Dynamic Routing for Data Integrity and Delay Differentiated Services in Wireless Sensor Networks

Jiao Zhang, *Member, IEEE*, Fengyuan Ren, *Member, IEEE*, Shan Gao, Hongkun Yang, and Chuang Lin, *Senior Member, IEEE*

Abstract—Applications running on the same Wireless Sensor Network (WSN) platform usually have different Quality of Service (QoS) requirements. Two basic requirements are low delay and high data integrity. However, in most situations, these two requirements cannot be satisfied simultaneously. In this paper, based on the concept of *potential* in physics, we propose IDDR, a multi-path dynamic routing algorithm, to resolve this conflict. By constructing a virtual hybrid potential field, IDDR separates packets of applications with different QoS requirements according to the weight assigned to each packet, and routes them towards the sink through different paths to improve the data fidelity for integrity-sensitive applications as well as reduce the end-to-end delay for delay-sensitive ones. Using the Lyapunov drift technique, we prove that IDDR is stable. Simulation results demonstrate that IDDR provides data integrity and delay differentiated services.

Index Terms—Wireless sensor networks, data integrity, delay differentiated services, dynamic routing, potential field

1 Introduction

WSNs, which are used to sense the physical world, will play an important role in the next generation networks. Due to the diversity and complexity of applications running over WSNs, the QoS guarantee in such networks gains increasing attention in the research community.

As a part of an information infrastructure, WSNs should be able to support various applications over the same platform. Different applications might have different QoS requirements. For instance, in a fire monitoring application, the event of a fire alarm should be reported to the sink as soon as possible. On the other hand, some applications require most of their packets to successfully arrive at the sink irrespective of when they arrive. For example, in habitat monitoring applications, the arrival of packets is allowed to have a delay, but the sink should receive most of the packets.

WSNs have two basic QoS requirements: low delay and high data integrity, leading to what are called *delay-sensitive applications* and *high-integrity applications*, respectively. Generally, in a network with light load, both requirements can be readily satisfied. However, a heavily loaded network will suffer congestion, which increases the end-to-end delay.

This work aims to simultaneously improve the fidelity for high-integrity applications and decrease the end-to-end delay for delay-sensitive ones, even when the network is congested. We borrow the concept of *potential field* from the discipline of physics and design a novel potential-based routing algorithm, which is called integrity and delay differentiated routing (IDDR). IDDR is able to provide the following two functions:

- *Improve fidelity for high-integrity applications*. The basic idea is to find as much buffer space as possible from the idle and/or under-loaded paths to cache the excessive packets that might be dropped on the shortest path. Therefore, the first task is to find these idle and/or underloaded paths, then the second task is to cache the packets efficiently for subsequent transmission. IDDR constructs a potential field according to the depth[1] and queue length information to find the under-utilized paths. The packets with high integrity requirement will be forwarded to the next hop with smaller queue length. A mechanism called *Implicit Hop-by-Hop Rate Control* is designed to make packet caching more efficient.
- *Decrease end-to-end delay for delay-sensitive applications.* Each application is assigned a weight, which represents the degree of sensitivity to the delay. Through building local dynamic potential fields with different slopes according to the weight values carried by packets, IDDR allows the packets with larger weight to choose shorter paths. In addition, IDDR also employs the priority queue to further decrease the queuing delay of delay-sensitive packets.

- *J. Zhang is with the School of Information and Communication Engineering, BUPT and State Key Laboratory of Networking and Switching Technology, BUPT, China. E-mail: jiaozhang@bupt.edu.cn.*
- *F. Ren, S. Gao, and C. Lin are with the Department of Computer Science and Technology, Tsinghua University, Beijing, China and Tsinghua National Laboratory for Information Science and Technology, China. E-mail: {renfy, gaoshan, clin}@csnet1.cs.tsinghua.edu.cn.*
- *H. Yang is with the Department of Computer Science, University of Texas at Austin, Texas. E-mail: yang.hongk@gmail.com.*

Manuscript received 19 Aug. 2013; revised 23 Dec. 2013; accepted 18 Mar. 2014. Date of publication 26 Mar. 2014; date of current version 23 Dec. 2014. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TMC.2014.2313576

1. In this paper, *depth* of a node is defined as the least hops that the node is away from the sink.

IDDR inherently avoids the conflict between *high integrity* and *low delay*: the high-integrity packets are cached on the underloaded paths along which packets will suffer a large end-to-end delay because of more hops, and the delay-sensitive packets travel along shorter paths to approach the sink as soon as possible. Using the Lyapunov drift theory, we prove that IDDR is stable. Furthermore, the results of a series of simulations conducted on the TOSSIM platform [1] demonstrate the efficiency and feasibility of the IDDR scheme.

The remainder of the paper is organized as follows. Section 2 introduces the related work and motivation. The details of IDDR are described in Section 3. Section 4 proves the stability of IDDR. The performance of IDDR is evaluated through experiments on a small testbed and simulations on the TOSSIM platform in Sections 5 and 6, respectively. Finally, the conclusions are drawn in Section 7.

2 Related Work and Motivation

2.1 Related Work

Most QoS provisioning protocols proposed for traditional ad hoc networks have large overhead caused by end-to-end path discovery and resource reservation [2], [3], [4], [5], [6]. Thus, they are not suitable for resource-constrained WSNs. Some mechanisms have been designed to provide QoS services specifically for WSNs. Here we mainly focus on the metrics of delay and reliability.

2.1.1 *Providing Real-Time Service*

RAP exploits the notion of velocity and proposes a velocity-monotonic scheduling policy to minimize the ratio of missed deadlines [7]. However, the global information of network topology is required. Implicit Earliest Deadline First (EDF) mainly utilizes a medium access control protocol to provide real-time service [8]. The implicit prioritization is used instead of relying on control packets as most other protocols do. SPEED maintains a desired delivery speed across the network through a novel combination of feedback control and non-deterministic QoS-aware geographic forwarding [9]. In [10], a two-hop neighbor information-based gradient routing mechanism is proposed to enhance real-time performance. The routing decision is made based on the number of hops from a source to the sink and the two-hop information.

2.1.2 *Providing Reliability Service*

Adaptive Forwarding Scheme (AFS) employs the packet priority to determine the forwarding behavior to control the reliability [11]. ReInforM uses the concept of dynamic packet states to control the number of paths required for the desired reliability [12]. However, both of AFS and ReInforM require to know the global network topology. LIEMRO [13] utilizes a dynamic path maintenance mechanism to monitor the quality of the active paths during network operation and regulates the injected traffic rate of the paths according to the latest perceived paths quality. However, it does not consider the effects of buffer capacity and service rate of the active nodes to estimate and adjust the traffic rate of the active paths.

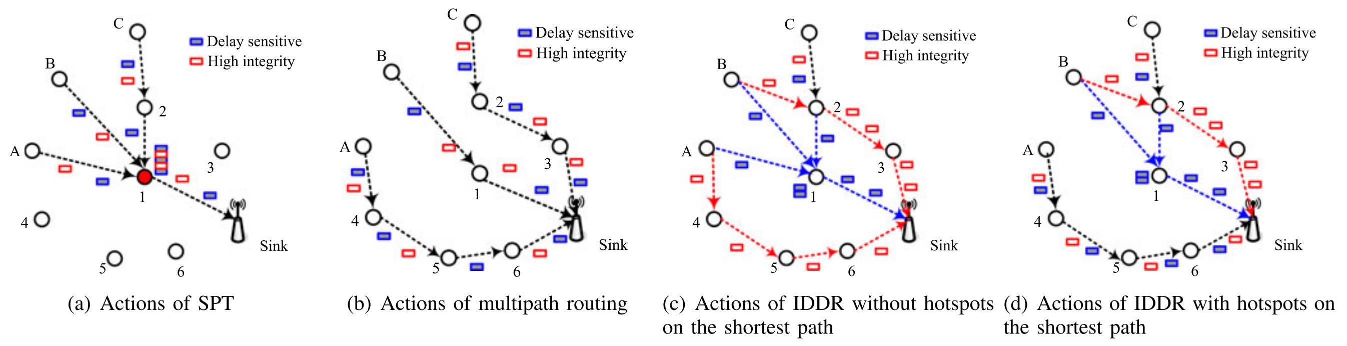
2.1.3 *Providing Real-Time and Reliability Services*

MMSPEED extends SPEED for service differentiation and probabilistic QoS guarantee [6]. It uses the same mechanism as SPEED to satisfy the delay requirements for different types of traffic, and uses redundant paths to ensure reliability. The MAC layer function is modified to provide prioritized access and reliable multicast delivery of packets to multiple neighbors. However, when the network is congested, all the source nodes still continuously transmit packets to the sink along multipaths without taking some other mechanisms, such as caching packets for some time. This not only deteriorates reliability but also retards the delay-sensitive packets. Energy-Efficient and QoS-based Multipath Routing Protocol (EQSR) [14] improves reliability through using a lightweight XOR-based Forward Error Correction (FEC) mechanism, which introduces data redundancy in the data transmission process. Furthermore, in order to meet the delay requirements of various applications, EQSR employs a queuing model to manage real-time and non-real-time traffic. DARA [15] considers reliability, delay and residual energy. But it only differentiates the applications into two classes: critical and non-critical. The neighbor sets of a node for the two kinds of applications are different and all the packets belonging to the same category will be forwarded to the next hop computed by the same function. Obviously, two classifications of the applications in WSNs are not enough. D. Djenouri and Balasingham proposed LOCALMOR, which considers latency, reliability and energy [16]. It puts the incoming packets into three queues according to their requirements. LOCALMOR satisfies the requirement of reliability-sensitive applications by transmitting the data to both the primary sink and the secondary sink, which incurs much overhead. What's more, it combines the queue management mechanism and routing to provide differentiated services.

How to design a routing protocol that provides data integrity and delay differentiated services over the same WSN simultaneously without incurring much overhead is an extremely challenging problem. The main contribution of this paper is to borrow the concept of the *potential* field from physics and design a novel potential-based dynamic routing algorithm, IDDR, which can provide high integrity and delay-differentiated services using only local information.

2.2 Motivation

Fig. 1 illustrates a small part of a WSN. Suppose node 1 is a hotspot and there are both *high-integrity packets* (hollow rectangles) and *delay-sensitive packets* (solid rectangles) from source nodes A, B and C. A commonly used routing algorithm will choose the optimal path for all the packets. For example, the standard shortest path tree (SPT) routing will forward all of them to node 1 as shown in Fig. 1a. This will cause congestion and thus lead to many *high-integrity packets* loss and large end-to-end delay for *delay-sensitive packets*. A multipath routing algorithm as shown in Fig. 1b can utilize more paths to avoid hotspots. However, the low delay and high throughput are hardly met simultaneously. The reasons are:



(a) Actions of SPT (b) Actions of multipath routing (c) Actions of IDDR without hotspots on the shortest path (d) Actions of IDDR with hotspots on the shortest path

Fig. 1. Motivation of IDDR.

