Lightweight, Latency-Aware Routing for Data Compression in Wireless Sensor Networks with Heterogeneous Traffics

You-Chiun Wang and Chia-Ting Wei

Abstract—In many applications and scenarios, sensors have to regularly report what they monitor from the environment and quickly notify the sink node of event occurrence in the sensing field. In-network data reduction technique such as data aggregation and data compression can help diminish the amount of data sent from sensors, which not only saves the network bandwidth but also preserves sensors' energy. However, such technique does not consider packet latency due to the aggregation or compression operation. When some sensors generate regular reports in lower data rates, their packets have to spend longer time to be aggregated or compressed, resulting in higher packet delays. Besides, when events occur, the network could suffer from instant congestion due to the generation of numerous event notifications. Motivating from the above observations, the paper develops a *lightweight, latency-aware routing for data compression (L2DC)* scheme to reduce packet latency when applying the compression technique to reduce the amount of data generated from sensors. L2DC gives event notifications a higher priority over regular reports and eliminates unnecessary notifications to avoid bursty network congestion. In addition, L2DC facilitates the data compression process by allowing each sensor to determine whether to keep packets for compression locally or to send them to a neighbor to be compressed in a distributed manner. Our L2DC scheme can be easily built on most ad hoc and sensor routing protocols because it provides auxiliary *redundant packet elimination* and *relay node selection* mechanisms to reduce packet latency. By using the AODV protocol as the example, simulation results demonstrate the effectiveness of the L2DC scheme.

Index Terms—data aggregation and compression, in-network data processing, packet delay, routing protocol, wireless sensor network.

1 Introduction

Due to the self-organization and cooperatively sensing capabilities, *wireless sensor networks (WSNs)* have received lots of research attention and been applied to various scenarios [1], [2]. A WSN contains many small but autonomous sensors, which are able to collect and analyze their surrounding information continually. Such information is then transmitted to a central sink node through a multi-hop routing manner. Recently, many WSN platforms and applications have been also developed [3].

In most WSN applications, the data traffics generated from sensors can be classified into two major categories. The first category is that sensors 'periodically' report what they have monitored to the sink node, which we call *regular data report (RDR)*. Sensors usually generate RDR packets in a constant rate and the sink node could tolerate moderately long delays of these packets. On the other hand, the second category is that sensors detect abnormal environmental data (for example, the data exceed a predefined threshold) and notify the sink node of this special situation, where we call it *urgent event notification (UEN)*. Because events often appear in an *unexpected* manner (in particular, at arbitrary positions and arbitrary time), sensors may generate UEN packets in a variable rate but the sink node should receive these packets as soon as possible. It is noteworthy that RDR and UEN packets can coexist in a WSN. Take the wildfire-monitoring application as an example. Sensors have to periodically send RDR packets to the sink node in order to construct the temperature map of a forest. However, once some sensors detect unusually high temperature, they have to immediately send UEN packets to help firemen extinguish forest fires as quickly as possible.

Owing to the nature of RDR traffics, sensors will send a large number of packets to the sink node, which not only occupies a great deal of network bandwidth but also consumes much energy of sensors. This means that important UEN packets have to compete with a large number of RDR packets and thus they may not arrive at the sink node in time. Furthermore, because sensors are usually powered by non-chargeable batteries, energy is a critical concern to them. One feasible solution to deal with the above dilemma of continually sending RDR data or conserving the network bandwidth and sensors' energy is to reduce the 'size' of sensing data [4]. In other words, we can provide *in-network data reduction* by aggregating or compressing RDR packets. However, many data aggregation/compression solutions require to combine a threshold number of RDR packets in order to efficiently conduct the aggregation/compression operation. When some sensors generate RDR packets in lower data rates, their packets have to wait for longer time to be aggreagated/compressed, thereby significantly increasing packet delays.

On the other hand, numerous WSN scenarios require k-coverage deployment to improve network robustness [5]–[7]. This means that when an event occurs, it will be detected by many nearby sensors, which instantly and simultaneously generate a large number of UEN packets to describe the 'same' event. These packets not only contend for the wireless medium to be transmitted but also require other sensors to relay them to the sink node. We call such a phenomenon *the bursty network congestion* because the network will be congested by these UEN packets during a small period (of event occurrence). Ideally, the sink node can know event occurrence if it receives just

The authors are with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, 80424, Taiwan.
E-mail: ycwang@cse.nsysu.edu.tw; m013040034@student.nsysu.edu.tw

few UEN packets describing that event[1]. This implies that the network will be actually filled up with 'redundant' UEN packets when events occur.

Based on the above two motivations, this paper proposes a *lightweight, latency-aware routing for data compression (L2DC)* scheme which considers the coexistence of RDR and UEN traffics in a WSN. Assuming that sensors generate RDR packets in different data rates and events may arbitrarily appear at any time, our L2DC scheme seeks to reduce the *latency* of both RDR and UEN packets when applying the data compression technique to RDR packets. To achieve this objective, the L2DC scheme involves three special designs:

1) prioritizing UEN packets over RDR packets in order to catch their deadlines,
2) removing unnecessary UEN packets to avoid bursty network congestion when events occur,
3) allowing each sensor to choose between doing data compression locally or finding the most suitable neighbor to relay and compress RDR packets according to the packet generating rates, buffer sizes, and channel conditions.

Our L2DC scheme is *distributed* in nature because every sensor relies on the information from only its one-hop neighbors to make its own decision. Simulation results show that our L2DC scheme can efficiently reduce packet latency compared with other methods. The contributions of this paper are three-fold:

- We point out that most in-network data reduction solutions (such as data aggregation and data compression) for WSNs do not consider the packet latency of compressed data. When sensors generate sensing data using different rates, it may cause some packets to suffer from larger latency. In this case, these packets may become useless even though they are (eventually) received by the sink node because of missing their timeliness.
- To solve the above problem, this paper develops the L2DC scheme which provides the *redundant packet elimination* mechanism to deal with UEN packets and the *relay node selection* mechanism to compress RDR packets. These two mechanisms can be easily built on many ad hoc and sensor routing protocols. In the simulation, we apply our L2DC scheme to the well-known *ad hoc on-demand distance vector (AODV)* protocol [8] to evaluate its performance in terms of network throughput, packet latency, and energy consumption.
- Our L2DC scheme is *lightweight* in the sense that it does not involve complex computation. Each sensor follows a simple rule to discard unnecessary UEN packets. Besides, the sensor can fast and easily determine whether to keep RDR packets for compression locally or to send them to the next-hop neighbor. This is especially practical for most WSNs because sensors are usually simple devices and therefore cannot conduct too complicated calculation [9].

The rest of this paper is organized as follows. The next section presents related work. The network model is given in Section 3. Section 4 proposes our L2DC scheme and Section 5 shows the experimental results. Finally, conclusions are drawn in Section 6.

1. In many cases, one UEN packet is enough to identify event occurrence.

2 Related Work

According to the study of [10], in-network data reduction schemes for WSNs can be categorized into *data aggregation* and *data compression*. In data aggregation (sometimes called *data fusion*), a subset of sensors are selected as aggregation nodes to 'merge' multiple sensing data by, for instance, taking their average, maximum, or minimum values. On the other hand, data compression allows the sink node to recover each RDR packet (possibly with some data loss) from the compressed packet. Then, we discuss several routing strategies which consider traffic differentiation in WSNs.

2.1 Data Aggregation in WSNs

Many data aggregation schemes organize the WSN into a 'structural' architecture such as cluster, tree, or chain, so that they can select a subset of sensors to serve as the *aggregation nodes* to deal with data aggregation. LEACH (low-energy adaptive clustering hierarchy) [11], HEED (hybrid energy-efficient distributed clustering) [12], and CCR (cluster-based coordination and routing) [13] belong to cluster-based data aggregation schemes because they group sensors into clusters (depending on the network conditions) and then select one *cluster head* to be responsible for gathering and aggregating the sensing data transmitted from its member sensors in each cluster. On the other hand, the research efforts in [14]–[16] seek to jointly optimize the routing-tree construction operation and the data aggregation process. Some intermediate nodes in the tree have to do the aggregation operation to reduce the amount of data transmitted along the tree. As can be seen easily, the above methods aim at efficiently selecting a small subset of sensors to do the aggregation job, while our L2DC scheme allows each sensor to do the compression operation by itself with the goal of reducing packet latency.

PEGASIS (power-efficient data-gathering protocol for sensor information systems) [17] forms a linear chain for sensors to transmit their sensing data to the sink node, where each sensor on the chain not only relays the data from its neighbor but also conducts data aggregation. On the other hand, the work of [18] considers a *long-thin* WSN composed of several long branches of sensors, where along a branch each sensor has one parent node toward the sink node. Then, it discusses how to dynamically select the aggregation node according to the data generation rates of sensors such that packet delays can be reduced. Although sharing the similar goal, the above two studies can only be applied to special (chain-based) topology. On the contrary, our work seeks to reduce packet latency when applying data compression to a general-topology WSN.

2.2 Data Compression in WSNs

The data compression technique designed for WSNs usually takes advantage of the correlation of sensing data to condense their size. For example, the studies of [19] and [20] respectively adopt LZW (Lempel-Ziv-Welch) coding and Huffman coding to interpret sensing data so that multiple packets can be combined and compressed together. Both the studies in [21] and [22] treat the sensing data as image pixels. Then, they adopt image processing solutions such as wavelet transformation to condense the sensing reports transmitted from sensors.

