheads as well as performance measurement overheads. The overview of EQRRoute protocol including different modules is shown in figure 5.1. we will further elaborate EQRRoute in a protocol view in this chapter.

![Figure 5.1. Overview of EQRRoute Protocol Design](image)

There are three essential issues to integrate above algorithms into our EQRRoute protocol: a) nodes’ roles and states in aggregation tree for maintaining tree
structure, b) routing for forwarding data from source nodes to MEs, c) buffer management for different priority packets in order to decrease high priority packets’ delay and reduce packet loss.

5.1 Nodes Roles and States

For the distribution operation of the EQRoute in all sensor nodes, each node, except MEs, can have one of three roles in the network: None, Root, and Child; Each node can also have one of three states: Wait, Transfer, and Recovery.

All the nodes start in the Wait state with the None role. If a Wait state node receives a "Hello" message directly from MEs, the node’s state changes to Root role and starts requesting the capacities. The node will not start rebroadcasting "Hello" message until free capacities allocated, which follows Algorithm 2. Any nodes received this "Hello" message will change its role into Child role. The Child node will request for capacity from their parents. Each father node (including root and children node) will maintain a so called "Partition Timer" to decide when to start allocate its free capacity to its children nodes.

Any nodes, except MEs, with allocated capacities will change to Transfer state for transferring data, both relaying and sending. If any of the Child role nodes receive a "MobileHere" message from MEs, they will become Root role that directly send data to MEs, and responsible for maintaining their own trees.

A root node must maintain the ME lost timer during receiving the "MobileHere" message. If the ME moves far from the range of root node, the root node cannot hear the next "MobileHere" message, and therefore the timer expires. The root will change to Recovery state as soon as the timer expires. As mentioned in section 4.2, root node will start tree-recovery process to rebuild the connection, however, at the end of recovery process, either the root node change to Child role if successfully connected to other node, or destroy its tree.

Any nodes inside the tree will become Wait state with None role immediately if it consumes all its allocated capacity or destroyed by its parent node. It will wait until further "Hello" message to start above processes.

5.2 Data Routing

The EQRoute intentionally designed to cope well with the ME so that data routing from source to ME can be fluently accomplished along the collection tree. The distributed EQRoute protocol can provide seamless data aggregation without concern about the ME’s movement. The tree structure is well layered and self-organized according to the Algorithm 2 so that nodes can easily swap
between different aggregation trees by initialized parameters, which enable more flexible and stable for data routing.

Furthermore, the child nodes inside the tree will not store any information about the connection status between root and ME, neither maintaining any information regarding routing paths or routing tables. This approach hides the detailed end-to-end information for a child node to further reducing memory occupation and energy consumption. Child needs remember its father and its own children nodes (if it owns) to sending or relaying or stop sending if capacity consumed or it has received a "Destroy-tree" command from parents node.

However, because the limitation of bandwidth in ME, each node will have some limited service rate accordingly, e.g. the deeper layer node will have less service rate, so that the data forwarding rate will not exceed the bound to better guarantee the packet will send to ME successfully in time. The collision between nodes will also be reduced by adapting this limitation strategy because each node only occupies some of the bandwidth to parents instead of greedily taking all of it.

5.3 Buffer Management

Whilst every sensor node in the network has to have a buffer in order to save the data, there are a few studies on the buffer management. [15] proposed an idea with 3-layered limited size buffers: pre-client queue, send queue and transmit cache. They further reason that Link layer acknowledgements are not perfect: they are subject both to false positives and false negatives. Therefore, a transmit cache provides mechanism to detect and suppress duplicates. However, they propose very aggressive retransmission policy (up to 32 times) in order to reach 90% of delivery ratio, which is not proper in the delay constraint manner and neither with our QoS scenario. [40] proposed two shared queue, one for low priority packets, the other for high priority packets. They proposed an geographic based algorithm to dispatch limited service rate to two buffers according to the best service rate \( \mu \) for their objective function. However, their buffers size are unlimited and there was no retransmission policy. [41] studied a priority buffer management scheme in every node to ensure that if congestion occurs and if they have the global event ordering list, they can decide to give priority to certain events. However, this mechanism based on certain global information and this kind of information is very hard to grab.