- Delay-sensitive packets occupy the limited bandwidth and buffers, worsening drops of high-integrity ones.
- High-integrity packets block the shortest paths, compelling the delay-sensitive packets to travel more hops before reaching the sink, which increases the delay.
- High-integrity packets occupy the buffers, which also increases the queuing delay of delay-sensitive packets.

To overcome the above drawbacks, we intend to design a mechanism which allows the delay-sensitive packets to move along the shortest path and the packets with fidelity requirements to detour to avoid possible dropping on the hotspots. In this way, the data integrity and delay differentiated services can be provided in the same network. Motivated by this understanding, we propose the IDDR scheme, a potential-based multi-path dynamic routing algorithm.

As shown in Fig. 1c, the *high-integrity packets* do not choose node 1 due to its large queue length. Some other idle and/or underloaded paths, such as path $2 \rightarrow 3 \rightarrow Sink$ and $4 \rightarrow 5 \rightarrow 6 \rightarrow Sink$, are used to cache and route these packets efficiently so as to protect them from being dropped in the hotspot. On the other hand, IDDR gives *delay-sensitive packets* priority to go ahead in the shortest path to achieve low delay. Furthermore, if the traffic on the shortest path is heavy, IDDR can also select other paths for the delay-sensitive packets, such as path: $A \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow Sink$ shown in Fig. 1d, the link from node 1 to the sink is so busy that node A or B will bypass node 1 and send packets to the sink along other under-utilized paths to avoid packets being dropped.

IDDR distinguishes different types of packets using the weight values inserted into the header of packets, and then performs different actions on them. Its cornerstone is to construct proper potential fields to make right routing decisions for different types of packets. Next the potential-based IDDR algorithm will be described in detail.

3 Details on IDDR

We first describe the potential fields on which IDDR is based. Then we present how the potential fields improve the data fidelity and decrease the end-to-end delay of packets.

3.1 Design of Potential Fields

A potential-based routing paradigm has been designed for traditional wireline networks [17]. However, it did not attract widespread attention because of its huge management overhead. It is quite expensive to build an exclusive virtual field for each destination in traditional networks where numerous destinations might be distributed arbitrarily. On the contrary, the potential-based routing algorithm is much suitable for the many-to-one traffic pattern in WSNs. In some special applications and environments, more than one sink may exist. However, generally the data-centric WSNs only require nodes to transmit their sampling data to one of them. Therefore, in this work, we build a unique virtual potential field to customize a multipath dynamic routing algorithm, which finds proper paths to the sink for the packets with high integrity and delay requirements. Next, the potential-based routing algorithm for WSNs with one sink is described. It is straightforward to extend the algorithm to work in WSNs with multiple sinks. In Section 3.4.3, we will introduce this extension in detail.

Fig. 2 depicts a general potential field whose shape looks like a *bowl*. All data packets are transmitted to the bottom along the surface like water. In WSNs with light traffic, IDDR works similar to the shortest path routing algorithm. But in WSNs with heavy load, large backlogs will form some bulges on the bowl surface. The bulges will block the paths and prevent packets from moving down to the bottom directly.

3.1.1 Potential Field Model

In the bowl model shown in Fig. 2, we can view the whole network as a gravity field. A packet can be viewed as a drop of water, moving down to the bottom along the surface of the bowl. The trajectory of this packet is determined by the force from the potential field.

A single-valued potential $V_v(t)$ is assigned to node v on the bowl surface at time t to form a scalar potential field. Let $\Omega_v(t)$ be the neighbor set of node v during time t. Consider a packet p at node v, to reach the sink, p will be forwarded to a neighbor $w \in \Omega_v(t)$. We define a force acting on the packet p at node v based on the potential difference between node v and node w at time t as follows:

$$F_{v \rightarrow w}(t) = V_v(t) - V_w(t). \tag{1}$$

The packet will be forwarded to the neighbor x at time t for which the force $F_{v \rightarrow x}(t)$ is the maximum, namely, the neighbor in the direction of the steepest gradient. If the surface is smooth, the packet will move straightly down to the bottom,

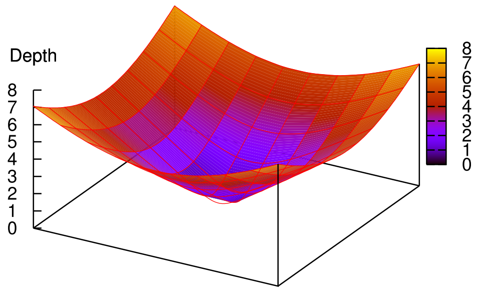


Fig. 2. The smooth "bowl" of depth potential field.

but if the surface is somewhat rough, the packet will move along an irregular curve which is formed by a series of *valleys*.

3.1.2 Depth Potential Field

To provide the basic routing function, i.e., to make each packet move towards the sink, the proposed IDDR algorithm defines a depth potential field $V_v^d(t) = D_v(t)$, where $D_v(t)$ is the depth of node v at time t. Thus the depth field force $F_{v\to w}^d(t)$ from node v to its neighbor $w \in \Omega_v(t)$ is

$$F_{v\to w}^d(t) = D_v(t) - D_w(t). \tag{2}$$

The depth difference $(D_v(t) - D_w(t)) \in \{-1, 0, 1\}$ since two nodes with more than one hops distance cannot become neighbors.

3.1.3 Queue Length Field

Define the queue length potential field of node v at time t as $V_v^q(t) = Q_v(t)$, where $Q_v(t)$ denotes the queue length normalized to the buffer size of node v at t. Then the queue length potential force $F_{v\to w}^q(t)$ from node v to $w \in \Omega_v(t)$ at time t is

$$F_{v\to w}^q(t) = Q_v(t) - Q_w(t). \tag{3}$$

The range of $Q_v(t)$ is $[0, 1]$, hence we get $F_{v\to w}^q(t) \in [-1, 1]$.

Note that the less is the backlog in the queue at node w, the larger is the potential force. Hence, driven by this queue potential field, packets will always be forwarded towards the underloaded areas, bypassing the hotspots.

3.1.4 Hybrid Potential Field

We construct a virtual hybrid potential on the basis of the depth and queue length potential fields defined above. The two independent fields are linearly combined together: $V_v^m(t) = \alpha V_v^d(t) + V_v^q(t)$, where $V_v^m(t)$ is the potential of the mixed field at node v, and $\alpha > 0$. Then the mixed force from node v to one of its neighbor $w \in \Omega_v(t)$

$$F_{v\to w}^m(t) = (Q_v(t) + \alpha D_v(t)) - (Q_w(t) + \alpha D_w(t)). \tag{4}$$

Note that if $\alpha = 0$, then only the queue potential field works, which cannot ensure that the packets generated by sensors will be transmitted to the sink at last. Hence we let $\alpha > 0$.

In the next two subsections, we will illustrate how the potential field and steepest gradient method improve fidelity and decrease delay.

3.2 High-Integrity Services

How to provide *high-integrity services* for applications? The basic idea of IDDR is to consider the whole network as a big buffer to cache the excessive packets before they arrive at the sink. There are two key steps: (1) Finding enough buffer spaces from the idle or underloaded nodes, which is actually *resource discovery*. (2) Caching the excessive packets in these idle buffers efficiently for subsequent transmissions, which implies an *implicit hop-by-hop rate control*.

3.2.1 Resource Discovery

In a under-utilized WSN, the queue length is very small, the hybrid potential field is governed by the depth potential field. IDDR performs like the shortest path algorithm, that is, a node always chooses one neighbor with lower depth as its next hop. However, in a over-utilized WSN, the shortest paths are likely be full of packets. Therefore, new coming packets will be driven out of the shortest paths to find other available resource. If a node knows the queue length information of its neighbors, it can forward packets to the underloaded neighbors to stand against possible dropping. The following two propositions explain how IDDR reaches this goal.

Proposition 1. *Denote the depth of node v as d. Let S denote the neighbors of node v with the same depth, that is, $S = \{x | D_x(t) = d, x \in \Omega_v(t)\}$ and L denote the neighbors of node v with smaller depth, that is, $L = \{x | D_x(t) = d - 1, x \in \Omega_v(t)\}$. Let $l \in L$ be the node with the minimal queue length in L. If node $s \in S$ has the minimum queue length in S and satisfies that $Q_s(t) < Q_l(t) - \alpha$, then node v will choose node s rather than node l as the next hop at time t.*

Proof. If node v does not choose node l as its parent, it will not choose any other nodes in L since node l has the minimal queue length and all the nodes in L have the same depth. The potential values at nodes, v, l and s, are:

$$V_v^m(t) = \alpha d + Q_v(t), \tag{5}$$

$$V_l^m(t) = \alpha(d-1) + Q_l(t), \tag{6}$$

$$V_s^m(t) = \alpha d + Q_s(t). \tag{7}$$

We can derive the force values at node v as follows:

$$F_{v\to l}^m(t) = \alpha + (Q_v(t) - Q_l(t)), \tag{8}$$

$$F_{v\to s}^m(t) = (Q_v(t) - Q_s(t)). \tag{9}$$

On the other hand, we can rewrite $Q_s(t) < Q_l(t) - \alpha$ as

$$(Q_v(t) - Q_s(t)) > \alpha + (Q_v(t) - Q_l(t)). \tag{10}$$

Hence, we can readily have $F_{v\to s}^m(t) > F_{v\to l}^m(t)$. According to the potential field model, node v will choose s as its next hop rather than l, which means that the packets from node v will be forwarded to the neighbors at the same depth since they have more available buffer space to cache packets. □

Remarks. Proposition 1 describes the tradeoff between the path length and the queue length. If the lightest node $l \in L$ has a small queue length, then packets will be forwarded to it. However, if the queue length of node l is large, sending packets to it will likely cause packets dropping and thus severely deteriorate the performance of the high-integrity applications. In this situation, it is better to choose a node with smaller queue length even with the same depth, such as node v, as the next hop.

Then how to quantify the queue length difference between the nodes with smaller depth and that with the same depth. Denote $\Delta Q = Q_l(t) - Q_s(t)$. Inequality $Q_l(t) - Q_s(t) > \alpha$ can be rewritten as $\alpha < \Delta Q$, which implies that α is actually the threshold of the queue length difference between node l and node s that node v begins to send packets to the neighbor with the same depth rather than that with lower depth. Since the normalized queue length difference is not larger than 1, α should be smaller than 1 to ensure that a node can send packets to the under-utilized nodes with the same depth when all the nodes with smaller depth are congested. The smaller is the α, the larger is the possibility that a node selects a neighbor with the same depth instead of with smaller depth as the next hop.