The Slepian-Wolf theorem [23] proves that two correlated data streams can be encoded independently and then be decoded jointly at a receiver with a rate equal to their joint

entropy. The distributed source coding technique is then developed by exploiting this theorem's property to support in-network data compression for WSNs [24], [25]. In addition, the compressive sensing schemes [26], [27] point out that any sufficiently compressible data can be correctly recovered from a small number of non-adaptive, randomized linear projection samples. Therefore, sensors can take advantage of this compressibility without any prior knowledge on their sensing data. However, the above research efforts target at how to reduce the amount of data transmitted from sensors, but they do not address how to reduce packet latency caused by their compression operations. This motivates us to develop an efficient routing strategy to deal with the long packet-latency problem caused by the data compression process.

2.3 Traffic-differentiation Routing in WSNs

A number of WSN routing protocols address *traffic differentiation* by dealing with various types of sensing data in different ways. Specifically, the work of [28] develops a multi-path routing protocol to support two QoS domains for data traffics: *timeliness* (that is, UEN traffic) and *reliability* (that is, RDR traffic). However, it requires to construct redundant routing paths to guarantee data reliability, which spends more network resource. In addition, this work needs special MAC operations to provide prioritized access and reliable multicast transmission to multiple neighbors. The study of [29] differentiates the WSN applications into *critical* and *non-critical* classes (which correspond to UEN and RDR applications, respectively). Each sensor uses different subsets of neighbors to relay different classes of sensing data but the same class of sensing data are forwarded to the neighbor calculated from the same function. Compared with this study, our L2DC scheme can adaptively select the most suitable neighbor for each sensor to help compress its sensing data such that the packet latency can be significantly reduced.

A QoS routing protocol is proposed in [30] by delivering packets into three queues: DSQ, RQ, and CQ, according to whether the packet is delay-sensitive (that is, a UEN packet), reliability-sensitive (that is, an RDR packet), or both of them. However, it has to send delay-sensitive data to both a primary sink node and a secondary sink node. This obviously results in a high overhead. On the other hand, the study of [31] adopts a potential field model to maintain accuracy for integrity-sensitive data (for example, RDR data) and reduce end-to-end delay for delay-sensitive data (for example, UEN data). However, this study does not consider using data compression to reduce the amount of integrity-sensitive data transmitted from sensors. Compared with existing schemes, our work contributes in developing a novel routing strategy to reduce packet latency of both RDR and UEN traffics when applying the data compression technique to WSNs.

3 Network Model

We are given a WSN to periodically collect environmental data and events will arbitrarily appear in the sensing field at any time. Each sensor s_i has its own data rate r_i to generate RDR packets in order to report its monitoring data to the sink node. In particular, sensor s_i spends r_i unit time (for example, second) to generate one RDR packet. Every $\delta \in \mathbb{N}$ RDR packets can be compressed into exact one *CDR (compressed data report)* packet, where $\delta \geq 2$ is a predefined parameter. CDR packets cannot be further compressed and they will be decompressed only at the sink node (The reason will be discussed in Remark 1). We define the latency of a CDR packet as the maximum latency of all its member RDR packets when it arrives at the sink node. Each sensor has two choices to determine whether to send out CDR or RDR packets:

1) Accumulate δ RDR packets and then send only one CDR packet (using data compression) to a neighbor. In this case, the neighbor just relays the CDR packet without any further processing (for example, data compression).
2) Send a number of RDR packets to a neighbor. In this case, the neighbor may compress these packets together with its own RDR packets.

For convenience, we assume that every sensor s_i maintains a small buffer to accumulate RDR packets, whose length is denoted by the variable b_i. When sensor s_i compresses some RDR packets, they will be removed from the buffer accordingly.

When a sensor detects an event, it 'immediately' generates the corresponding UEN packet to notify the sink node. UEN packets have urgent deadlines and they are thus *incompressible*. Instead, sensors should transmit UEN packets as fast as possible to meet their deadlines. Since there could be multiple UEN packets generated because of the same event, we define the latency of these UEN packets (describing the same event) as the latency of the first UEN packet arriving at the sink node. Then, our objective is to develop a routing strategy with data compression which can reduce the latency of both CDR and UEN packets. Table 1 summarizes the notations used in this paper.

Remark 1 (Data decompression). One may suggest decompressing the received CDR packet at a sensor and then compressing more than δ RDR packets together. Although this method can further reduce the amount of transmitted data, it has two major drawbacks. First, sensors have to conduct complex decompression and compression operations many times, which may not be suitable for simple sensor devices. Second, this method may increase the latency of CDR packets because sensors should spend more time to decompress packets and to wait its RDR packets to be generated. Therefore, we assume that sensors will simply relay CDR packets in this paper.

4 The Proposed L2DC Scheme

The objective of our L2DC scheme is to help each sensor select the most suitable neighbor to relay and compress the sensing data. Therefore, it can be built on most ad hoc and sensor routing protocols. In particular, for table-driven routing protocols, each sensor keeps a *routing table* whose routing metric is determined by our L2DC scheme to select the next-hop neighbor. For on-demand routing protocols, *route request (RREQ)* and *route reply (RREP)* packets are exchanged between sensors to construct routing paths. Then, our L2DC scheme assists a sensor in selecting the next-hop neighbor based on the received RREP packets.

Except for the original operations defined in the routing protocol, the L2DC scheme involves the following four special designs:

1) **Information exchange:** Sensors are assumed to know the δ threshold value in advance because they have

TABLE 1: Summary of notations.

notation	definition
δ	the threshold number of RDR packets required to be compressed into one CDR packet
r_i	the RDR packet generating rate of sensor s_i
b_i	the number of RDR packets residing in the buffer of sensor s_i
ξ	the initial value of TTE_i, which is the expected diameter of an event detection region (in hop count)
φ_G	a small gap to help a sensor choose between keeping RDR packets or sending them to a neighbor
t_P	a short time to let a sensor postpone sending out RDR packets (to deal with the situation in Remark 6)
I_u, V_u	two threshold values to help a sensor check for redundant UEN packets
$\mathcal{N}_i$	the set of one-hop neighbors of sensor s_i that have shorter or equal hop counts to the sink node
T_i^A	the expected queuing time of a CDR packet if sensor s_i keeps accumulating RDR packets in its local buffer
$T_{i,j}^S$	the expected queuing time of a CDR packet if sensor s_i sends its current RDR packets to a neighbor sensor s_j
$c(i,j)$	the transmission rate of the communication link between two sensors s_i and s_j
$f_c(\delta)$	the compression time to merge δ RDR packets into one CDR packet

to use this value to conduct the data compression algorithm. However, they should also exchange some additional information with their neighbors. In particular, each sensor s_i should announce its generating rate r_i of RDR packets and the current buffer length b_i to its one-hop neighbors. If the L2DC scheme is built on a table-driven routing protocol, then s_i can add both r_i and b_i parameters to its routing table. In this way, the above announcement can be realized by the table-exchange mechanism (originally defined in the routing protocol). On the other hand, if an on-demand routing protocol is employed, then s_i should embed both r_i and b_i parameters in the RREP packets. As we will discuss later, each sensor requires the r_i and b_i values from its one-hop neighbors which are closer to the sink node to do the calculation. That is why we add these parameters in RREP packets instead of RREQ packets. Alternatively, we can embed both r_i and b_i parameters in *hello messages*. Since sensors need to periodically broadcast hello messages to maintain the relationship with their neighbors, this scheme can allow a faster update of r_i and b_i values.

2) **Packet differentiation:** Because of their characteristics, we have to assign different *priorities* to RDR, CDR, and UEN packets. Obviously, UEN packets should always have the highest priority owning to their critical deadlines. On the other hand, CDR packets should be given precedence over RDR packets since they cannot be compressed (and therefore can be directly relayed to the sink node without any further processing). Two possible schemes can deal with the above packet differentiation. One scheme is to maintain a *priority queue* for each sensor to arrange the transmission sequences of different types of packets [32]. The other scheme is to allow every sensor to possess three queues (or buffers) to separately store RDR, CDR, and UEN packets to be transmitted. This multi-queue scheme is also considered in some wireless standards such as IEEE 802.11e [33].

3) **Elimination of redundant UEN packets:** When an event occurs, it is usually detected by a number of sensors, especially when sensors are densely deployed in the sensing field. In this case, multiple UEN packets will be generated by these sensors. However, as we have discussed earlier, UEN packets are assigned with the highest priority. Therefore, a large number of sensors will relay (and possibly rebroadcast) the UEN packets. This could result in two serious effects. First, the network would be suddenly congested by a large number of packets, where we call this phenomenon *bursty congestion*. Many UEN packets could be dropped by intermediate sensors due to their full queues or the loss of wireless medium contention. Second, other non-UEN packets may be starved because of their lower priorities. To solve the above problems, we will develop a *redundant packet elimination* mechanism in Section 4.1.

4) **Transmission and compression of RDR packets:** Because sensors will continually transmit what they have monitored to the sink node, the network is expected to be dominated by RDR traffics in most time. Therefore, we should conduct in-network data reduction by compressing RDR packets to alleviate potential network congestion. However, conventional WSN data compression schemes require each sensor to accumulate a threshold number of RDR packets. They will thus significantly increase RDR packet latency. Therefore, we should allow every sensor to adaptively choose between accumulating RDR packets in its local buffer or transmitting some RDR packets to its neighbor(s). In Section 4.2, we will propose an efficient *relay node selection* mechanism to facilitate the compression process for RDR packets.