In our EQRoute, to increase the packet delivery ratio, we also have an adaptive aggressive retransmission policy that every packet will be retransmitted at least 2 to 5 times according the priority of the packet (e.g, high priority packet retransmits 5 times while low priority packet retransmitting 2 times) if the link layer response signals an error (e.g, no ack, packet loss, etc.). Addition to this semi-aggressive retransmission policy, we have a hybrid queue system like
[15], which a transmit cache guarantees the retransmission policy. Hence we have a two types of size limited shared queue: private and forward, and both of them have low and high priority queue, we also adapt an idea from [40]. Generally, figure 5.2 illustrates our buffer design of EQRoute.

![Buffer Design Diagram]

**Figure 5.2. Buffer Design**

Based on such buffer design, the EQRoute and easily distinguish different packets and adapt different policy regarding to different level packets. The design provides two major advantages: one is delay constraint concerned, the other is delivery ratio guaranteed. Furthermore, we set the same level (low priority or high priority) forwarding packets always have higher priority against private packets, that is, the forwarding packets always preamp the private packets. We believe the forwarding packets have more delay constraints (because they have consumed some time on transmission) and more chance of loss (because they have travelled multi-hops).
6. Analysis

We consider a data collection tree with the mobile user as the root. Each node has \( n \) children and generates \( d \) data packets in each time slot. The amount of data packets generated in layer \( h \) is \( n^h d \) for a time slot. The generated data have probability \( p_H \) and \( p_L \) (high priority or low priority), where \( p_H + p_L = 1 \). The information value of a high priority packet and a low priority packet are denoted by \( w_H \) and \( w_L \), where \( 0 \leq w_L \leq w_H \leq 1 \).

6.1 Worst Case for Optimality

We analyze the optimality of our distributed algorithm in terms of \( U \) as stated in the objective function of the problem formulation. Let \( \alpha_1, \alpha_2, \ldots, \alpha_k \) and \( \beta_1, \beta_2, \ldots, \beta_k \) be the proportion of high and low priority data collected by the mobile user in each layer \( h \), where \( 0 \leq \alpha_h \leq 1, 0 \leq \beta_h \leq 1, h = 1, \ldots, k \). The sum of information value per communication cost, \( U \), in a tree can be calculated by

\[
U = \sum_{h=1}^{k} n^h d \frac{w_H}{h} p_H \alpha_h + \sum_{h=1}^{k} n^h d \frac{w_L}{h} p_L \beta_h. \tag{6.1}
\]

Given the capacity \( C_j \) of mobile \( j \), we also have

\[
C_j = \sum_{h=1}^{k} n^h d p_H \alpha_h + \sum_{h=1}^{k} n^h d p_L \beta_h. \tag{6.2}
\]

We then compare the \( U' \) obtained by our distributed algorithm with the \( U^* \) obtained by the optimal centralized algorithm. We show that the \( U^*/U' \) ratio has the following properties.

**Theorem 2.** The worst \( U^*/U' \) ratio occurs when \( w_H >> w_L \). It is bounded by

\[
\frac{w_H p_H A(k^*)}{w_H p_H A(m^*) + w_L p_L A(m^*)}, \text{ where } k^* = \left\lceil \frac{\log \left( \frac{C(n-1)}{d p_H} + n \right)}{\log n} + 1 \right\rceil, m^* = \left\lceil \frac{\log \left( \frac{C(n-1)}{d} + n \right)}{\log n} + 1 \right\rceil
\]

and \( A(k) = \sum_{h=1}^{k} n^h / h \).

**Proof.** The capacity is assigned from the top to the bottom of the data collection tree for the same data type due to the property of \( w > w/2 > \ldots > w/k \). Hence, \((\alpha_1, \alpha_2, \ldots, \alpha_k)\) is a sequence of number in pattern of \((1, 1, 1, \ldots, 0)\),
such that $1 = \alpha_1 = \ldots = \alpha_{k^* - 1} \geq \alpha_{k^*} > \alpha_{k^* + 1} = \ldots = 0$. Similarly, the sequence of $(\beta_1, \beta_2, \ldots, \beta_k)$ has the same property. Let $k^*$ and $m^*$ be the number of layers that are granted capacity for high and low priority data in the optimal algorithm. Since $w_H \geq w_L$, we have $k^* \geq m^*$. Consider that partial capacity may be granted in layer $k^*$ and layer $m^*$, the optimal $U^*$ must be bounded by

$$ U^* \leq \sum_{h=1}^{k^*} n^h d \frac{w_H}{h} p_H + \sum_{h=1}^{m^*} n^h d \frac{w_L}{h} p_L $$

$$ = dw_H p_H A(k^*) + dw_L p_L A(m^*), \quad (6.3) $$

where $A(k) = \begin{cases} \sum_{h=1}^{k} \frac{n^h}{h} & \text{if } k \geq 1 \\ 0 & \text{otherwise.} \end{cases}$