Proposition 1 confirms that IDDR allows a node to choose its next hop from the neighbors with the same depth.

Proposition 2. *Denote the depth of node v as d. Let $H = \{x|D_x(t) = d+1, x \in \Omega_v(t)\}, S = \{x|D_x(t) = d, x \in \Omega_v(t)\}$ and $L = \{x|D_x(t) = d-1, x \in \Omega_v(t)\}$. Node $s \in S$ has the minimal queue length in S, and node $l \in L$ has the minimal queue length in L. If h has the minimum queue length in H and node h satisfies $Q_h(t) < Q_s(t) - \alpha$ as well as $Q_h(t) < Q_l(t) - 2\alpha$, then node v will choose node h rather than node s or l as the next hop at time t.*

Proof. Similar to the proof of Proposition 1. □

Remarks. Proposition 2 describes when IDDR chooses the next hop from the neighbors with higher depth, namely, forwards packets backwards. If there is not enough buffers in the neighbors with the same and lower depth, then packets will be sent to the node with the higher depth to avoid congestion.

Rewrite $Q_h(t) < Q_l(t) - 2\alpha$ as $Q_l(t) - Q_h(t) > 2\alpha$. Since the maximum value of $(Q_l(t) - Q_h(t))$ is 1, we can obtain that only when $\alpha \leq 0.5$, the IDDR algorithm can allow a node to choose its next hop from the neighbors with higher depth.

Combining the above two Propositions, we can conclude that nodes can only forward packets along the shortest path when $\alpha > 1$, when $0.5 < \alpha \leq 1$, packets will possibly be transmitted to the neighbors with the same depth, and when $0 < \alpha \leq 0.5$, packets can be transmitted to the neighbors with the same or even higher depth to improve the data integrality.

3.2.2 Implicit Hop-by-Hop Rate Control

Once detecting a hotspot, if no optimal path, e.g., the shortest path, exists, IDDR will send packets along a suboptimal path. However, if all the neighbors of node v are congested, which is likely to happen near the sink due to the many-to-one traffic pattern in WSNs, node v should cache the arrived packets before some neighbors have available buffer space. Actually, this process is equivalent to a hop-by-hop rate control, which is opposite to the end-to-end flow control of TCP and the sink-source rate control in WSNs [18]. The IDDR uses a simple rule described below to ensure that it can efficiently cache the excessive packets that are to be sent to the hotspots.

Rule 1. For node v, if $Q_w(t) = 1$, where $w \in \Omega_v(t)$, then w should not be selected as the parent in any case at time t.

Thus, the excessive packets will be cached in the local node rather than be dropped at successive nodes. Additionally, Rule 1 also implies an implicit source rate control. If all the paths between a source node and the sink is full of cached packets, this source node will be compelled to slow down.

In a word, combining the queue length potential field with Rule 1, the traffic-aware IDDR mechanism can spatially and temporally spread packets in a reasonable pattern to improve the overall throughput so as to meet the fidelity requirements of *high-integrity applications*.

3.3 Delay-Differentiated Services

There are mainly four factors that affect the end-to-end delay in WSNs: (1) Transmission delay. It is limited by the link bandwidth; (2) Competition of the radio channel. Especially under a contention based MAC, a packet has to compete for the access of the channel and wait for transmission until the channel is idle; (3) Queuing delay. A large queue will seriously delay packets; (4) Path length. Generally, the more hops a packet travels, the large propagation delay it will suffer. The physical limitation determines the transmission delay, and the MAC affects the competition of the radio channel. They are both beyond the scope of this paper. The IDDR aims to decrease the queuing delay and shorten the path length for delay sensitive packets.

Before describing how IDDR provides the delay-differentiated services, we first observe some interesting properties of the hybrid potential field. Then, we propose two effective mechanisms to decrease the end-to-end delay of delay-sensitive packets.

3.3.1 Slope of the Hybrid Potential Field

From Eq. (4), we know that parameter α is actually the slope of the depth potential field:

$$\frac{\partial F^m_{v \to w}(t)}{\partial d} = \alpha, \tag{11}$$

where $d = D_v(t)$. Thus α significantly influences the choice of the next hop. We now use an example to illustrate the behavior of a packet in the potential fields with different values of α. Fig. 3 shows the longitudinal section of two depth fields with $\alpha = 1.0$ and $\alpha = 0.5$. The normalized queue length of node x with depth $d = 3$ is 0.6, and node y with depth $d = 4$ has an empty queue. For a packet at node y in the potential field with $\alpha = 1.0$, it will directly be forwarded to node x because the potential difference between node y and node x is $0.4 = (4.0 - 3.6)$. However,

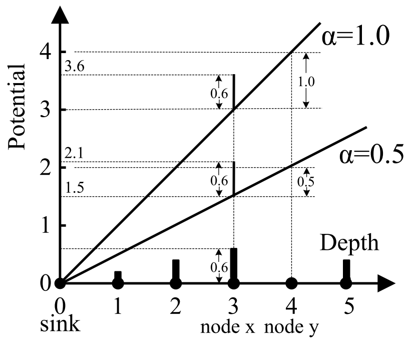


Fig. 3. The Slope of the potential field. The solid line represents the longitudinal section of depth potential field; the solid dots on the horizontal axis are nodes; the short vertical bars over the dots denote the normalized queue length of these nodes.

in the potential field with $\alpha = 0.5$, the packet at node y cannot be directly forwarded to node x because the potential value (2.0) of node y with depth $d = 4$ is lower than that (2.1) of node x with depth $d = 3$. Thus, we can see that packets will be more easily driven out of the shortest paths with smaller α. If we can use different α values to build different depth fields, packets moving in these different fields will gain different services. IDDR uses this interesting property to achieve different end-to-end delay for different type of packets. Next we will present how to obtain these different α values.

3.3.2 Packet Weight

Each packet header contains a 8-bit weight to represent the level of delay sensitivity. The larger is the weight, the more delay-sensitive is the packet. IDDR uses the weight values to build different potential fields with different slopes as follows:

$$\alpha' = \alpha + \frac{\text{Packet Weight}}{0xff}, \tag{12}$$

where $0xff$ is the maximum of the 8-bit weight. A larger α' value means higher weight of the depth potential field. Thus it is harder to drive *heavy* packets with larger weight out of the shortest paths than *light* ones with smaller weight. They will immediately occupy most of the buffer along the shortest paths. The backlog filled by the heavy packets will further reject the light ones. IDDR will find other nodes and paths to cache the *light* packets to improve their throughput, while the shortest paths are used to forward the heavy ones to decrease their end-to-end delay.

Once a node receives a packet with a non-zero weight, α' will be calculated to form an assistant routing table relative to the main routing table established using the original value of α. Note that IDDR just builds multiple potential fields temporally and locally, but does not maintain all the possible fields (at most 256) across the whole network all the time, which can decrease the implementation overhead. Furthermore, Rule 1 will be disabled for delay-sensitive packets. In other words, delay-sensitive packets are not cached for the purpose of decreasing the end-to-end delay.

3.3.3 Priority Queue

To further decrease the queuing delay, IDDR employs the priority queue mechanism to allow the delay-sensitive packets to be transmitted prior to the other packets. Specifically, the most *heavy* packets are transmitted first and the others are ranked according to their weights. The packets with the same weight are ordered according to their arrival time.

3.4 Design of IDDR Algorithm

3.4.1 Procedure of IDDR

Consider a WSN with different high-integrity or delay-sensitive applications. Let c be the identifier of different applications. In summary, the main procedure of the IDDR algorithm at node i works as follows:

1. If the queue at node i is not empty, then $\alpha^{(c)} = \alpha + \frac{\text{packet weight of } p}{0xff}$ is computed for packet p at the head of the queue.
2. Let $W_{i,b}(t) = \{(Q_i(t) + \alpha^{(c)} D_i(t)) - (Q_b(t) + \alpha^{(c)} D_b\ (t))\}$. Select the next hop $b^* = \text{argmax}_{b \in \Omega_i(t), Q_j(t) \neq 1} W_{i,b}(t)$.
3. Node i sends packet p to node b^*. Go to (1).

3.4.2 Construction of Depth Potential Field

The depth potential field is important because it provides the basic routing function. It is constructed based on the depth value of each node. At the beginning, the depth values of all the nodes are initialized to $0xff$, except that the default depth of the sink is 0. The sink first sends a depth update message, the nodes one hop away from the sink obtain their own depth by adding 1 to the depth value in the update message and then send new update messages with their own depth values. Similarly, all the other nodes can obtain their own depth by receiving update messages from their neighbors who already know the depth value. Multiple sinks may exist in large scale WSNs. According to the procedure of the depth potential field construction, these sinks will periodically broadcast their update messages of depth. The nodes receive these update messages, compare the different depth values from different sinks, and then choose the nearest sink as its destination. If the smallest depth value is not unique, the node can choose one of them randomly. Actually, when multiple sinks exist in a large scale WSN, IDDR will naturally partition the whole networks into subregions managed by different sinks. Therefore, IDDR can work in large scale WSNs with multiple sinks.

3.4.3 Signaling

Each node requires the depth and queue length of its neighbors to make forwarding decisions. How often to update the depth and queue length between neighbors is quite important since too small period leads to much overhead while too large period leads to imprecise information. IDDR defines a Maximum Update Interval (MUI) and a Least Update Interval (LUI) between two successive update messages. MUI is always larger than LUI. The update messages should be sent between a LUI and a MUI at least once. If no message is received from a neighbor during two MUIs intervals, this neighbor will be considered dead, and IDDR will recalculate the depth and other related values. An update message will be sent when any one of the following events occurs: (1) MUI timer expires. If the elapsed time since sending the last update message exceeds the MUI, a new update message will be sent immediately no matter whether the depth or queue length has changed. (2) Queue length variation exceeds a certain threshold. If the queue length of a node has varied 10 percent compared with that in the last successful update message, and the elapsed time exceeds the LUI since the last update message. (3) Depth changes. If the depth of a node has changed, and the elapsed time exceeds the LUI since the last successful update message.

4 Performance Analysis

As a decentralized algorithm, the proposed IDDR algorithm needs to be stable to guarantee its normal running. In this section, we will prove that IDDR is stable and throughput-optimal using the Lyapunov drift technique. To use the technique, we assume that IDDR operates in slotted time with slots normalized to integral units n, $n \in \{0, 1, 2, \ldots\}$. The length of the time slot can be LUI of update messages.