4.1 Redundant Packet Elimination Mechanism for UEN Packets

When an event appears, there could be multiple sensors which detect its appearance. Furthermore, some types of events such as leakage of chemicals or wild fire usually 'spill' over a small region containing a number of sensors. In this case, we say that the sensing field has an *event detection region* inside which all sensors are aware of the same event. For example, the event detection region contains seven sensors (denoted by $s_1 \sim s_7$) in Fig. 1. Since these sensors may be the neighbors with each other, there is a high possibility that they choose the same (or overlapped) routing path(s) to reach the sink node. However, because UEN packets have the highest priority due to their critical deadlines and multiple sensors in the same event detection region usually generate their UEN packets *simultaneously*, these routing paths could be instantly congested by a large number of UEN packets. In this case, we say that these routing paths form a *bursty congested region* by UEN packets, as shown by the gray region in Fig. 1.

It is apparent that bursty congested regions would occupy the most area in the sensing field if there are more event detection regions (due to multiple event occurrence). In this case, the network performance may significantly degrade because it is congested by numerous high-priority UEN packets. However,

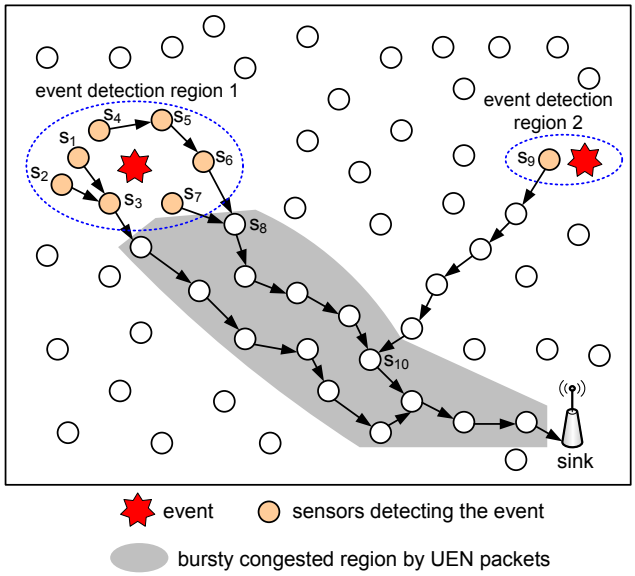


Fig. 1: Bursty network congestion due to UEN packets.

since the sensors in an event detection region actually observe the same event, their sensing readings should possess *spatial correlation* [22]. In other words, they could generate the similar UEN packets with almost the same sensing values. Therefore, we can allow the sensors in the same event detection region to discard *redundant* UEN packets so that the corresponding bursty congested region can be eliminated.

However, the major challenge is how to identify the sensors in the same event detection region. One intuitive approach is to let each sensor know the exact positions of its nearby sensors and allow it to exchange the sensing readings with neighbors. Then, the sensor can check whether it has detected the same event with a neighbor according to their sensing readings and relative positions. Nevertheless, this approach has three disadvantages. First, sensors require the positioning knowledge, which may not be feasible in some WSN applications. Second, sensors have to exchange extra sensing readings with their neighbors, which waste more network bandwidth. Third, this approach may rely on complex geometric calculation to identify the event detection region.

Therefore, we propose a simple but practical approach to remove *redundant* UEN packets without the positioning information of sensors in the L2DC scheme. Our idea is based on the observation from Fig. 1. In particular, since sensors will transmit UEN packets to the same destination (that is, the sink node) and they locate inside the same event detection region, there is a high possibility that they will relay their UEN packets to the same neighbor. In this case, the neighbor in fact can determine whether these UEN packets describe the same event or not. If so, it can send out only one UEN packet and discard others. Therefore, we can avoid forming the corresponding bursty congested region.

Our approach is detailed as follows. We define a five-tuple format for UEN packets generated by a sensor s_i:

$$(\mathrm{ID}_i, t_i, \mathrm{type}_i, \mathrm{value}_i, \mathrm{TTE}_i), \tag{1}$$

where ID_i is the identification of sensor s_i, t_i is the current timestamp (that is, the time when s_i detects the event), type_i is the type of event being detected, value_i is the value of sensing reading from s_i, and TTE_i means *time-to-elimination*. When s_i detects an event and generates the corresponding UEN packet, its TTE_i value is initially set to a small constant $\xi \in \mathbb{N}$. Then, whenever the UEN packet is relayed in one hop, its TTE_i value will be decreased by one (until $\mathrm{TTE}_i = 0$). Here, the value of ξ depends on the expected diameter of an event detection region (in hop count). For example, if we expect that an event detection region will cover at most three hops of sensors, then we can set $\xi = 3$. Obviously, when events may spread out a large range (for example, leakage of chemicals), a larger ξ value can be assigned. Otherwise, a smaller ξ value should be given.

Two threshold values I_u and V_u are also defined to help a sensor determine whether the received UEN packet is redundant or not, where I_u and V_u depend on the application requirements. In particular, suppose that a sensor s_j receives a UEN packet, say, $p_i^U = (s_i, t_i, \mathrm{type}_i, \mathrm{value}_i, \mathrm{TTE}_i)$ from its one-hop neighbor s_i. If sensor s_j neither generates its own UEN packet nor receives any UEN packet, then s_j sends out packet p_i^U and keeps a copy of p_i^U for the future reference. Otherwise, sensor s_j must either generate (and send) its own UEN packet or have a reference UEN packet, say, $p_j^U = (s_j, t_j, \mathrm{type}_j, \mathrm{value}_j, \mathrm{TTE}_j)$. Then, sensor s_j can discard packet p_i^U if the following four conditions are all satisfied:

- $|t_i - t_j| \leq I_u$: This condition indicates that both sensors s_i and s_j observe an event at the similar time, which implicitly implies that their observations may have temporal correlation.
- $\mathrm{type}_i = \mathrm{type}_j$: This condition indicates that these two sensors observe the same type of event.
- $|\mathrm{value}_i - \mathrm{value}_j| \leq V_u$: This condition (together with the previous one) indicate that the readings of sensors s_i and s_j are similar, which implicitly implies that their observations could have spatial correlation.
- $\mathrm{TTE}_i > 0$: This condition indicates that sensors s_i and s_j could be in the same event detection region.

Specifically, these four conditions together imply that there is a very high possibility that both sensors s_i and s_j in fact observe the same event. Therefore, it is safe for sensor s_j to discard the redundant packet p_i^U.

Fig. 1 presents an example, where we assume that $\xi = 3$ and focus on event detection region 1. Sensor s_3 will send out only one packet p_3^U on behalf of the three sensors s_1, s_2, and s_3. On the other hand, sensor s_5 will discard packet p_4^U sent from sensor s_4 while sensor s_6 will send only its own UEN packet p_6^U to sensor s_8. Then, taking packet p_7^U as the reference, sensor s_8 will discard packet p_6^U sent from sensor s_6. Therefore, only two UEN packets (that is, p_3^U and p_7^U) are eventually transmitted to the sink node (to describe the event occurring in event detection region 1), and we can eliminate the bursty congested region in Fig. 1.

Several interesting issues arise in the above design, which will be addressed in the following three remarks.

Remark 2 (Stale UEN reference problem). One may argue that a sensor keeps an old UEN reference and thus makes a wrong decision, which we call it the *stale UEN reference problem*. However, this problem can be easily defeated by the condition of $|t_i - t_j| \leq I_u$. Specifically, t_i is the time when sensor s_i detects the event and it cannot be modified, so an old UEN reference will make the above condition failed. Therefore, sensors can always allow 'fresh' UEN packets to be sent out. In other words, we can use the t_i

field of UEN packets to help distinguish different events in the *temporal* domain.

Remark 3 (Simultaneous multi-event occurrence problem). It is possible that two or more same-type events occur simultaneously in the sensing field and these events cause sensors to have the similar readings. In this situation, the first three conditions together (that is, $|t_i - t_j| \leq I_u$, $\text{type}_i = \text{type}_j$, and $|\text{value}_i - \text{value}_j| \leq V_u$) cannot help distinguish these events and thereby making sensors drop important UEN packets. Fig. 1 gives an example, where there are two event detection regions (causing by the same-type events). In this example, we require UEN packets p_7^U and p_9^U to respectively describe the events occurring in the event detection regions 1 and 2. However, sensor s_{10} will receive both p_7^U and p_9^U and thus drop p_9^U (since the above three conditions are all satisfied). In this case, the sink node will not know that an event occurs in event detection region 2. We call this problem the *simultaneous multi-event occurrence problem*. To solve the problem, we introduce the concept of time-to-elimination TTE_i in the fourth condition. In particular, if a UEN packet has a zero TTE_i, it means that the UEN packet in fact leaves its event detection region. Therefore, it should be kept and relayed to the sink node. We take the example in Fig. 1 again, where $\xi = 3$. When packets p_7^U and p_9^U are relayed to sensor s_{10}, their TTE_i values will be decreased to zero. Therefore, s_{10} will directly relay both UEN packets to the sink node. To sum up, we can use the TTE_i field of UEN packets to help distinguish different events in the *spatial* domain.