For our distributed algorithm, the capacity is assigned to both high and low priority data from the top layer. Let $k'$ and $m'$ be the number of layers that are granted capacity for high and low priority data. Since preemption for the high priority data occurs only in the lowest layer of the data collection tree, we have $m' \leq k' \leq m' + 1$. If we take the smaller value $m'$, $U'$ must be greater than the following

$$ U' \geq \sum_{h=1}^{m'} n^h d \frac{w_H}{h} p_H + \sum_{h=1}^{m'} n^h d \frac{w_L}{h} p_L $$

$$ = dw_H p_H A(m') + dw_L p_L A(m'). \quad (6.4) $$

Thus,

$$ \frac{U^*}{U'} \leq \frac{w_H p_H A(k^*) + w_L p_L A(m^*)}{w_H p_H A(m') + w_L p_L A(m')}. \quad (6.5) $$

From Eq. 6.5, we observe that $\frac{U^*}{U'}$ is maximized when $k^*$ is maximized. It occurs when $w_H >> w_L$, i.e. $m^* = 0$. In this case, only high priority data are selected in the optimal algorithm.

We further analyze the worst $U^*/U'$ ratio in this scenario. Considering $m^* = 0$, all the available capacity $C_j$ will be allocated for the high priority data in the optimal algorithm. Due to the capacity constraint, the total amount of high priority data collected must be smaller than $C_j$, i.e.

$$ C_j \geq \sum_{h=1}^{k} n^h d p_H = d p_H \frac{n^{k-1} - n}{n - 1}. \quad (6.6) $$

By solving this equation, we can obtain $k^* = \lceil k \rceil$ by taking the maximum $k$ with

$$ k \leq \log \left[ \frac{C_j (n-1)}{d p_H + n} \right] + 1. \quad (6.7) $$
Figure 6.1. The $U^*/U'$ ratio varying $w_L$

Figure 6.2. The $U^*/U'$ ratio varying $p_H$
Similarly, the total amount of high and low priority data collected must be smaller than $C_j$ in our distributed algorithm. We can obtain $m' = \lfloor m \rfloor$ by taking the maximum $m$ with

$$m \leq \frac{\log\left(\frac{C_j(n-1)}{d} + n\right)}{\log n} + 1. \quad (6.8)$$

By substituting $k^*$ and $m'$ into Eq. 6.5 with $m^* = 0$, we can obtain a bound for the ratio $U^*/U'$.

### 6.2 Numerical Results

We show the results of $U^*/U'$ with $p_H = 0.3$, $p_L = 0.7$, and $n = 3$ in Figure 6.1. It indicates that $U^*/U'$ decreases when $w_L$ increases. This is because the capacity is more evenly distributed among the high priority and low priority data in the optimal approach, so that it achieves similar value as our distributed approach. We also notice that $U^*/U'$ gets closer to one when capacity $C_j$ increases from 1000 data packets to 10000 data packets. The reason is that the data collection tree becomes bigger when $C_j$ increases. Hence, the difference between $w_H/c_{ij}$ and $w_L/c_{ij}$ becomes smaller as $c_{ij}$ increases.

We show the $U^*/U'$ ratio varying $p_H$ in Figure 6.2. Note that $p_L = 1 - p_H$, which indicates the proportion of high and low priority data generated from the network. When $p_H$ increases, the distributed approach allocates more capacity to the high priority data in each layer. This may lead to closer performance compared with the optimal approach. Again, the result shows that the $U^*/U'$ ratio is higher (worse) when $w_L$ is small.
7. Results and Evaluation

We evaluate the performance of our distributed algorithm, EQRoute, in OMNet++ simulator [42]. The sensors and the mobile devices communicate with IEEE 802.15.4 in non-beacon mode. The radio operates in the 2.4GHz band with an effective data rate of 250kbps. There are 100 sensors uniformly deployed in a 1000 m × 1000 m sensing field. The communication range $R$ of the wireless sensors is set to 100 m. We set $\Delta D = 100 m$, which is the same as $R$ in our experiments. The mobile users move independently following the exMass Mobility model (see 7.1), but we do not include any pause times.