Consider a WSN as a graph $\mathcal{G}[n] = (\mathcal{N}[n], \mathcal{L}[n])$, where $\mathcal{N}$[n] is the set of nodes and $\mathcal{L}$[n] is the set of links at time slot n. Let $\mu_{a,b}[n]$ denote the transmission rate over link (a, b) at time slot n. Before proceeding further, we first summarize the assumptions for the analysis and define two concepts, i.e., stability and capacity region.

Assume that the packet arrival processes of different applications to node i at time slot n, $A_i^{(c)}[n]$, are i.i.d., and ergodic with rates $\lambda_i^{(c)}$, hence $\lim_{n\to\infty} \frac{1}{n}\sum_{\tau=0}^{n-1} A_i^{(c)}[\tau] = \lambda_i^{(c)}$. All the links have the same transmission rate. The transmission rate and the arrival rate are assumed to be bounded: $\sum_b \mu_{i,b}[n] \le \mu_{max,i}^{out}, \sum_a \mu_{a,i}[n] \le \mu_{max,i}^{in}, \sum_c \lambda_i^{(c)} \le \lambda_i^{max}$, where $\mu_{max,i}^{out}, \mu_{max,i}^{in}$ and λ_i^{max} are constants and represent the upper bounds of the summation of outgoing traffic, incoming traffic from neighbors and exterior incoming traffic at node i, respectively. All nodes have the same buffer size B.

Besides, since IDDR focuses on providing differentiated services at the routing layer, we assume a proper MAC layer protocol is provided in WSNs.

Stability. The network is *stable* under a policy if the sum of the number of backlogged packets in the network has an upper bound that does not depend on the initial condition.

Network capacity region. All the rate matrices $\lambda_i^{(c)}$ that can be supported over a network $\mathcal{G}$ composes the network capacity region $\Lambda_{\mathcal{G}}$.

Under a given stationary randomized transmission scheduling policy, let $\vec{f} = (f_{a,b}^{(c)})_{(a,b)\in\mathcal{L}}$ where $f_{a,b}^{(c)} = \mathbb{E}\{\mu_{a,b}^{(c)}[n]\}$. A routing algorithm providing differentiated services will stabilize the network if and only if the following conditions are satisfied:

$$f_{a,b}^{(c)} \ge 0, f_{a,a}^{(c)} = 0 \quad \forall a, b, c, \tag{13}$$

$$\lambda_i^{(c)} + \sum_a f_{a,i}^{(c)} \le \sum_b f_{i,b}^{(c)} \tag{14}$$

$$\sum_{i,c} \lambda_i^{(c)} = \sum_{a\in\Omega_{sink}} f_{a,sink}. \tag{15}$$

The establishment of equation (15) results from the fact that the sum of the exogenous flow is equivalent with the sum of the data that reaches to the sink when the network becomes stable.

We say traffic $\vec{A}[n] = (A_i^{(c)}[n])_{i\in\mathcal{N}}$ can be stabilized if a routing algorithm exists under which the mean of the number of packets queued in the network is bounded. According to the definition of the capacity region, $((1+\epsilon)\vec{A}[n]) \in \Lambda_{\mathcal{G}}$ implies that there exists a stationary randomized control algorithm that makes valid decisions $\tilde{\mu}$ such that $\mathbb{E}\{\sum_b \tilde{\mu}_{i,b}[n] - \sum_a \tilde{\mu}_{a,i}[n]\} = \lambda_i^{(c)} + \epsilon$, and the expectation is based only on the current topology state $S_i[n]$ and independent of the queue length of node i at time slot n, $U_i[n]$ [19].

Next we will prove that IDDR is stable whenever $\vec{A}[n]$ is interior in the capacity region $\Lambda_{\mathcal{G}}$. $\Omega_i[n]$ is the set of neighbors of node i at time slot n. $\hat{\Omega}_i[n] = \{b|b \in \Omega_i[n], Q_b[n] \ne 1\}$ and $\hat{\mathcal{L}}[n] = \{(a, b)|Q_b[n] \ne 1\}$.

Lemma 1. *Under the assumption that the link capacity is equal for all links, the distributed IDDR algorithm is equivalent to the centralized optimization problem*

$$\max \sum_{(i,b)\in\hat{\mathcal{L}}[n]} W_{i,b}[n]\mu_{i,b}[n]. \tag{16}$$

Proof. In IDDR, node i selects the next hop b^* according to the solution of $\max_{b\in\hat{\Omega}_i[n]} W_{i,b}[n]$. Hence the proposed IDDR algorithm is based on solving the optimization problem: $\max \sum_i \max_{b\in\hat{\Omega}_i[n]} W_{i,b}[n]$. Since all links have the same link capacity, the problem of $\max \sum_i \max_{b\in\hat{\Omega}_i[n]} W_{i,b}[n]$ can be transformed into $\max \sum_i \max_{b\in\hat{\Omega}_i[n]} \{W_{i,b}[n]\mu_{i,b}[n]\}$.

Since the routing scheme does not consider the interference in MAC layer and a node will send packets to only one neighbor of it at the same time, we can obtain that:

$$\begin{aligned}
&\max \sum_i \max_{b\in\hat{\Omega}_i[n]} \{W_{i,b}[n]\mu_{i,b}[n]\} \\
&= \max \sum_i \left(\max \left\{ \sum_{b\in\hat{\Omega}_i[n]} W_{i,b}[n]\mu_{i,b}[n] \right\} \right), \\
&= \max \sum_i \sum_{b\in\hat{\Omega}_i[n]} W_{i,b}[n]\mu_{i,b}[n].
\end{aligned} \tag{17}$$

Note that $\{\sum_i (i, b)|b \in \hat{\Omega}_i[n]\}$ can be written as $\{(i, j)|(i, j) \in \hat{\mathcal{L}}[n]\}$. So we have $\max \sum_i \sum_{b\in\hat{\Omega}_i[n]} W_{i,b}[n] \times \mu_{i,b}[n] = \max \sum_{(i,b)\in\hat{\mathcal{L}}[n]} W_{i,b}[n]\mu_{i,b}[n]$. □

Lemma 1 shows that the properties of IDDR could be derived by analyzing the centralized algorithm.

$U_i[n]$ is the queue length of node i at time slot n. Define the Lyapunov function $L(\vec{U}[n])$ as follows:

$$L(\vec{U}[n]) \triangleq \frac{1}{2}\sum_{i=1}^{N} (U_i[n])^2. \tag{18}$$

Note that $L(\vec{U}[n]) = 0$ if and only if all network queues are empty at time slot n, and that $L(\vec{U}[n])$ is large whenever one or more components of $\vec{U}[n]$ is large.

$\Delta(\vec{U}[n])$ is defined as the one-step conditional Lyapunov drift:

$$\Delta(\vec{U}[n]) \triangleq \mathbb{E}\{L(\vec{U}[n+1]) - L(\vec{U}[n])|\vec{U}[n]\}. \quad (19)$$

In the following proofs of Lemma 2 and Theorem 1, the subscripts b denotes $b \in \hat{\Omega}_i[n]$ and a denotes $a \in \Omega_i[n]$. For the sake of saving space, some subscripts are not explained.

Lemma 2. *The one-step conditional Lyapunov drift satisfies the following constraint for all n and all $\vec{U}[n]$,*

$$\Delta(\vec{U}[n]) \le M + \sum_i \Big(U_i[n]\mathbb{E}\{\lambda_i^{(c)}|U_i[n]\} - U_i[n]\mathbb{E}\Big\{\Big(\sum_{b\in\hat{\Omega}_i[n]} \mu_{i,b}[n] - \sum_{a\in\Omega_i[n]} \mu_{a,i}[n]\big)|U_i[n]\Big\}\Big). \quad (20)$$

With constant $M \triangleq \frac{1}{2}\sum_i[(\mu_{max,i}^{out})^2 + (\mu_{max,i}^{in} + \lambda_i^{max})^2]$, where $\mu_{max,i}^{out}$, $\mu_{max,i}^{in}$ and λ_i^{max} are constants and represent the upper bounds of the summation of outgoing traffic, incoming traffic from neighbors and exterior incoming traffic in terms of node i, respectively.

Proof. The queue length at slot $n+1$ at node i can be bounded in terms of the current backlog at slot n as follows:

$$U_i[n+1] = max\Big\{U_i[n] - \sum_{b\in\hat{\Omega}_i[n]} \mu_{i,b}[n], 0\Big\} + \sum_{a\in\Omega_i[n]} \mu_{a,i}[n] + A_i[n],$$

where $A_i[n]$ represents the exogenous arrival process at node i during time slot n. Hence we have:

$$(U_i[n+1])^2 \le \Big(U_i[n] - \sum_b \mu_{i,b}[n]\Big)^2 + \Big(A_i[n] + \sum_a \mu_{a,i}[n]\Big)^2 + 2U_i[n]\Big(A_i[n] + \sum_a \mu_{a,i}[n]\Big).$$

Because the arrival processes are i.i.d., we can get $\lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1} A_i^{(c)}[\tau] = \lambda_i^{(c)}$. Thus, the Lyapunov drift is

$$\Delta(\vec{U}[n]) \le \frac{1}{2}\sum_i \Big[\Big(\sum_b \mu_{i,b}[n]\Big)^2 + \Big(\lambda_i^{(c)} + \sum_a \mu_{a,i}[n]\Big)^2 - U_i[n]\mathbb{E}\Big\{\Big(\sum_b \mu_{i,b}[n] - \lambda_i^{(c)} - \sum_a \mu_{a,i}[n]\big)|U_i[n]\Big\}\Big].$$

With constant $M \triangleq \frac{1}{2}\sum_i[(\mu_{max,i}^{out})^2 + (\mu_{max,i}^{in} + \lambda_i^{max})^2]$, we can obtain Eq. (20). □

Theorem 1. *Given the arrival processes $\vec{A}[n]$ such that $((1+\epsilon)\vec{A}[n]) \in \Lambda_{\mathcal{G}}$ and the topology state $\vec{S}[n]$ is i.i.d. from slot to slot, the network is stochastically stable under the IDDR algorithm.*