Remark 4 (Queuing policy for multiple events). In L2DC, UEN packets have the highest priority to be sent out. However, when a sensor receives multiple UEN packets (and they are not redundant), if the sensor simply adopts the *first-in, first-out (FIFO)* strategy to transmit them, those UEN packets that describe earlier occurrence of events would be delayed. In this case, the event notification latency by the sink node may significantly increase. To address this issue, the sensor can sort the received UEN packets by their t_i values (that is, the event occurrence time) in an ascending order. Therefore, those 'older' UEN packets can arrive to the sink node earlier.

4.2 Relay Node Selection Mechanism for RDR Packets

Since the L2DC scheme aims at reducing the packet latency caused by the data compression process, sensors have to evaluate the *expected queuing time* of every CDR packet under different network situations. In particular, the evaluation should be done by each sensor according to the following three factors:

- the RDR packet generating rate,
- how many RDR packets residing in the local buffer (in other words, the buffer length),
- the transmission rate of the communication link to a neighbor.

Specifically, each sensor s_i computes the queuing time of a CDR packet (which expects to stay in its buffer) if s_i chooses to accumulate up to δ RDR packets and compresses these packets by itself:

$$T_i^A = r_i \times (\delta - b_i) + \frac{1}{c(i,k)} + f_c(\delta), \tag{2}$$

where $c(i,k)$ is the transmission rate of the communication link between sensors s_i and its next-hop neighbor s_k, and $f_c(\delta)$ is the time spent to compress δ RDR packets to one CDR packet. Obviously, we should guarantee that $b_i < \delta$. In case of $b_i \geq \delta$, sensor s_i has to immediately compress δ RDR packets and then sends one CDR packet to sensor s_k. In Eq. (2), the first term, $r_i \times (\delta - b_i)$, indicates the expected waiting time to calculate the CDR packet while the second term, $\frac{1}{c(i,k)}$, indicates the transmission time spent to relay the CDR packet from sensor s_i to sensor s_k. Notice that since sensor s_i already has b_i RDR packets in its buffer, it needs to wait for generating the remaining $(\delta - b_i)$ RDR packets for compression, where s_i spends r_i time to generate each RDR packet.

Let $\mathcal{N}_i$ be the set of sensor s_i's one-hop neighbors that have shorter or equal hop counts to the sink node. For each sensor $s_j \in \mathcal{N}_i$, sensor s_i calculates the expected queuing time of a CDR packet if it decides to transmit its current RDR packets to s_j and ask s_j to generate the CDR packet:

$$\begin{aligned} W_j &= r_j \times (\delta - (b_j + \min\{(\delta - b_j), b_i\})), \\ R_j &= \frac{\min\{(\delta - b_j), b_i\}}{c(i,j)}, \\ T_{i,j}^S &= \max\{W_j, R_j\} + f_c(\delta). \end{aligned} \tag{3}$$

In Eq. (3), W_j indicates the expected waiting time of sensor s_j to generate the CDR packet if it receives the RDR packets from sensor s_i. On the other hand, R_j indicates the transmission time required to send these RDR packets to sensor s_j. Since it is possible that $b_i + b_j > \delta$, which means that sensor s_i currently has more RDR packets than sensor s_j requires. Thus, it is sufficient for sensor s_i to transmit only $(\delta - b_j)$ RDR packets to sensor s_j in order to save the network bandwidth. That is why we choose the minimum value between $(\delta - b_j)$ and b_i in Eq. (3).

Notice that when sensor s_i transmits packets to sensor s_j, s_j still keeps accumulating its own RDR packets in the meanwhile. This implies that either W_j is a subperiod of R_j or R_j is a subperiod of W_j. Therefore, $T_{i,j}^S$ should take the maximum value between W_j and R_j. It is noteworthy that W_j is no smaller than R_j in most situations. However, when the communication link (s_i, s_j) encounters a bad-channel condition (that is, $c(i,j)$ becomes smaller) or sensor s_j generates RDR packets in a fast rate (that is, r_j becomes larger), it is possible that $R_j > W_j$ in Eq. (3). In this case, $T_{i,j}^S$ will be dominated by the transmission time of RDR packets from sensor s_i to sensor s_j.

By combining Eqs. (2) and 3, sensor s_i can choose between keeping its RDR packets or transmitting these packets to a neighbor according to the following equation:

$$\arg\min \left\{ T_i^A, \min_{s_j \in \mathcal{N}_i} \{T_{i,j}^S\} + \varphi_G \right\}, \tag{4}$$

where φ_G is a small gap (for instance, 0.5 second). In particular, if T_i^A is the minimum value in Eq. (4), sensor s_i then decides to accumulate the RDR packets in the local buffer to compress these packets by itself. Otherwise, sensor s_i decides to transmit the RDR packets to the neighbor s_j such that $T_{i,j}^S$ has the minimum value, where the number of RDR packets required to be sent out can be determined by Eq. (3). Here, the small gap φ_G helps sensor s_i determine whether it is 'worth' sending RDR packets to a neighbor s_j when the difference between T_i^A and $T_{i,j}^S$ is quite small. In particular, when T_i^A is no larger than $\min_{s_j \in \mathcal{N}_i} \{T_{i,j}^S\}$ by the gap φ_G, sending RDR packets to a

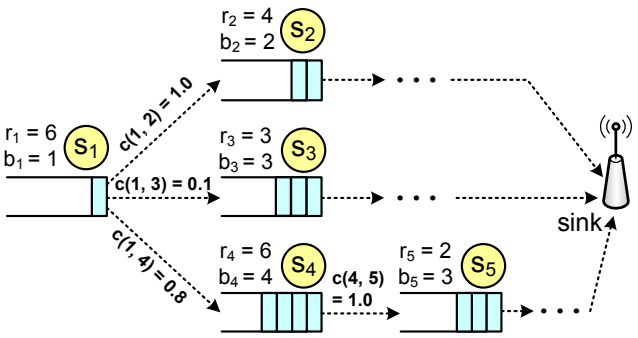


Fig. 2: An example of the relay node selection mechanism.

neighbor for compression can only save time of φ_G. However, if sensor s_i chooses to keep these RDR packets and compress them by itself, s_i will send out only one CDR packet instead of multiple RDR packets. Therefore, we can save sensor s_i's bandwidth at the cost of increasing the packet latency by no more than φ_G in this situation.

We then discuss how to estimate the transmission rate $c(i,j)$ of a communication link between sensors s_i and s_j, which is required by both Eqs. (2) and 3. Two possible solutions can help estimate the transmission rate. The first solution is to adopt the CSMA/CA (carrier sense multiple access with collision avoidance) throughput analysis [34], [35] to predict the transmission rate of a communication link. However, this solution may involve complex calculation and can be only applied to sophisticated sensors. Alternatively, we can allow each sensor to compute the average number of packets successfully transmitted through a communication link in the near past[2]. This solution is much easier and therefore could be applied to (most) simple sensor platforms.

Remark 5 (Removal of compression time in calculation). In Eqs. (2) and 3, the calculation of both T_i^A and $T_{i,j}^S$ take the compression time $f_c(\delta)$ into account. In other words, $f_c(\delta)$ is in fact a 'constant' cost no matter *where* sensor s_i decides to compress these δ RDR packets (that is, by itself or by another sensor). Therefore, we can remove the compression time $f_c(\delta)$ from both Eqs. (2) and 3 to simplify the calculation.

Fig. 2 presents an example to illustrate how a sensor s_1 makes the decision to generate its CDR packet by our relay node selection mechanism, where $\mathcal{N}_1 = \{s_2, s_3, s_4\}$, $\delta = 6$, and φ_G = 0.5 second. Suppose that sensors s_1, s_2, s_3, and s_4 require 6, 4, 3, and 6 seconds to generate an RDR packet and their buffers currently have 1, 2, 3, and 4 RDR packets, respectively. The transmission rates of the communication links (s_1, s_2), (s_1, s_3), and (s_1, s_4) are 1.0, 0.1, and 0.8 packets per second, respectively. Then, according to Eq. (2), sensor s_1 can calculate the expected queuing time of the CDR packet if it decides to keep all RDR packets in its local buffer:

$$T_1^A = 6 \times (6-1) + \frac{1}{1.0} = 31 \text{ seconds},$$

where sensor s_1 selects its neighbor s_2 to relay the CDR packet. As mentioned in Remark 5, the compression time $f_c(\delta)$ can be removed from both Eqs. (2) and 3, so we omit $f_c(\delta)$ in the following calculation. Following Eq. (3), sensor s_1 can also compute the expected queuing time of the CDR packet if it chooses to transmit the current RDR packet to one of its three neighbors:

sensor s_2:

$$T_{1,2}^S = \max\left\{4 \times (6-(2+1)), \frac{1}{1.0}\right\} = 12 \text{ seconds},$$

sensor s_3:

$$T_{1,3}^S = \max\left\{3 \times (6-(3+1)), \frac{1}{0.1}\right\} = 10 \text{ seconds},$$

sensor s_4:

$$T_{1,4}^S = \max\left\{6 \times (6-(4+1)), \frac{1}{0.8}\right\} = 6 \text{ seconds}.$$

From the example in Fig. 2, it can be observed that conventional data compression schemes will force sensor s_1 to wait at least $T_1^A = 31$ seconds to send out the CDR packet. Despite considering the gap φ_G, sensor s_1 should transmit its RDR packet to one of its neighbors so as to reduce the latency of CDR packet. Furthermore, it is not always true to minimize the CDR packet latency if the sensor transmits the RDR packet to the neighbor which has the highest RDR generating rate (that is, sensor s_3). Instead, we need to consider not only the RDR generating rate but also the buffer length and the channel quality of the communication link. Therefore, according to Eq. (4), sensor s_1 will decide to send its RDR packet to sensor s_4 so that we can reduce the CDR packet latency to only 6 seconds (from sensor s_1's perspective). This example demonstrates the design rationale of our relay node selection mechanism to deal with RDR packets.