7.1 exMass Mobility vs Random Waypoint Mobility

To better represent the uncontrollable mobility of user, we extended the Mass Mobility model [43] here called the exMass Mobility. The mobile user continuously moves in the rectangular area with the following pattern. Like the original Mass Mobility, the mobile user moves along a straight line for a certain period of time before it makes a turn. The moving period is a random number which normally distributed with average 100 seconds and standard deviation of 20 seconds. However, when it makes a turn, no limitation of direction is set to the mobile user compare to MassMobility (originally set to $\pm30$ degree), and a random number that follows normal distribution with average pre-set speed and standard deviation $\sigma m/s$ will be picked up as newly moving speed. When the mobile user hits the wall, it reflects off the wall at the same angle.

On the contrary, the random waypoint mobility only supports constant speed and only when the ME reach some pre-randomed position, the ME will get another destination. The random waypoint mobility model cannot fully reflect the walking pattern of humans’, especially it cannot present the unpredictable as well as uncertain speed during the movement.

However, both mobility models can not fully present the movement of people because in fact no one will reflect off the wall at the same angle. The only conclusion for these two comparison is has exMass Mobility has better presenting than random waypoint so that we adapt exMass Mobility in our simulation evaluations.
Figure 7.1. Evaluation with single mobile user
7.2 Single Mobile User

We first evaluate EQRoute in a single mobile user scenario with $p_H = 0.3$ and $p_L = 0.7$. The data generation rate of the sensors is set to $8\text{bytes/s}$. We vary the mean moving speed of the mobile user from $2m/s$ to $22m/s$ in our experiments with a standard deviation of $0.5m/s$. We compare our EQRoute algorithm with a recently proposed $\lambda$-Flooding algorithm [44] for ubiquitous data collection. In $\lambda$-Flooding, the mobile user builds a global data collection tree and updates the tree according to a predefined threshold $\lambda$ to reduce energy consumption in data collection.

Figure 7.1a shows the total number of received packets from EQRoute and $\lambda$-Flooding. We can see that both algorithms receive comparable number of packets. However, our EQRoute algorithm consumes only half of the energy compared with $\lambda$-Flooding as shown in Figure 7.1b. We believe that the energy consumption increases in $\lambda$-Flooding due to its prolonged paths to keep the connection with the mobile user. This is further verified by the increasing average hopcount in $\lambda$-Flooding as the speed increases (see Figure 7.1d). On the contrary, EQRoute adapts very well to the increasing speed as it allocates the capacity dynamically according to the available contact time and wireless capacity. This mechanism allows the sensors to stop reporting data automatically without sending extra control messages. EQRoutes achieves relatively constant energy consumption and average hopcount as shown in Figure 7.1b and Figure 7.1d. We also observe that the high priority packets may have slightly higher communication cost than the low priority packets. This is because the high priority packets may achieve higher $w_i/c_{ij}$ even though they are located farther away from the mobile user.

Figure 7.1c shows the average information value per communication cost (hopcount) of the collected data. EQRoute can achieve much higher information value per communication cost than $\lambda$-Flooding, since the highly valued data are given higher priority for collection. We also find that the information value per cost increases with the speed in EQRoute. This is because the mobile user builds smaller data collection trees with small hopcounts when it is moving fast.

7.3 Multiple Mobile Users

Next, we evaluate our EQRoute algorithm with multiple mobile users. We wanted to compare our work with existing approaches. Unfortunately, to the best of our knowledge, the existing works either focus on multiple mobile users with controllable mobility and rendezvous points [45], [46], or single mobile user for collecting delay tolerant events [22]. Although it was mentioned in [44] that $\lambda$-Flooding can be extended for multiple users, we find that it needs further investigation to support updates of multiple data collection trees and to handle packet collisions of multiple mobile users. Hence, we
Figure 7.2. Evaluation with multiple mobile users
focus on evaluating the performance of our EQRRoute algorithm in this experiment. We keep the same $p_H$ and $p_L$ settings as in the single user experiment, but we increase the data generation rate to 800 bytes/s to explore the full utilization of available capacity from multiple mobile users. We test with 1, 5, and 9 mobile users in the sensing field varying their average moving speeds.

Figure 7.2a shows that the total number of packets received by the mobile users. Obviously, more packets can be collected when there are more mobile users in the field. We show the average energy consumption of the sensors in Figure 7.2b. Since the sensors find more opportunities to report data to multiple mobile users, they consume more energy than reporting data to single mobile user. We measure the information value per communication cost in Figure 7.2c. It shows that EQRRoute can achieve higher information value per communication cost in general when the number of mobile users increases. However, the value per cost does not increase as much as expected with 9 mobile users at high moving speed. We believe that some capacity is wasted for the sensors to switch their routes among different mobile users. Figure 7.2d shows the average delay for transmitting high priority packets from the sensors to the mobile users. We can see that the average delay is the smallest with 9 mobile users. This is because the mobile users can divide the sensing field and construct smaller data collection trees. Similarly, the data collection trees become smaller when the mobile users are moving in high speed.