Proof. By adding and subtracting $B\alpha^{(c)}D_i[n]\mathbb{E}\{(\sum_{b\in\hat{\Omega}_i[n]} \mu_{i,b}[n] - \lambda_i^{(c)} - \sum_{a\in\Omega_i[n]} \mu_{a,i}[n])|U_i[n]\}$ and noting that $U_i[n] = BQ_i[n]$, we get

$$\begin{aligned}\Delta(\vec{U}[n]) \le M + \sum_i B\Big[&(Q_i[n] + \alpha^{(c)}D_i[n])\mathbb{E}\{\lambda_i^{(c)}|U_i[n]\} \\ &- (Q_i[n] + \alpha^{(c)}D_i[n])\mathbb{E}\Big\{\Big(\sum_b \mu_{i,b}[n] - \sum_a \mu_{a,i}[n]\big)|U_i[n]\Big\}\Big] \\ &+ B\alpha^{(c)}D_i[n]\mathbb{E}\Big\{\Big(\sum_b \mu_{i,b}[n] - \lambda_i^{(c)} - \sum_a \mu_{a,i}[n]\Big)|U_i[n]\Big\}\end{aligned} \quad (21)$$

The IDDR algorithm maximizes $\sum_{(i,b)\in\hat{\mathcal{L}}[n]} W_{i,b}[n]\mu_{i,b}[n]$. Also, $((1+\epsilon)\vec{A}[n]) \in \Lambda_{\mathcal{G}}$ implies that there exists $\tilde{\mu}$ such that $\mathbb{E}\{\sum_b \tilde{\mu}_{i,b}[n] - \sum_a \tilde{\mu}_{a,i}[n]\} = \lambda_i^{(c)} + \epsilon$. Thus, we can get

$$\begin{aligned}&\sum_i \Big[(Q_i[n] + \alpha^{(c)}D_i[n]) \mathbb{E}\Big\{\Big(\sum_b \mu_{i,b}[n] - \sum_a \mu_{a,i}[n]\big)|U_i[n]\Big\}\Big] \\ &= \sum_a (Q_a[n] + \alpha^{(c)}D_a[n])\mathbb{E}\Big\{\sum_b \mu_{a,b}[n]|U_i[n]\Big\}[n] \\ &\quad - \sum_b (Q_b[n] + \alpha^{(c)}D_b[n])\mathbb{E}\Big\{\sum_a \mu_{a,b}[n]|U_i[n]\Big\} \\ &= \sum_{(a,b)\in\hat{\mathcal{L}}[n]} W_{a,b}[n]\mathbb{E}\{\mu_{a,b}[n]|U_i[n]\} \\ &\ge \sum_{(a,b)\in\hat{\mathcal{L}}[n]} W_{a,b}[n]\mathbb{E}\{\tilde{\mu}_{a,b}[n]|U_i[n]\} \\ &= \sum_i \Big[(Q_i[n] + \alpha^{(c)}D_i[n]) \mathbb{E}\Big\{\Big(\sum_b \tilde{\mu}_{i,b}[n] - \sum_a \tilde{\mu}_{a,i}[n]\big)|U_i[n]\Big\}\Big] \\ &= \sum_i [(Q_i[n] + \alpha^{(c)}D_i[n])(\lambda_i^{(c)} + \epsilon)].\end{aligned} \quad (22)$$

Furthermore, considering that $\mu_{a,b} > 0$ if and only if when node a and node b are neighbors, i.e., $|D_a[n] - D_b[n]| \le 1$, we can get

$$\begin{aligned}&\sum_i B\alpha^{(c)}D_i[n]\mathbb{E}\Big\{\Big(\sum_b \mu_{i,b}[n] - \sum_a \mu_{a,i}[n]\Big)|U_i[n]\Big\} \\ &= \sum_{a,b} B\alpha^{(c)}(D_a[n] - D_b[n])\mathbb{E}\{\mu_{a,b}[n]|U_i[n]\} \\ &\le \sum_i B\alpha^{max}\mu_{i,max}^{out}.\end{aligned} \quad (23)$$

Combining Eqs. (21)-(23) and defining a constant $M_1 \triangleq M + \sum_i B\alpha^{max}\mu_{max,i}^{out}$, we can obtain that

$$\Delta(\vec{U}[n]) \le M_1 - \sum_i \epsilon\big(U_i[n] + \alpha^{(c)}BD_i[n]\big). \quad (24)$$

Hence, $\exists U^{max}$ such that when $\sum_i U_i[n] > U^{max}$, $\Delta(\vec{U}[n]) < 0$, i.e., the network is stable. □

5 Experimental Evaluation

IDDR is proved stable as long as the exogenous arrival rates are interior in the network capacity region in last

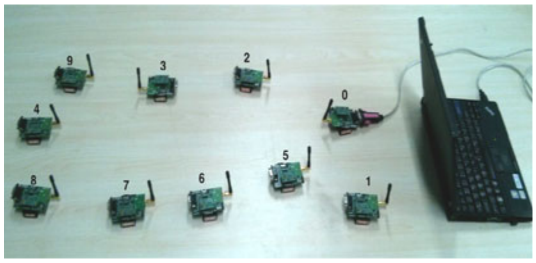


Fig. 4. Testbed. Node 4 and 8 generate packets with weight of 200 and 0, respectively. Node 1 is used for clock synchronization. Node 9 generates background traffic along the shorter path. The others are intermediate nodes.

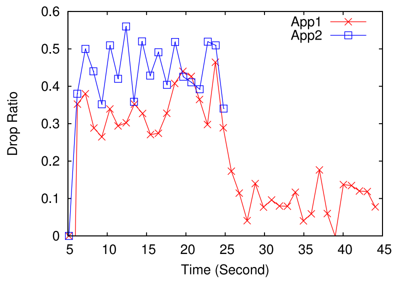


Fig. 5. Drop ratio of each application under IDDR on the testbed.

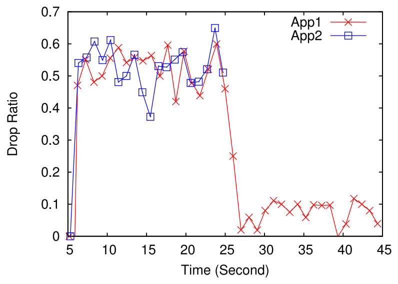


Fig. 6. Drop ratio of each application under MintRoute on the testbed.

section. In this section, we will evaluate whether IDDR can provide high data-integrity and delay-differentiated services using experiments.

5.1 Experimental Setup

The testbed is comprised of 10 sensors as shown in Fig. 4. There are one data-integrity application App 1 with weight of 0 and one delay-sensitive application App 2 with weight of 200. Nodes 8 and 4 are the source nodes of Apps 1 and 2, respectively. Both of the two types of packets have two paths to the sink. One path has three hops while the other one has four hops. The power of the sensor nodes except from node 1 is set to five to allow node 4 and node 8 to reach both of the two paths. To evaluate the effectiveness of IDDR, we let node 9 generate background traffic that transmit along the shorter path with three hops. Thus, the shorter path will be more congested than the longer path. If IDDR works normally, the high-integrity application App 1 will choose the longer path if the shorter path is too congested, while the delay-sensitive packets of App 2 will choose the shorter path with three hops to achieve small end-to-end delay. The parameter α in IDDR is set to 0.4.

Traffic generation. Source nodes 4, 8 and 9 generate packets at interval of 20 ms. Both of App 1, App 2 and the background traffic start from 5 second and App 1 and the background traffic end at 25 second while App 2 ends at 45 second. The packet size is 25 Bytes. The buffer size in each node is eight packets. The LUI of the depth and queue length information is 200 milliseconds and the MUI is 20 seconds.

Performance metrics. To investigate whether IDDR can provide high integrity and low end-to-end delay, we use the average drop ratio and packet delay as the performance metrics. To obtain the end-to-end delay suffered by a packet, the sink needs to subtract the packet send time from the packet receive time. Since the packet send time is assigned by the sender, time synchronization is required among all the sensor nodes to obtain accurate end-to-end delay. In our testbed, we use node 1 to synchronize the clock of all the sensor nodes. The power of node 1 is set to the maximum value, 31, to make it can communicate with all the other sensors. At the start of our experiments, node 1 sends a packet to all the other sensors to restart their clock. The TinyOS's standard routing algorithm, MintRoute, is selected as the reference protocol.

5.2 Experimental Results

Figs. 5 and 6 show the drop ratio of the two applications versus time with IDDR and MintRoute on our testbed, respectively. We can see that both of App 1 and App 2 in IDDR achieve smaller drop ratio than MintRoute. From 5 to 25 second, the drop ratio of App 1 is smaller than that of App 2. This is because if the load difference of the shorter path and the longer path is quite large, then the packets of App 1 will choose the longer path since the longer path has much smaller queue length potential field. Otherwise, the packets of both of Apps 1 and 2 will be sent to node 3. However, after some time, the queue length potential field of node 3 will increase to a large enough value to force the packets of App 1 to choose node 7 as the next hop to achieve high fidelity according to Proposition 1. Therefore, the congestion along the shorter path is always more severe than the longer path. Correspondingly, App 2 has higher drop ratio. While in MintRoute, App 1 and App 2 suffer almost the same drop ratio, and the drop ratio is larger than that in IDDR. Especially, the drop ratio of App 1 in MintRoute is about twice of that in IDDR. This is because all the packets are not differentiated in MintRoute, they will choose the shorter path no matter whether it is congested or not. From 25 to 45 second, App 2 does not generate more packets, the drop ratio of App 1 decreases in both IDDR and MintRoute since App 1 takes up the shorter path without contention with App 2. Fig. 7 shows the queue length at node 3 and node 7. It shows that before 25 second, node 3 has larger queue length since it is always more congested than node 7. After 25 second, the queue length of node 3 is quite small and node 7 is zero.

Figs. 8 and 9 depict the end-to-end delay with IDDR and MintRoute, respectively. From 5 to 25 seconds, the end-to-end delay of App 1 is quite larger since most packets of App 1 are sent to the sink along the longer path. The shorter path has three hops while the longer path has four hops. The ratio of the end-to-end delay suffered by Apps 1 and 2 is accord with the ratio of the length of the two paths. While in MintRoute, the packets of Apps 1 and 2 have almost the same end-to-end delay. From 25 to 45 seconds,

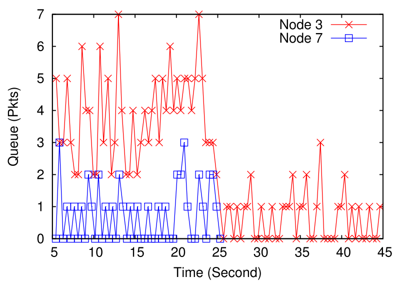


Fig. 7. Queue length of node 3 and node 7 under IDDR.

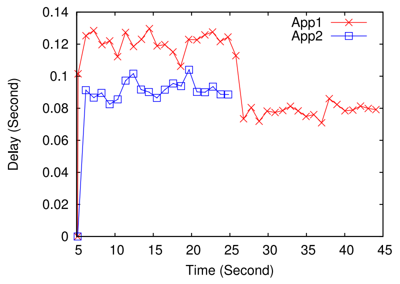


Fig. 8. Average packet delay of each application under IDDR on the testbed.