We finally discuss several issues extending from the above example, as given in the following two remarks.

Remark 6 (Issue of when to make the decision). Sometimes, it is possible that a sensor s_i decides to transmit the RDR packets to its neighbor s_j, but before these RDR packets arrive to sensor s_j, sensor s_j has already sent out its RDR packets to another neighbor. Fig. 2 gives an example, where we focus on sensor s_4. Suppose that sensor s_4 has not received the RDR packet coming from sensor s_1 yet. Then, it computes the expected queuing time of the CDR packet based on the current RDR packets in its buffer by Eq. (2):

$$T_4^A = 6 \times (6-4) + \frac{1}{1.0} = 13 \text{ seconds}.$$

On the other hand, sensor s_4 also computes the expected queuing time of the CDR packet if it chooses to transmit its RDR packets to the neighbor s_5 by Eq. (3):

$$T_{4,5}^S = \max\left\{2 \times (6-(3+3)), \frac{3}{1.0}\right\} = 3 \text{ seconds}.$$

Then, according to Eq. (4), sensor s_4 will send three RDR packets to sensor s_5 for compression. However, in our previous example, sensor s_1 decides to send its RDR packet to sensor s_4. In this case, this RDR packet has to wait (in sensor s_4's buffer) for

$$6 \times (6-((4-3)+1) = 24 \text{ seconds},$$

which is larger than the expected 6 seconds in the previous example. This situation may sometimes occur but it is inevitable since sensors rely on the information from only the one-hop neighbors to make their decisions in a distributed manner. One possible solution to alleviate this problem is

2. For example, in our simulation each sensor estimates its $c(i,j)$ value every five seconds.

to ask each sensor to postpone making the decision (by a small time t_P) if it just announced the r_i and b_i parameters to the neighbors. Take Fig. 2 as an example again. After sensor s_4 announces its r_4 and b_4 values to sensor s_1, it does not decide whether to keep the RDR packets or send them to sensor s_5 right away. Instead, sensor s_4 waits time t_p so that it can get the RDR packet from sensor s_1 (and thus avoiding the above situation). Notice that sensors do not announce their r_i and b_i at the same time, so adding t_P could help each sensor receive the RDR packets coming from its upstream neighbor, before the sensor sends its RDR packets to the downstream neighbor (that is, the next-hop neighbor). Besides, the length of t_P should be shorter than the period that a sensor announces its r_i and b_i parameters. For instance, we can take the half announcement period as t_P's length.

Remark 7 (Issue of residual RDR packets). When $b_i > \delta - b_j$, a sensor s_i will not empty out its buffer when it decides to send RDR packets to a neighbor s_j for compression. Let us consider the example in Fig. 2 again, but sensor s_1 now has three RDR packets in its buffer. Based on Eq. (2), we can calculate that

$$T_1^A = 6 \times (6-3) + \frac{1}{1.0} = 19 \text{ seconds}.$$

Then, according to Eq. (3), we can also derive that

sensor s_2 (for sending three RDR packets):

$$T_{1,2}^S = \max\left\{4 \times (6-(2+3)), \frac{3}{1.0}\right\} = 4 \text{ seconds},$$

sensor s_3 (for sending three RDR packets):

$$T_{1,3}^S = \max\left\{3 \times (6-(3+3)), \frac{3}{0.1}\right\} = 30 \text{ seconds},$$

sensor s_4 (for sending two RDR packets):

$$T_{1,4}^S = \max\left\{6 \times (6-(4+2)), \frac{2}{0.8}\right\} = 2.5 \text{ seconds}.$$

In this example, one may suggest estimating the waiting time of the residual RDR packet in s_1's buffer when s_1 chooses neighbor s_4 to do the compression. If the waiting time is longer, then s_1 should choose neighbor s_2 to do the compression. The above operation seems to be more sophisticated. However, it is not easy for *simple sensors* to estimate the waiting time of residual RDR packets because the estimation involves the prediction of 'future' network condition. Furthermore, even though we let sensors estimate the waiting time based on the 'current' network condition, the result may become inaccurate when some sensors change their RDR packet generating rates r_i or the transmission rates of some communication links $c(i,j)$ are varied. Such inaccurate estimation may increase the latency to generate CDR packets. Therefore, we still comply with the rule in Eq. (4) and thus ask sensor s_1 to relay its two RDR packets to neighbor s_4 to speed up the compression process.

5 Experimental Results

In this section, we measure the performance of our L2DC scheme through the *network simulator (NS-2)* [37], where Table 2 presents the simulation parameters. A rectangle-shaped sensing field with length of 150 meters and width of 100 meters is considered in the simulation. Totally 150 nodes (including the sink) are deployed inside the sensing field by using a grid-based fashion, as shown in Fig. 3. Specifically, the grid length is set to 10 meters while the transmission range of sensors is set to 18 meters. In this way, most sensors each can have three neighbors with shorter hop counts to the sink node. In addition, we can apply a coordinate system to the above grid-based deployment by selecting the sensor at the bottom left corner to be the origin $(0,0)$. Then, the sink node locates at coordinate $(14,4)$.

TABLE 2: Simulation parameters.

parameter	value
size of sensing field	150 meters × 100 meters
number of nodes	150
sensor deployment	grid-based deployment (in Fig. 3)
transmission range	18 meters
antenna	omnidirectional
packet size	1000 bytes
MAC protocol	CSMA/CA
PHY path loss model	free-space path loss model
PHY propagation model	shadowing model
RDR data rate	0.01–0.1 Mbps (random)
UEN data rate	0.01–0.1 Mbps (random)
channel bandwidth	11 Mbps
simulation time	1500 seconds
energy per bit transmission	10^{-5} J [36]

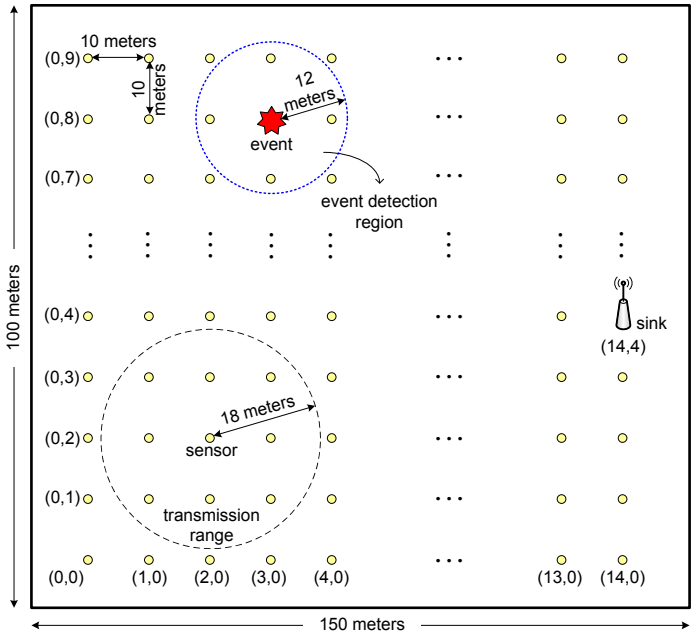


Fig. 3: The grid-based sensor deployment in the simulation.

Each sensor continually generates RDR packets during the whole 1500-second simulation time, whose generating rate is randomly picked from [0.01, 0.1] Mbps (megabits per second). Every 60 seconds a number of events occur at some coordinates in the sensing field, where an event will continue for [5, 10] seconds and then disappear. The maximum radius of an event detection region is set to 12 meters. In other words, at most five sensors can detect the same event, as shown in Fig. 3. However, any two event detection regions have no overlap so that one single sensor will not detect two or more events at the same time. We consider three types of events and their corresponding sensor readings (that is, value$_i$ in Eq. (1)) range in [30, 35], [60, 65], and [90, 95].

We compare three schemes in our simulation:

- **AODV:** This is the pure AODV protocol, where sensors

construct their routing paths by exchanging RREQ and RREP packets. Each sensor selects the one-hop neighbor that has a shorter hop count to the sink node to relay its packets. However, sensors neither employ any data compression technique (so there are no CDR packets generated) nor distinguish UEN packets from RDR packets.

- **AODV-FC:** Here, 'FC' means 'full compression'. Sensors use the AODV protocol to choose their next-hop neighbors. However, each sensor has to accumulate up to δ RDR packets in its queue and sends out only CDR packets. UEN packets need not be compressed but they have to wait to be transmitted if there exist uncompressed RDR packets in front of the queue (because AODV-FC does not differentiate between RDR/CDR and UEN packets).
- **L2DC:** We build our L2DC scheme on the AODV protocol[3], where sensors can not only determine where to compress RDR packets but also eliminate redundant UEN packets. In L2DC, we set the parameters as follows: ξ = 2 hops, I_u = 7.5 seconds (that is, the average period of event occurrence), V_u = 10, φ_G = 0.5 second, and t_P = 1 second.