7.4 $p_H$ and $w_L$

![Figure 7.3. $p_H$ vs $w_L$](image-url)
We validate our analysis result in the simulation by varying the $p_H$ against $w_L$ with single ME and 2 m/s walking speed. We can see that figure 7.3 showing that information value per cost gains along with the $p_H$ and $w_L$. Obviously, the $w_L$ significant increase the information value when $p_H$ is low because the certain capacity can obtain same number of packets, either high priority packets or the other one so that increasing $w_L$ means gaining the total received packets information. However, the result also shows, the $w_L$ will not change the result if $p_H = 1$ when there is no low priority packet in the network.

Recall the analysis part in chapter 6, our simulation results show the same trend as in figure 6.2. However, the results for optimal algorithm can not be grabbed from the simulation due to many limitations, e.g. packet loss, packet retransmissions, etc. Hence we cannot compare the EQRoute and optimal in the simulation environment.

7.5 $\Delta D$ Investigation

![Graphs showing # Received Packets vs $\Delta D$, # Control Message vs $\Delta D$, Avg. Energy vs $\Delta D$, value Gain per Cost vs $\Delta D$](image)

Figure 7.4. $\Delta D$ Investigation

In this experiment, we study the dependency of $\Delta D$ in our EQRoute algorithm. Figure 7.4b shows the total number of received packets varying $\Delta D$. 
The number of received packets increases with $\Delta D$ at the beginning. It reaches maximum at around $\Delta D = R$. Then, it drops gradually as $\Delta D$ increases further. Recall Eq. 4.1, the mobile user estimates its available capacity based on $\Delta D$ and its moving speed $v$. If $\Delta D$ is too small, the capacity of the mobile user may be underestimated. The underestimated capacity limits size of the data collection tree and reduces the number of packets received. On the contrary, the capacity may be overestimated if $\Delta D$ is set too large. The tree size then becomes bigger than the network can actually support, which will cause collision and packet losses. Figure 7.4c shows the corresponding average energy consumption of the sensors. Intuitively, the energy consumption follows the same trend of the total number of received packets.

Figure 7.4d shows the information value per communication cost varying $\Delta D$. When $\Delta D$ is small, the information value per cost is high due to the small size of the data collection trees. However, the number of received packets is very low when $\Delta D$ is small (see Figure 7.4b), so that this parameter setting is not preferred. We also observe that the value per cost drops at the beginning and then it becomes quite steady afterwards. The reason is that the mobile user may move out of range from the data collection tree when $\Delta D$ is large. The loss of connection to the mobile users stops the sensors from transmitting data or expanding the tree, so that the communication overheads remain roughly the same.

From the above results, we conclude that the proper $\Delta D$ setting to be around $\pm 10\%$ of the wireless communication range $R$, so as to maximize the sum of the information value per communication cost from the received packets.
8. Conclusion and Future Work

In this thesis, we proposed EQRoute as a distributed and energy-efficient QoS ubiquitous data collection algorithm for multiple mobile users in WSNs. Our algorithm is able to estimate the available capacity and construct the data collection trees dynamically according to the moving speeds of the mobile users. This approach effectively controls the size of the data collection trees and reduces unnecessary packets losses and retransmissions due to disconnection with the mobile users. EQRoute obtains high information value by providing better QoS for high priority data. It achieves a good balance between information value and communication overhead by selecting the data with high information value per communication cost. Our distributed algorithm is scalable for multiple mobile users without extra coordination messages among the mobile users. Extensive simulation results demonstrated that our distributed algorithm can improve information value up to 50% and reduce energy consumption by 50% compared with the $\lambda$-flooding approach. It scales very well with multiple mobile users and achieves high information value and low communication overhead even with mobile users moving at high speeds.

For future work, we plan to further improve our distributed algorithm. We observe that our approach still have space to improve in particular when $p_H$ and $w_L$ are small. For instance, the mobile user may allocate capacity to a proportion of low priority data, so that more capacity can be saved for high priority data in the lower layers in the data collection tree. Another possible future direction is to explore the possibility to apply mobility prediction to improve information value and to reduce communication overhead.
References