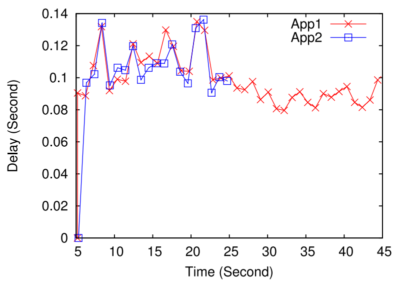


Fig. 9. Average packet delay of each application under MintRoute on the testbed.

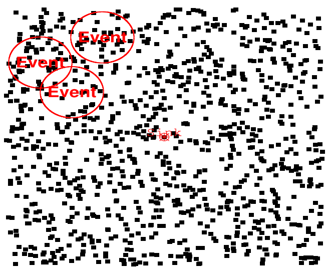


Fig. 10. Simulation topology: Randomly deployed network.

App 2 terminates. In IDDR, we can see that App 1 moves traffic to the shorter path and thus achieves smaller end-to-end delay. While in MintRoute, Since App 1 always choose the shorter path, the end-to-end delay only decreases a little. The reduction is caused by the smaller queuing delay at the intermediate nodes.

6 Simulations

To evaluate the performance of IDDR in large-scale WSNs, a series of simulations are conducted on the TOSSIM platform built in TinyOS [1]. We specify IDDR with $\alpha = 0.6$ and $\alpha = 0.4$ as the representative algorithms. As previously stated after the remarks of Proposition 1 and 2, when $\alpha = 0.4$, the packets are allowed to be sent to the neighbors with the same or higher depth to bypass the hotspots, whereas when $\alpha = 0.6$, the packets are only allowed to be transmitted to the neighbors with the same depth. The performance of IDDR with $\alpha = 0.6$ and $\alpha = 0.4$ is compared with MintRoute and IDDR with $\alpha = 0.1$.

6.1 Simulation Setup

Fig. 10 shows a randomly deployed rectangular network and three monitoring areas. 600 nodes spreading over a 100×100 meters square form a flat multi-hop network. There is only one sink residing at the center, and the communication range is 6 meters. The detailed deployment configuration is summarized in Table 1.

There are three applications running over the network from $100\ \mathrm{s}$ to $160\ \mathrm{s}$: APP 1 is one high-integrity application generating packets with weight of 0 at the sampling rate of 4 Kbps. APP 2 and APP 3 are delay-sensitive applications and generate packets with weights of 50 and 200, respectively at the sampling rate of 8 Kbps. Fig. 10 indicates the positions of the monitoring areas. and Table 2 describes when and how these applications generate packets.

6.2 High-Integrity Services

In this subsection, we evaluate the ability of IDDR to provide high integrity services. Only the basic *Resource Discovery* and Implicit *Hop-by-hop Rate Control (Rule 1)* functions that are designed to support *high-integrity services* are enabled, while the delay differentiated service is shielded, that is, the packet weight of all the applications are zero.

Table 3 shows the throughput under MintRoute and IDDR with different α. The three applications have generated totally $3,571$ packets. IDDR ($\alpha = 0.6$) receives 2,893 of them and IDDR ($\alpha = 0.4$) gets 3,142, but MintRoute and the shortest path algorithm (IDDR with $\alpha = 1.0$) receive 1,599 and 2,106 packets, respectively. These results indicate IDDR significantly improves the throughput.

Fig. 11 presents the receiving packet rate, i.e., the rate at which the sink receives packets, which explains why IDDRs have higher throughput. A lot of packets that are likely dropped under other routing algorithms are cached for a short time and eventually reach the sink in IDDR. Thus, IDDR successfully smooths the bursts and prevents the burst packets from dropping. On the contrary, MintRoute drops most of the burst packets. Although the shortest path algorithm (IDDR with $\alpha = 1.0$) performs much better than MintRoute, both of them receive fewer packets at the sink out of the bursting time $(100\ \mathrm{s} \sim 130\ \mathrm{s})$, while IDDRs continue to receive a lot.

Rule 1 plays an important role in improving the throughput. It ensures that the excessive packets are efficiently cached and prevents them from dropping. Another simulation experiment has confirmed this statement: without Rule 1, the throughput of IDDR ($\alpha = 0.6$) and IDDR ($\alpha = 0.4$) drops to 2,192 (from 2,893) and 2,333 (from 3,142), respectively. In addition, Rule 1 indeed slows down the transmission rates in a hop-by-hop way. When the paths between the source nodes and the sink are full of packets, the source will be compelled to

TABLE 1
Configuration of Parameters: Random Deployment

Deployment	Area Size	$100m \times 100m$
	Deployment Type	Randomly
	Network Architecture	Homogeneous, Flat
	Number of Nodes	600
	Largest Depth	26
	Average Node Degree	8
	Sink	$(50m, 50m)$
	Radio Range	$6m$
Task	Application Type	Event-driven
	Packet Size (Payload)	30 bytes
	Header Size	27 bytes
IDDR	Buffer Size	31 pkts
	α	$1, 0.6$ and 0.4
	LUI	1 s
	MUI	20 s
Simulation	Time	300 seconds

TABLE 2
Applications and Packet Bursts Description

	Weight	Time	Rate	Type
App 1	0	$100s \sim 160s$	20 p/s	High-integrity
App 2	50	$100s \sim 130s$	40 p/s	Delay-sensitive
App 3	200	$100s \sim 130s$	40 p/s	Delay-sensitive

TABLE 3
Integrity Services: Throughput of the Whole Network

	MintRoute	IDDR($\alpha = 1.0$)	IDDR($\alpha = 0.6$)	IDDR($\alpha = 0.4$)
App 1	710	831	982	1024
App 2	442	713	867	1052
App 3	447	562	1044	1066
Total	1599	2106	2893	3142

decrease its rate. Fig. 12 illustrates that the source node does not inject all the packets generated by the applications into the network during the bursting time (100 s $\sim$ 130 s), which confirms that the implicit source rate control endowed by Rule 1 is effective.

To improve the fidelity required by high data integrity applications, IDDR likely introduces some extra delay. Table 4 shows the end-to-end delay of each application under MintRoute and IDDRs with different α. As stated in Section 2, this extra delay is unavoidable because of the many-to-one traffic pattern of WSN.

Fortunately, IDDR can provide delay differentiated services to decrease the end-to-end delay for real time applications (App2 and APP3) by establishing multiple potential fields with different slopes. A priority queue is also used to shorten the queuing delay. In the next subsection, we will show how IDDR provides delay differentiated services without affecting the fidelity of high integrity applications.

6.3 Delay Differentiated Services

Table 5 shows the end-to-end delay of each application under IDDR with $\alpha = 0.6$ and $\alpha = 0.4$. Compared with Table 4, the full version of IDDR successfully decreases the end-to-end delay for delay-sensitive applications (App2 and App3), while the throughput of the non-delay-sensitive application remains relatively high, i.e., 1,171 packets are generated and 1,042 packets are received.

Fig. 13 shows the receiving packet rate of each application at the sink. IDDR caches most packets generated by

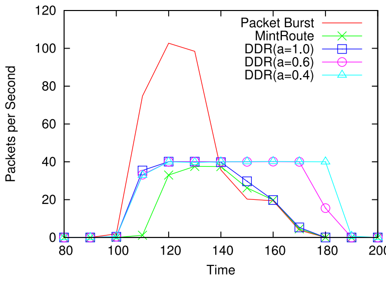


Fig. 11. Integrity services: The distribution of transmission over the time. The RPR value is an average in every 10 seconds.

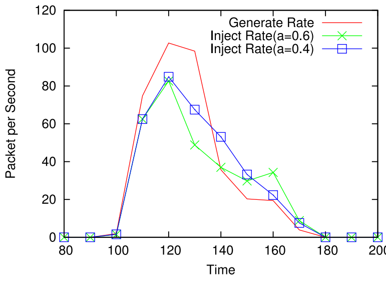


Fig. 12. The implicit source rate control of Rule 1.

TABLE 4
Integrity Services: End-to-End Delay

	MintRoute	IDDR($\alpha = 1.0$)	IDDR($\alpha = 0.6$)	IDDR($\alpha = 0.4$)
App 1	1.52	2.80	14.31	19.91
App 2	3.13	3.70	23.90	20.49
App 3	4.14	5.96	10.15	15.58
Average	2.70	3.95	15.69	18.64

App1 in the intermediate nodes during the bursting time (100 s $\sim$ 130 s) to give way to the delay-sensitive applications. After the burst period, all these packets are eventually sent to the sink, which keeps high throughput for the high-integrity application (App1). Thus, what IDDR has done is to give higher priority to delay-sensitive packets, and at the same time, cache non-delay-sensitive packets in the idle and underloaded paths for later transmission to avoid packet losses.

Fig. 14 plots the average end-to-end delay of each application. We can see that both of the delay-sensitive applications gain a steady low delay while App 1 suffers much larger delay. The reason is already shown in Fig. 13: most packets from App 1 wait to be transmitted after the packets from App 2 and 3. At 170 s, since most of the blocked packets have arrived at the sink, the end-to-end delay of App 1 becomes quite small.

As mentioned in Section 3.3, one important reason that IDDR can decrease the delay is that IDDR shortens the path of the heavy packets by building fields with different slopes. Fig. 15 presents the distribution of hops experienced by packets. Obviously, most of the packets from two delay-sensitive applications have been received within 26 hops, which is close to the maximum depth in the network, while the packets from App1 travel more hops because they may be forwarded in non-shortest paths.

Priority queue is another mechanism used to decrease the end-to-end delay. Two sets of simulations are conducted to evaluate the impact of priority queue. One is only with

TABLE 5
Delay Differentiated Services: Throughput and End-to-End Delay

	IDDR($\alpha = 0.6$)		IDDR($\alpha = 0.4$)	
	Throughput	EED	Throughput	EED
App 1	1042	17.09	1050	18.21
App 2	797	6.10	842	6.56
App 3	709	3.24	691	3.35
Total/Average	2548	9.80	2583	10.44

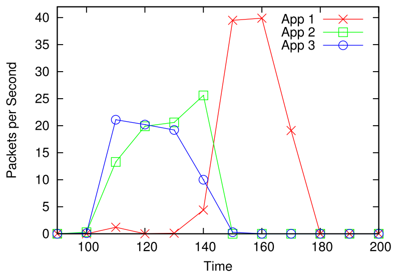


Fig. 13. Receive packet rate of each application ($\alpha = 0.4$).