The δ threshold value and the number of events are varied to investigate their effects. We then evaluate four performance metrics in the simulation:

- **RDR data throughput:** Its definition is the average amount of RDR data successfully received by the sink node (measured in megabytes in every second, 'MB/sec' for short). For both the AODV-FC and L2DC schemes, if the sink node correctly receives a CDR packet, it means that δ RDR packets have been successfully sent to the sink node.
- **CDR packet latency:** We follow the definition in Section 3, where the CDR packet latency is the maximum latency of all its member RDR packets (measured in seconds, 'sec' for short). For the AODV scheme, since it does not generate CDR packets, we simply measure its average latency of RDR packets.
- **Event notification latency:** According to the definition in Section 3, when multiple UEN packets are generated to describe the same event, we take the latency of the first UEN packet arriving at the sink node (measured in seconds).
- **Energy consumption:** We evaluate the total energy consumption of each sensor due to the transmission of packets (measured in joules, 'J' for short), which include RDR, CDR, UEN, and control packets required in the AODV/AOMDV protocols.

Remark 8 (Grid-based sensor deployment). We employ the grid-based deployment (instead of random deployment) of sensors in the simulation due to two reasons. First, it is a common and robust deployment manner in many WSN applications [39]. Second, because sensors are static, they can keep the similar number of neighbors in their $\mathcal{N}_i$ sets. Specifically, a sensor has averagely five neighbors in $\mathcal{N}_i$ for choice so that we can easily observe the benefit of using our L2DC scheme. On the contrary, if we just use the random deployment of sensors, then some sensors may have only one neighbor in their $\mathcal{N}_i$ sets. In this case, our L2DC scheme has insignificant effect because these sensors have no other choices to select their next-hop neighbors. That is why we adopt the grid-based sensor deployment in our simulation. Notice that we still require a routing protocol for sensors to send their data to the sink node even in such 'regular' grid-based deployment. This is because the network topology could be changed when the communication links between some sensors become weak or even broken (due to interference or serious packet dropping). Besides, sensors may occasionally alter their routing paths to the sink node if they find some more suitable neighbors to relay the packets (for example, the neighbors with better channel conditions).

Remark 9 (Selection of AODV). Most of the current sensor platforms follow the ZigBee specification [40] to construct the network. In ZigBee, AODV is the basic routing protocol for the mesh topology. Therefore, we build our L2DC scheme on the AODV protocol to evaluate its performance in our simulation.

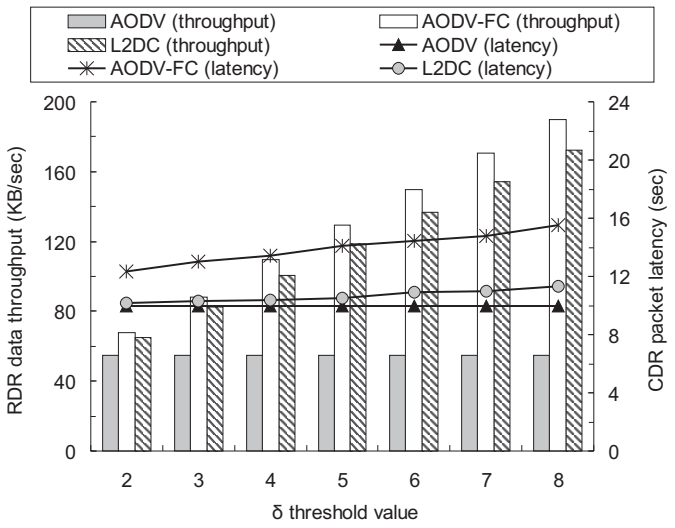
5.1 Effect of Different δ Threshold Values

We first evaluate the effect of different δ threshold values on both the RDR data throughput and the CDR packet latency. In particular, the δ threshold value is gradually increased from 2 to 8. Notice that $\delta = 1$ implies that there is no data compression employed. That is why we start the δ value from 2 instead of 1.

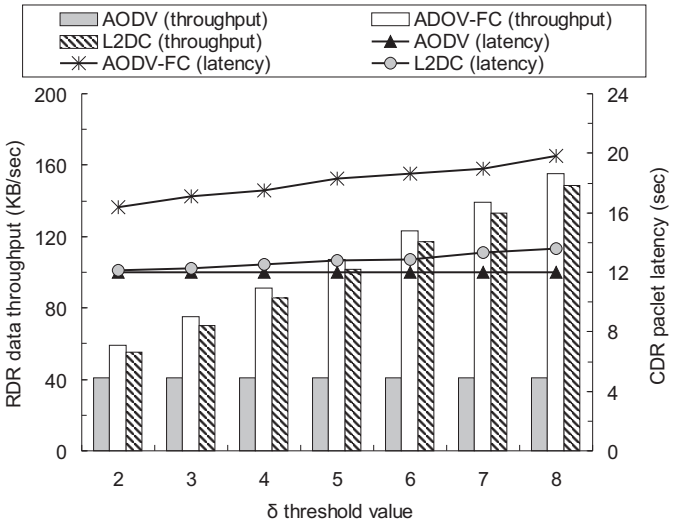
Fig. 4 (a) and (b) present the experimental results of the AODV, AODV-FC, and L2DC schemes under different δ threshold values when the number of events is set to 2 and 5, respectively. Without data compression, the AODV scheme always suffers from the lowest RDR data throughput because of the serious network congestion (since there is a large amount of RDR packet transmission in the network). On the other hand, both AODV-FC and L2DC schemes take advantage of data compression to significantly improve their RDR data throughput. Since the L2DC scheme allows sensors to directly transmit RDR packets (under some network conditions), it would result in certain degree of RDR throughput degradation as compared with the AODV-FC scheme. However, this operation also helps greatly reduce the CDR packet latency in the L2DC scheme. It can be observed that our L2DC scheme incurs slightly higher CDR packet latency compared to the AODV scheme, which implies that our L2DC scheme can alleviate the long packet-latency problem caused by the data compression process.

From Fig. 4, changing the δ threshold value has no effect on the AODV scheme because it does not compress RDR packets. However, increasing the δ threshold value can significantly improve the RDR data throughput in both the AODV-FC and L2DC schemes. In this case, since more RDR packets can be combined into one CDR packet, we can reduce the amount of data traffics transmitted in the network. However, the CDR packet latency will also increase when the δ threshold value grows, because sensors have to wait for longer time to compress RDR packets. Such effect is much obvious in the AODV-FC scheme. In addition, when more events occur in the sensing field, the RDR data throughput decreases while

3. In NS-2, AODV returns only one next-hop neighbor for each sensor. However, our L2DC scheme allows a sensor s_i to choose one neighbor from the set $\mathcal{N}_i$. In order to calculate $\mathcal{N}_i$, we employ the module of the AOMDV (ad hoc on-demand multipath distance vector routing) protocol [38] in our simulation.

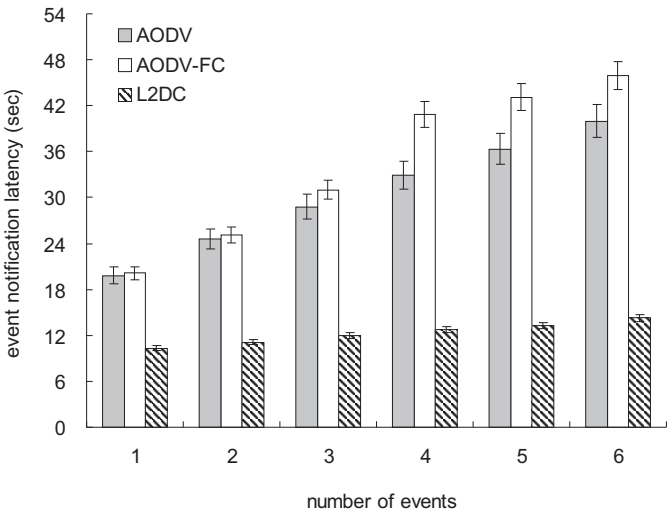


(a) the case of 2 events

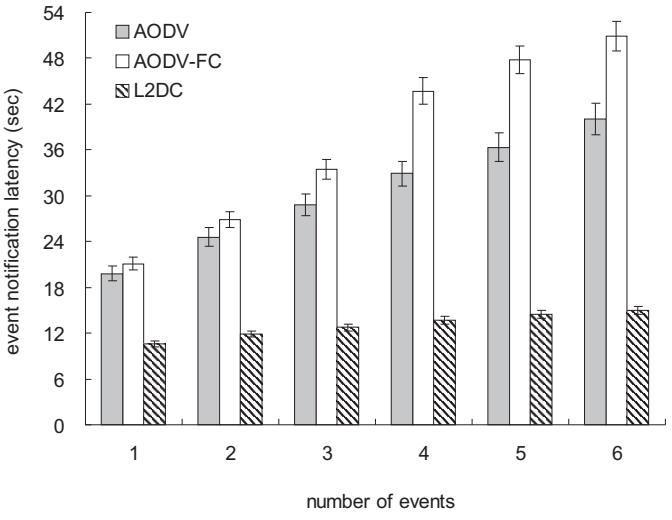


(b) the case of 5 events

Fig. 4: Comparison on the RDR data throughput and the CDR packet latency by the AODV, AODV-FC, and L2DC schemes under different δ threshold values.



(a) the case of $\delta = 4$



(b) the case of $\delta = 8$

Fig. 5: Comparison on the event notification latency by the AODV, AODV-FC, and L2DC schemes under different number of events in the sensing field.