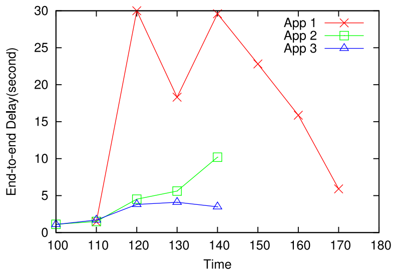


Fig. 14. Distribution of delay of each application ($\alpha = 0.4$).

the priority queue, but without the support of the different sloped fields, another has only the different sloped fields, but without the priority queue. By comparing the data in Tables 5 and 6, we can conclude that the priority queue significantly influences the performance of IDDR. On one hand, without the different sloped fields, delay-sensitive packets will be driven out of the shortest path and compelled to travel more hops before reaching the sink. On the other hand, without the priority queue, they will be delayed in the full queue or even be blocked by the non-delay-sensitive packets. The combination of these two mechanisms shows a nice performance (see Table 6).

6.4 Other Considerations

6.4.1 Energy Efficiency

Energy is the most critical resource for WSNs. The energy efficiency of IDDR is investigated. In all simulations, we use a simple linear energy model. Since sending a packet needs more energy than receiving one [20], without loss of generality, we assume that 3 units of energy consumed for sending, and 2 units for receiving. Table 7 presents the energy consumption per received packet which is usually used to evaluate the energy efficiency. The result show that IDDR with $\alpha = 0.6$ and $\alpha = 0.4$ have to spend more energy to successfully transfer a packet than the shortest path algorithm. The main reason is that the non-delay-sensitive packets would have to travel more hops

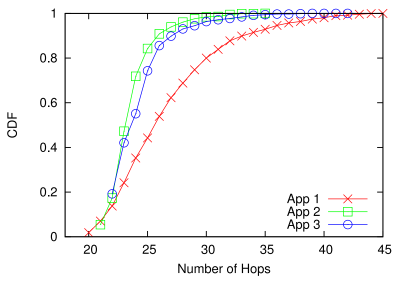


Fig. 15. The distribution of hops received packets travels ($\alpha = 0.4$).

TABLE 6
Evaluate the Priority Queue: End-to-End Delay

	Only Priority Queue		Only Slopped Fields	
	IDDR($\alpha = 0.6$)	IDDR($\alpha = 0.4$)	IDDR($\alpha = 0.6$)	IDDR($\alpha = 0.4$)
App 1	24.25	31.50	7.06	7.62
App 2	19.77	16.67	7.39	8.18
App 3	5.29	6.27	8.99	8.94
Average	15.97	18.08	7.78	8.22

TABLE 7
Energy Consumption per Received Packet

MintRoute	IDDR($\alpha = 1.0$)	IDDR($\alpha = 0.6$)	IDDR($\alpha = 0.4$)
199.5	122.36	128.69	135.23

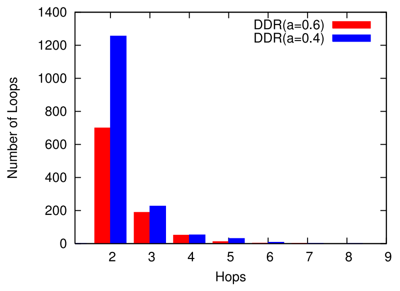


Fig. 16. The distribution of hops of loops.

(see Fig. 15) to be cached in the idle and/or underloaded areas. However, it is worthy to better services with some extra energy expenditure.

6.4.2 Reasonable Routing Loops

It is proved that the time-invariant potential field is loop-free [17]. Unfortunately, the queue length field in IDDR is a time-variant field, and we have indeed observed the routing loops.

A typical routing loop is caused by a local minimal potential, which is a hollow in our bowl model. At the beginning, nodes around this minimal potential node may send their packets to it, so this hollow will be filled up after some time. Once the potential of this node goes higher than that of any nodes around it, it will send back the packets just received from the neighbors, which causes a routing loop. We believe that this type of loops is reasonable because the node with a locally minimal potential acts like a packet pool to cache the excessive high-integrity packets. Note that, a characteristic of this kind of loops is that they just occur in two hops.

Fig. 16 presents the distribution of the number of radio links in the loops under IDDR ($\alpha = 0.6$ and $\alpha = 0.4$). Most of

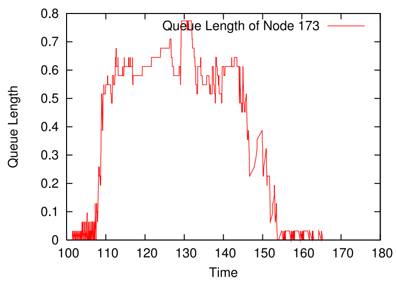


Fig. 17. The variation of the queue length on node 173.

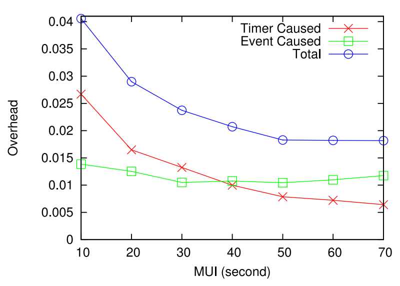


Fig. 18. The overhead caused by update messages with different MUI.

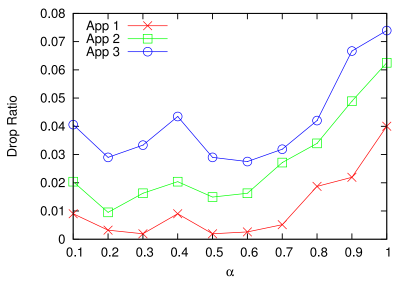


Fig. 19. Drop ratio versus different α values.

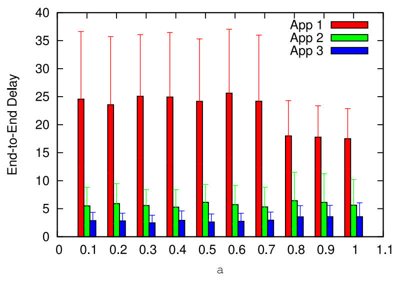


Fig. 20. End-to-End delay with standard deviation versus different α values.

the loops occur in two hops, which means most of them are involved in a reasonable packet pool. Therefore, we conclude that IDDR just suffers a tiny routing loop problem.

6.4.3 Queue Oscillation

Generally, using the queue length as a routing metric possibly causes routing oscillation. However, IDDR does not suffer from this problem. When a hot spot appears, the flows carrying the lightest packets would be first deviate this path, then the second lightest one, and so on. When a large queue begins to fall, the relatively heavy packets would firstly come back, and so on. Therefore, there are always some flows passing through the hot spot, and the oscillation may seldom happen because not all the flows are driven out simultaneously, and also they do not switch back at the same time. Fig. 17 illustrates the variation of the queue length at node 173 which resides on the shortest path of the application monitoring areas and near the sink. The queue length does not exhibit obvious oscillation.

6.4.4 Overhead

The cost of detecting topology variation or queue length information is unavoidable for IDDR. As stated in Section 3.4.3, when the queue length or depth changes or the MUI timer fires, an update message will be sent. We called the overhead caused by the depth or queue length variation as *Event Caused*, and that caused by MUI timer fires as *Timer Caused*. We measured the overhead caused by these operations. The overhead is defined as the ratio of the number of bytes used to transmit update messages to that used to transmit the sampled data. In the TOSSIM simulator, the packet header size is 27 bytes, and the payload has 30 bytes. While the update message includes depth and queue length information, which takes 2 bytes. Besides, it should contain the source address (generally 2 bytes) and type (1 bytes), which take 3 bytes. Hence, each update message has 5 bytes.

Fig. 18 shows the overhead caused by the event or MUI timer with different MUI values. We can see that the overhead is between 0.5 and 4 percent, which is acceptable. As the MUI becomes larger, the overhead of *Timer Caused* decreases, while the overhead of *Event Caused* increases. However, when MUI is larger than 50 s, the total overhead almost keeps stable.

6.5 Parameter Analysis

6.5.1 Parameter of IDDR α

The parameter α plays a vital role in the proposed IDDR algorithm. To evaluate the impact of it on the performance of IDDR, a series of simulations with different α values are conducted. Fig. 19 shows the drop ratio of the three applications versus different α values. We can see that as α increases, the drop ratio increases. This is because the larger is α, the smaller is the weights of the queue length potential field. Then the probability of a packet being transmitted to a node with lower depth but larger queue length grows. Therefore, more packets will be dropped due to contention for the congested buffer space.

Fig. 20 illustrates the end-to-end delay with standard deviation. The end-to-end delay of App 1 decreases as α increases. Similar to the analysis of Fig. 19, with larger α, the packets of App 1 would select shorter paths to reach the sink. However, the end-to-end delay of App 2 and 3 changes a little. This is because the weights of them are larger and thus their packets are likely to be routed along the shortest path even if α is small. For example, the packets of App 3 have the weight value of 200. Even if $\alpha = 0.1$, $\alpha + \frac{weight}{0xff} = 0.88$. What's more, when α increases, more packets of App 1 contend for the shortest path, leading to larger queuing delay. Therefore, the end-to-end delay of App 2 and 3 with different α change little.

Note that the results shown in Figs. 19 and 20 also indicate that the performance of IDDR is not very sensitive to α.

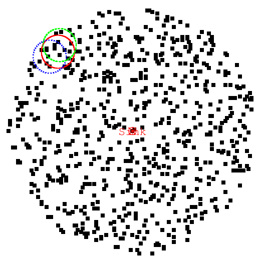


Fig. 21. Circular topology with 700 randomly deployed nodes and three events.

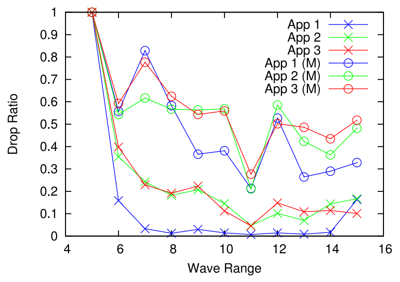


Fig. 22. Drop ratio versus different communication ranges.

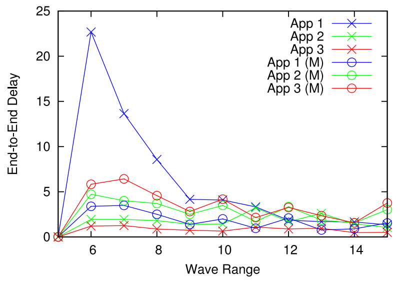


Fig. 23. Average End-to-End delay with different communication range.

Especially when α takes values between 0.1 and 0.7, the drop ratio and end-to-end delay are almost invariant.