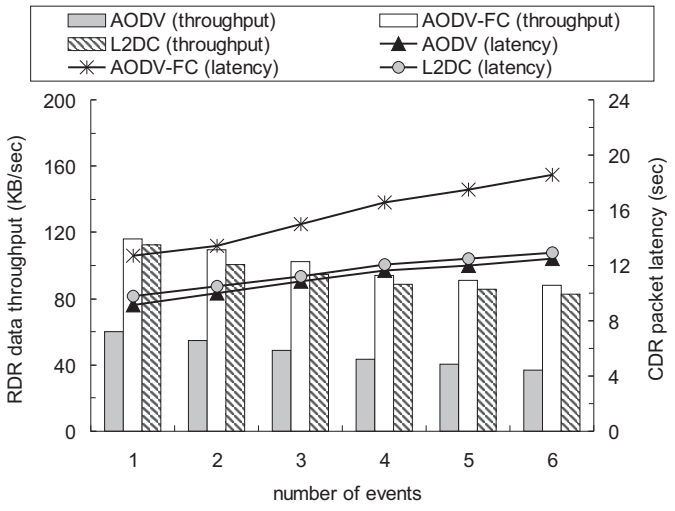
the CDR packet latency increases in all schemes. The reason is that there are more UEN packets competing for the wireless medium. By discarding unnecessary UEN packets, our L2DC scheme will not significantly increase the CDR latency when the number of events grows from 2 to 5 (in other words, the number of sensors that generate UEN packets increases from around 10 to 25).

On the average, our L2DC scheme can reduce about 23.54% and 29.30% of the CDR packet latency when the number of events is 2 and 5, respectively, as compared with the AODV-FC scheme. This experiment verifies that the L2DC scheme can efficiently solve the long packet-latency problem caused by the data compression process.

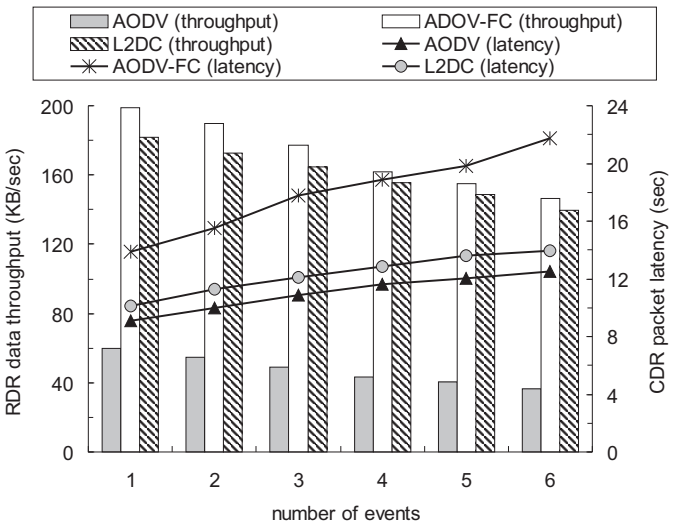
5.2 Effect of Different Number of Events

By increasing the number of events from 1 to 6, we then measure the event notification latency encountered in different schemes. Fig. 5 (a) and (b) show the experimental results when the δ threshold value is set to 4 and 8, respectively. Without differentiating UEN packets with other types of packets, both the AODV and AODV-FC schemes will suffer from significantly larger event notification latency. This is because UEN packets have to compete with a great number of RDR or CDR packets. When there are more sensors generating UEN packets, these UEN packets could congest the network and thereby increase their latency. For the AODV-FC scheme, because UEN packets have to wait to be transmitted if there exist uncompressed RDR packets in front of the queue, it thus has larger event notification latency than the AODV scheme (such effect is more obvious with a larger δ threshold value).

On the contrary, our L2DC scheme not only gives the UEN packets a higher priority to catch their deadlines but also removes redundant UEN packets which describe the same event(s). Therefore, it can slash the event notification latency compared with other two schemes. Besides, the event detection latency slowly increases as the number of events grows. In particular, our L2DC scheme can save averagely 58.36% and 62.12% of event notification latency compared with the AODV and AODV-FC schemes, respectively, when the δ threshold value is set to 4. On the other hand, when the δ threshold value is set to 8, the L2DC scheme respectively reduces 55.81% and 62.71% of event notification latency compared with the AODV and AODV-FC schemes on the average. This experi-

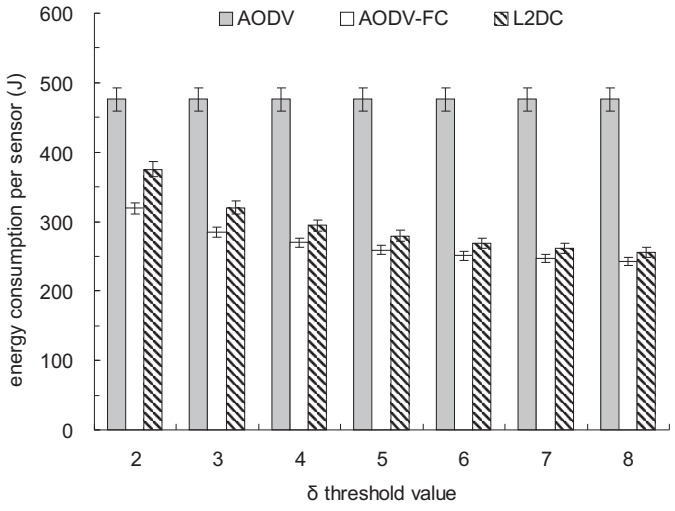


(a) the case of $\delta = 4$

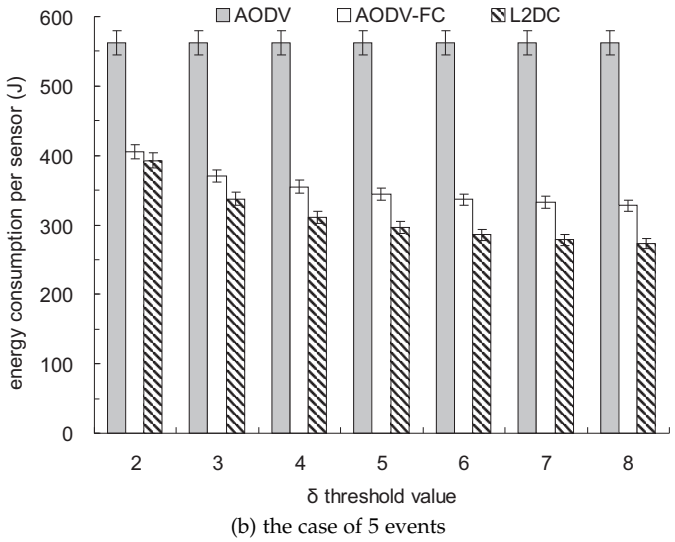


(b) the case of $\delta = 8$

Fig. 6: Comparison on the RDR data throughput and the CDR packet latency by the AODV, AODV-FC, and L2DC schemes under different number of events in the sensing field.



(a) the case of 2 events



(b) the case of 5 events

Fig. 7: Comparison on the energy consumption of each sensor by the AODV, AODV-FC, and L2DC schemes under different δ threshold values.

ment demonstrates the effectiveness of our L2DC scheme in terms of alleviating the event notification latency (and also UEN packet latency).

Following the same simulation parameters, we then investigate the effect of different number of events on both the RDR data throughput and CDR packet latency. Fig. 6 (a) and (b) present the experimental results when the δ threshold value is set to 4 and 8, respectively. Similarly, the AODV scheme always has the lowest RDR data throughput because it does not adopt the data compression technique to support in-network data reduction. On the other hand, the L2DC scheme can significantly reduce the CDR packet latency (as compared with the AODV-FC scheme) at the cost of slight RDR throughput degradation. This result shows that our L2DC scheme can still take advantage of data compression to improve the RDR data throughput.

From Fig. 6, it can be observed that the RDR data throughput decreases while the CDR packet latency increases in all schemes when the number of events increases. In this case, more UEN packets will compete with CDR packets for the wireless medium. On the other hand, because our L2DC scheme allows sensors to discard redundant UEN packets which describe the same event(s), it thus can keep relatively lower CDR packet latency. To summarize, our L2DC scheme can respectively reduce around 26.89% and 29.84% of the CDR packet latency as compared with the AODV-FC scheme when the δ threshold value is set to 4 and 8 on the average.

5.3 Energy Consumption of Sensors

We finally investigate the energy consumption of each sensor in different schemes, where the δ threshold value ranges from 2 to 8. Fig. 7(a) and (b) show the experimental results when there are 2 and 5 events in the sensing field, respectively. Obviously, without data compression, the AODV scheme always lets sensors consume the most amount of energy since they have to continually send out a great number of RDR packets. Besides, changing the δ threshold value has no effect on the AODV scheme. For both the AODV-FC and L2DC schemes, each sensor can save more energy when the δ threshold value increases, because it can compress more RDR packets into one CDR packet for transmission. When there are more events, sensors will consume more energy due to the transmission of UEN packets in all schemes.

From Fig. 7(a), our L2DC scheme consumes more energy than the AODV-FC scheme due to two reasons. First, the L2DC scheme allows sensors to directly send out uncompressed RDR packets to facilitate the compression process, which could increase the amount of packet transmission. Second, in order to construct the $\mathcal{N}_i$ set for each sensor, the L2DC scheme is built on the AOMDV protocol, which requires more control packets than the AODV protocol. On the contrary, our L2DC scheme saves more energy than the AODV-FC scheme in Fig. 7(b). In this case, there are 5 events occurring every 60 seconds, so around 25 sensors will periodically generate a large number of UEN packets. In this case, the L2DC scheme can take advantage of event detection regions to discard many redundant UEN packets, thereby avoiding unnecessary packet transmission. In brief, with the help of the packet elimination mechanism in Section 4.1, our L2DC scheme can further save sensors' energy than the AODV-FC scheme when more events occur in the sensing field.