6.6 Other Scenarios

To evaluate whether IDDR works well in other scenarios, we generate a circular topology with 700 nodes and three applications as shown in Fig. 21. The diameter of the WSN is 150 meters and the sink locates at the center of the circle. App 1 requires high integrity while App 2 and 3 are delay-sensitive applications.

Different communication range. In practice, the communication range of sensor nodes can be changed by adjusting the power value. Since IEEE 802.15.4 with the communication range of around 10 meters is a widely used MAC protocol in WSNs, we conduct a series of simulations by varying the communication range from 5 to 15 meters. Given that the topology is unchanged, the node density grows as the communication range increases.

Figs. 22 and 23 show the drop ratio and end-to-end delay of the three applications with different communication ranges. We can see that Apps 2 and 3 achieve small end-to-end delay and suffer higher drop ratio, while App 1 with high-integrity requirement suffers larger end-to-end delay but has the smallest drop ratio. This is because the packets of App 1 bypass the congested shortest path

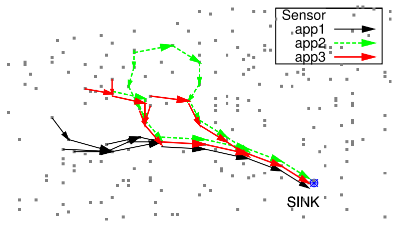


Fig. 24. Route paths of MintRoute.

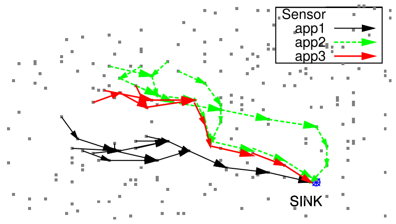


Fig. 25. Route paths of IDDR.

and move along the longer and lighter paths to the sink. The packets of App 1 do not contend the bandwidth of the shortest paths with the packets of Apps 2 and 3, thus Apps 2 and 3 loss fewer packets as well as achieve small end-to-end delay.

We investigate the log in the scenario with the communication range of 10 meters and draw their route paths to intuitively explain why IDDR can provide differentiated services. Figs. 24 and 25 show the route paths of the three different applications under MintRoute and IDDR, respectively. In MintRoute, three applications quickly converge to the same path since MintRoute makes routing decisions without considering the requirements of applications. While in IDDR, the packets of App 3, which is the most sensitive to delay, move along the shortest path to the sink. App 1, which requires the highest data-integrity, bypasses the shortest path to avoid contending bandwidth with App 3. And the packets of App 2, whose delay requirement is between that of Apps 1 and 3, move to the sink along one shortest path and one longer path. We can see that IDDR indeed differentiates different applications and select proper paths based on their requirements.

7 Conclusion

In this paper, a dynamic multipath routing algorithm IDDR is proposed based on the concept of potential in physics to satisfy the two different QoS requirements, high data fidelity and low end-to-end delay, over the same WSN simultaneously. The IDDR algorithm is proved stable using the Lyapunov drift theory. Moreover, the experiment results on a small testbed and the simulation results on TOSSIM demonstrate that IDDR can significantly improve the throughput of the high-integrity applications and decrease the end-to-end delay of delay-sensitive applications through scattering different packets from different applications spatially and temporally. IDDR can also provide good scalability because only local information is required, which simplifies the implementation. In addition, IDDR has acceptable communication overhead.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the anonymous reviewers for their constructive comments. This work is supported in part by National Basic Research Program of China (973 Program) under Grant No. 2014CB347800 and 2012CB315803, and National Natural Science Foundation of China (NSFC) under Grant No. 61225011.

REFERENCES

[1] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in *Proc. 1st Int. Conf. Embedded Networked Sensor Syst.*, 2003, pp. 126–137.

[2] T. Chen, J. Tsai, and M. Gerla, "QoS routing performance in multi-hop multimedia wireless networks," in *Proc. IEEE Int. Conf. Universal Personal Commun.*, 1997, pp. 557–561.

[3] R. Sivakumar, P. Sinha, and V. Bharghavan, "CEDAR: Core extraction distributed ad hoc routing algorithm," *IEEE J. Selected Areas Commun.*, vol. 17, no. 8, pp. 1454–1465, Aug. 1999.

[4] S. Chen and K. Nahrstedt, "Distributed quality-of-service routing in ad hoc networks," *IEEE J. Selected Areas Commun.*, vol. 17, no. 8, pp. 1488–1505, Aug. 1999.

[5] B. Hughes and V. Cahill, "Achieving real-time guarantees in mobile ad hoc wireless networks," in *Proc. IEEE Real-Time Syst. Symp.*, 2003.

[6] 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 Comput.*, vol. 5, no. 6, pp. 738–754, Jun. 2003.

[7] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He, "RAP: A real-time communication architecture for large-scale wireless sensor networks," in *Proc. IEEE 8th Real-Time Embedded Technol. Appl. Symp.*, 2002, pp. 55–66.

[8] M. Caccamo, L. Zhang, L. Sha, and G. Buttazzo, "An implicit prioritized access protocol for wireless sensor networks," in *Proc. IEEE Real-Time Syst. Symp.*, 2002, pp. 39–48.

[9] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A stateless protocol for real-time communication in sensor networks," in *Proc. IEEE 23rd Int. Conf. Distrib. Comput. Syst.*, 2003, pp. 46–55.

[10] P. T. A. Quang and D.-S. Kim, "Enhancing real-time delivery of gradient routing for industrial wireless sensor networks," *IEEE Trans. Ind. Inform.*, vol. 8, no. 1, pp. 61–68, Feb. 2012.

[11] S. Bhatnagar, B. Deb, and B. Nath, "Service differentiation in sensor networks," in *Proc. Int. Symp. Wireless Pers. Multimedia Commun.*, 2001.

[12] B. Deb, S. Bhatnagar, and B. Nath, "ReInForM: Reliable information forwarding using multiple paths in sensor networks," in *Proc. IEEE Intl Conf. Local Comput. Netw.*, 2003, pp. 406–415.

[13] M. Radi, B. Dezfouli, K. A. Bakar, S. A. Razak, and M. A. Nematbakhsh, "Interference-aware multipath routing protocol for QoS improvement in event-driven wireless sensor networks," *Tsinghua Sci. Technol.*, vol. 16, no. 5, pp. 475–490, 2011.

[14] J. Ben-Othman and B. Yahya, "Energy efficient and QoS based routing protocol for wireless sensor networks," *J. Parallel Distrib. Comput.*, vol. 70, no. 8, pp. 849–857, 2010.

[15] 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 transactions on communications," *IEICE Trans. Commun.*, vol. 91B, no. 8, pp. 2589–2601, 2008.

[16] D. Djenouri and I. Balasingham, "Traffic-differentiation-based modular qos localized routing for wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 10, no. 6, pp. 797–809, Jun. 2010.

[17] A. Basu, A. Lin, and S. Ramanathan, "Routing using potentials: A dynamic traffic-aware routing algorithm," in *Proc. Conf. Appl., Technol., Architectures, Protocols Comput. Commun.*, 2003, pp. 37–48.

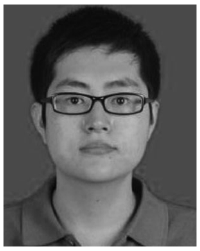
[18] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: Congestion detection and avoidance in sensor networks," in *Proc. 1st Int. Conf. Embedded Netw. Sensor Syst.*, 2003, pp. 266–279.

[19] L. Georgiadis, M. J. Neely and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–144, 2006.

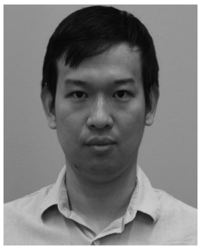
[20] A. Papadoulos and J. A. Mccann, "Towards the design of an energy-efficient, location-aware routing protocol for mobile, ad-hoc sensor networkFs," in *Proc. Int. Workshop Database Expert Syst. Appl.*, 2004, pp. 705–709.

Jiao Zhang is currently an assistant professor in the School of Information and Communication Engineering at Beijing University of Posts and Telecommunications (BUPT), Beijing, China. In July 2014, she received her Ph.D degree from the Department of Computer Science and Technology, Tsinghua University, China. Her supervisor is Prof. Fengyuan Ren. From August 2012 to August 2013, she was a visiting student in the networking group of ICSI, UC Berkeley. In July 2008, she obtained her Bachelor's Degree from the School of Computer Science and Technology from BUPT. Her research interests include traffic management in data center networks, routing in wireless sensor networks and future Internet architecture. Until now, she has (co)-authored more than 10 international journal and conference papers. She is a member of the IEEE.

Fengyuan Ren received the BA and MSc degrees in automatic control from Northwestern Polytechnic University, China, in 1993 and 1996, respectively, and the PhD degree in computer science from Northwestern Polytechnic University in December 1999. He is a professor of the Department of Computer Science and Technology at Tsinghua University, Beijing, China. From 2000 to 2001, he worked at the Electronic Engineering Department of Tsinghua University as a post doctoral researcher. In January 2002, he moved to the Computer Science and Technology Department of Tsinghua University. His research interests include network traffic management and control, control in/over computer networks, wireless networks, and wireless sensor networks. He authored and co-authored more than 80 international journal and conference papers. He is a member of the IEEE, and has served as a technical program committee member and local arrangement chair for various IEEE and ACM international conferences.



Shan Gao received the BE degree in software engineering from Southwest Jiaotong University, Chengdu, China, in 2011, and is working toward the master's degree in the Department of Computer Science and Technology of Tsinghua University, Beijing, China. He is supervised by professor Fengyuan Ren now. His research interests include wireless sensor networks and routing protocol.



Hongkun Yang received the BS and MS degrees in computer science from Tsinghua University, Beijing, in 2007 and 2010, respectively. He is currently working toward the PhD degree in the Computer Science Department, the University of Texas at Austin. His research interests include wireless networks and network security.

Chuang Lin received the PhD degree in computer science from Tsinghua University, China, in 1994. He is a professor of the Department of Computer Science and Technology at Tsinghua University, Beijing, China. He is an honorary visiting professor, the University of Bradford, United Kingdom. He has published more than 300 papers in research journals and IEEE conference proceedings in these areas and has published four books. He served as the technical program vice chair, the 10th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS 2004); the general chair, ACM SIGCOMM Asia workshop 2005, and the 2010 IEEE International Workshop on Quality of Service (IWQoS 2010). He is an associate editor of *IEEE Transactions on Vehicular Technology* and an area editor of *Computer Networks and the Journal of Parallel and Distributed Computing*. His current research interests include computer networks, performance evaluation, network security analysis, and Petri net theory and its applications. He is a senior member of the IEEE and the Chinese Delegate in TC6 of IFIP.