6 Conclusions

In this paper, we point out the long packet-latency problem caused by the data compression process and the bursty network-congestion problem due to numerous redundant UEN packets. By taking the computation and resource limitation of sensors into consideration, a lightweight but efficient L2DC scheme is developed to solve both problems. L2DC not only discards unnecessary UEN packets using a simple rule but also allows sensors to transmit either RDR or CDR packets to facilitate the data compression process. Through the simulation in NS-2, we demonstrate the effectiveness of our L2DC scheme in terms of reducing both the CDR packet latency and the event notification latency. In addition, the L2DC scheme can significantly reduce energy consumption of sensors when there are more events occurring in the sensing field.

References

[1] Y.C. Wang, F.J. Wu, and Y.C. Tseng, "Mobility management algorithms and applications for mobile sensor networks," *Wireless Comm. and Mobile Computing*, vol. 12, no. 1, pp. 7–21, 2012.

[2] C. Zhu, L. Shu, T. Hara, L. Wang, S. Nishio, and L.T. Yang, "A survey on communication and data management issues in mobile sensor networks," *Wireless Comm. and Mobile Computing*, vol. 14, no. 1, pp. 19–36, 2014.

[3] Y.C. Wang, "Mobile sensor networks: system hardware and dispatch software," *ACM Computing Surveys*, vol. 47, no. 1, pp. 12:1–12:36, 2014.

[4] Y.C. Wang, Y.Y. Hsieh, and Y.C. Tseng, "Compression and storage schemes in a sensor network with spatial and temporal coding techniques," *Proc. IEEE Vehicular Technology Conf.*, 2008, pp. 148–152.

[5] P.J. Wan and C.W. Yi, "Coverage by randomly deployed wireless sensor networks," *IEEE Trans. Information Theory*, vol. 52, no. 6, pp. 2658–2669, 2006.

[6] Y.C. Wang and Y.C. Tseng, "Distributed deployment schemes for mobile wireless sensor networks to ensure multilevel coverage," *IEEE Trans. Parallel and Distributed Systems*, vol. 19, no. 9, pp. 1280–1294, 2008.

[7] X. Mao, X. Xu, S. Tang, and X.Y. Li, "Providing and finding k-road-coverage efficiently in wireless sensor networks," *Wireless Comm. and Mobile Computing*, vol. 12, no. 12, pp. 1053–1065, 2012.

[8] C.E. Perkins and E.M. Royer, "Ad-hoc on-demand distance vector routing," *Proc. IEEE Workshop on Mobile Computing Systems and Applications*, 1999, pp. 90–100.

[9] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Comm. Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[10] Y.C. Wang, "Data compression techniques in wireless sensor networks," in *Pervasive Computing*, USA: Nova Science Publishers, 2012.

[11] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Trans. Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[12] O. Younis and S. Fahmy, "HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Trans. Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.

[13] G.A. Shah, M. Bozyigit, and F.B. Hussain, "Cluster-based coordination and routing framework for wireless sensor and actor networks," *Wireless Comm. and Mobile Computing*, vol. 11, no. 8, pp. 1140–1154, 2011.

[14] C. Hua and T.S.P. Yum, "Optimal routing and data aggregation for maximizing lifetime of wireless sensor networks," *IEEE/ACM Trans. Networking*, vol. 16, no. 4, pp. 892–903, 2008.

[15] M.O. Diaz and K.K. Leung, "Efficient data aggregation and transport in wireless sensor networks," *Wireless Comm. and Mobile Computing*, vol. 11, no. 8, pp. 1030–1041, 2011.

[16] L.A. Villas, A. Boukerche, H.S. Ramos, H.A.B.F. de Oliveira, R.B. de Araujo, and A.A.F. Loureiro, "DRINA: a lightweight and reliable routing approach for in-network aggregation in wireless sensor networks," *IEEE Trans. Computers*, vol. 62, no. 4, pp. 676–689, 2013.

[17] S. Lindsey, C. Raghavendra, and K.M. Sivalingam, "Data gathering algorithms in sensor networks using energy metrics," *IEEE Trans. Parallel and Distributed Systems*, vol. 13, no. 9, pp. 924–935, 2002.

[18] Y.C. Wang, C.H. Chuang, Y.C. Tseng, and C.C. Shen, "A lightweight, self-adaptive lock gate designation scheme for data collection in long-thin wireless sensor networks," *Wireless Comm. and Mobile Computing*, vol. 13, no. 1, pp. 47–62, 2013.

[19] C.M. Sadler and M. Martonosi, "Data compression algorithms for energy-constrained devices in delay tolerant networks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems*, 2006, pp. 265–278.

[20] F. Marcelloni and M. Vecchio, "A simple algorithm for data compression in wireless sensor networks," *IEEE Comm. Letters*, vol. 12, no. 6, pp. 411–413, 2008.

[21] D. Ganesan, B. Greenstein, D. Estrin, J. Heidemann, and R. Govindan, "Multiresolution storage and search in sensor networks," *ACM Trans. Storage*, vol. 1, no. 3, pp. 277–315, 2005.

[22] Y.C. Wang, Y.Y. Hsieh, and Y.C. Tseng, "Multiresolution spatial and temporal coding in a wireless sensor network for long-term monitoring applications," *IEEE Trans. Computers*, vol. 58, no. 6, pp. 827–838, 2009.

[23] D. Slepian and J.K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Information Theory*, vol. 19, no. 4, pp. 471–480, 1973.

[24] Z. Xiong, A.D. Liveris, and S. Cheng, "Distributed source coding for sensor networks," *IEEE Signal Processing Magazine*, vol. 21, no. 5, pp. 80–94, 2004.

[25] J. Zheng, P. Wang, and C. Li, "Distributed data aggregation using Slepian-Wolf coding in cluster-based wireless sensor networks," *IEEE Trans. Vehicular Technology*, vol. 59, no. 5, pp. 2564–2574, 2010.

[26] T. Xue, X. Dong, and Y. Shi, "Multiple access and data reconstruction in wireless sensor networks based on compressed sensing," *IEEE Trans. Wireless Communications*, vol. 12, no. 7, pp. 3399–3411, 2013.

[27] R. Xie and X. Jia, "Transmission-efficient clustering method for wireless sensor networks using compressive sensing," *IEEE Trans. Parallel and Distributed Systems*, vol. 25, no. 3, pp. 806–815, 2014.

[28] E. Felemban, C.G. Lee, and E. Ekici, "MMSPEED: multipath multi-speed protocol for QoS guarantee of reliability and timeliness in wireless sensor networks," *IEEE Trans. Mobile Computing*, vol. 5, no. 6, pp. 738–754, 2006.

[29] M. Razzaque, M.M. Alam, M. Mamun-Or-Rashid, and C.S. Hong, "Multi-constrained QoS geographic routing for heterogeneous traffic in sensor networks," *IEICE Trans. Comm.*, vol. 91B, no. 8, pp. 2589–2601, 2008.

[30] D. Djenouri and I. Balasingham, "Traffic-differentiation-based modular QoS localized routing for wireless sensor networks," *IEEE Trans. Mobile Computing*, vol. 10, no. 6, pp. 797–809, 2011.

[31] J. Zhang, F. Ren, S. Gao, H. Yang, and C. Lin, "Dynamic routing for data integrity and delay differentiated services in wireless sensor networks," *IEEE Trans. Mobile Computing*, vol. 14, no. 2, pp. 328–343, 2015.

[32] L. Karim, N. Nasser, T. Taleb, and A. Alqallaf, "An efficient priority packet scheduling algorithm for wireless sensor network," *Proc. IEEE Int'l Conf. Comm.*, 2012, pp. 334–338.

[33] J. Hu, G. Min, and M.E. Woodward, "Performance analysis and comparison of burst transmission schemes in unsaturated 802.11e WLANs," *Wireless Comm. and Mobile Computing*, vol. 12, no. 9, pp. 837–848, 2012.

[34] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE J. Selected Areas in Comm.*, vol. 18, no. 3, pp. 535–547, 2000.
[35] F. Wang, D. Li, and Y. Zhao, "Analysis of CSMA/CA in IEEE 802.15.4," *IET Comm.*, vol. 5, no. 15, pp. 2187–2195, 2011.
[36] K. Schwieger, H. Nuszkowski, and G. Fettweis, "Analysis of node energy consumption in sensor networks," *Wireless Sensor Networks*, vol. 2920, pp. 94–105, 2004.
[37] The Network Simulator (NS2). [Online]. Available: http://www.isi.edu/nsnam/ns/
[38] M.K. Marina and S.R. Das, "Ad hoc on-demand multipath distance vector routing," *Wireless Comm. and Mobile Computing*, vol. 6, no. 7, pp. 969–988, 2006.
[39] K. Xu, G. Takahara, and H. Hassanein, "On the robustness of grid-based deployment in wireless sensor networks," *Proc. ACM Int'l Conf. Wireless Comm. and Mobile Computing*, 2006, pp. 1183–1188.
[40] ZigBee Alliance. ZigBee Specification. [Online]. Available: http://www.zigbee.org/