

# SDC: A Scalable Approach to Collect Data in Wireless Sensor Networks

Niwat THEPVILOJANAPONG<sup>†a)</sup>, Nonmember, Yoshito TOBE<sup>††b)</sup>, and Kaoru SEZAKI<sup>†††c)</sup>, Members

**SUMMARY** In this paper, we present Scalable Data Collection (SDC) protocol, a tree-based protocol for collecting data over multi-hop, wireless sensor networks. The design of the protocol aims to satisfy the requirements of sensor networks that every sensor transmits sensed data to a sink node periodically or spontaneously. The sink nodes construct the tree by broadcasting a HELLO packet to discover the child nodes. The sensor receiving this packet decides an appropriate parent to which it will attach, it then broadcasts the HELLO packet to discover its child nodes. Based on this process, the tree is quickly created without flooding of any routing packets. SDC avoids periodic updating of routing information but the tree will be reconstructed upon node failures or adding of new nodes. The states required on each sensor are constant and independent of network size, thereby SDC scales better than the existing protocols. Moreover, each sensor can make forwarding decisions regardless of the knowledge on geographical information. We evaluate the performance of SDC by using the *ns-2* simulator and comparing with Directed Diffusion, DSR, AODV, and OLSR. The simulation results demonstrate that SDC achieves much higher delivery ratio and lower delay as well as scalability in various scenarios.

**Key words:** sensor networks, data collection, routing, scalability, simulation

## 1. Introduction

Recent advances in MEMS-based sensor technology and low-power RF and OS design have enabled the development of relatively inexpensive and low-power wireless sensors [1]–[3]. A great number of such sensors can coordinate amongst themselves to achieve a larger sensing task both in urban environments and in inhospitable terrain [4]. For instance, we can use wireless sensor networks for environmental and habitat monitoring, tracking system, failure detection, and intrusion detection [5]–[8]. However, the proclaimed limitations of sensor networks which are resource constraints including memory storage, computational power, communication bandwidth, and energy resources motivate the challenges in designing a routing protocol that fulfills the requirements of sensor networks.

Generally, a large number of sensors are deployed in

Manuscript received July 23, 2004.

Manuscript revised October 15, 2004.

<sup>†</sup>The author is with the Department of Information and Communication Engineering, University of Tokyo, Tokyo, 113-8656 Japan.

<sup>††</sup>The author is with the Department of Information Systems and Multimedia Design, Tokyo Denki University, Tokyo, 101-8457 Japan.

<sup>†††</sup>The author is with the Center for Spatial Information Science (CSIS), University of Tokyo, Tokyo, 153-8505 Japan.

a) E-mail: wat@mcl.iis.u-tokyo.ac.jp

b) E-mail: yoshito@unl.im.dendai.ac.jp

c) E-mail: sezaki@iis.u-tokyo.ac.jp

DOI: 10.1093/ietcom/e88-b.3.890

the remote terrain. These sensors coordinate to establish a communication network, monitor specified tasks, and report sensed data periodically or spontaneously to the nearest base station\*. When the existing sensors are out of order due to numerous reasons, they must reorganize by themselves to repair failed routes. On the other hand, the user may deploy additional sensors to mitigate the severe effect of many failed sensors, thereby enforcing the sensors to reconstruct themselves to take advantage of the added system resources. Hence, we consider a scalable protocol that is based on a specific communication pattern and also robust to the dynamic natures of sensor networks.

The contributions of the paper are as follows.

- We propose a *Scalable Data Collection* (SDC) protocol which is a tree-based protocol for collecting data over multi-hop, wireless sensor networks. Each sensor can route the data to a potential nearest sink.
- We demonstrate that SDC works well in various scenarios through simulated networks.

The remainder of the paper is organized as follows. Section 2 enumerates the SDC in details. Section 3 evaluates the performance of SDC through a number of simulations. Section 4 discusses related work. Finally, we conclude by summarizing our findings and identify future research opportunities in Sect. 5.

## 2. SDC Design

SDC is based on the hierarchical trees where the sinks are root nodes. Every sensor must be a member of the tree, i.e., an internal or leaf node, in order to communicate with the sink and it may act as a router if necessary.

The design of SDC has been driven by the following goals.

- **Simplicity and Scalability.** Since unconstrained scale is an inherent feature of a sensor network and the sensors have limited computing capability as well as memory resources, we seek to minimize the number of operations performed and the states maintained at each sensor. In particular, each node does not maintain all neighboring nodes and the path calculation is not based on the complex algorithms such as Dijkstra's or Bellman-Ford algorithm [9].

- **Robustness.** Our solution provides self-organized mechanisms in order to deal with the dynamic natures of

\*"Base station," "sink node," and "sink" are used interchangeably throughout the paper.

sensor networks, i.e., joining and leaving scenarios.

This section describes the details of SDC which satisfy the above mentioned goals. Section 2.1 presents the network model we consider. Section 2.2 explains how to construct the tree. Sections 2.3 and 2.4 deal with the dynamics of the network. Section 2.5 describes data collection using the tree. The cycle of all sensors, except the sinks, starts from the joining mechanism (Sect. 2.3) where the nodes attempt to attach to the tree.

## 2.1 Network Model

We consider a network composed of a small number of sink nodes and a numerous number of wireless sensor nodes randomly distributed in an interesting area. These sensors have limited processing power, storage, bandwidth, and energy, while the sinks have powerful resources so as to perform any tasks or communicate with the sensors. In particular, the sensor nodes have omni-directional antennas and use RF to communicate. We assume that the sensors are not mobile nodes, i.e., all sensors are fixed for the duration of their lifetime, however, the sensor network we consider has the dynamic natures described so far.

We design a routing protocol for wireless sensor networks whose communication pattern differs from conventional mobile ad hoc networks. Let  $N$  and  $BS$  be a set of sensors and base stations respectively. SDC is a multipoint-to-point protocol for communicating parties  $(s, d)$ , where  $s \in \{N\}$  and  $d \in \{BS\}$ , namely, every sensor tries to report sensed data to the nearest sink which is a concept of *anycast* communication. Unlike SDC, previous works [10]–[16] are point-to-point routing protocols, i.e.,  $s, d \in \{N, BS\}$ .

The format of the packets used in SDC is as follows:  $\langle type, ID_{src}, ID_{dst}, ID_{grp}, seq, len, data \rangle$ . The *type* field is used to specify the type of the packet. The  $ID_{src}$ ,  $ID_{dst}$ , and  $ID_{grp}$  fields are the source ID, destination ID, and group ID respectively. The group ID is an optional field, it is used to distinguish the redundant trees created by different base stations. Thus, we can use a base station ID as a group ID. The *seq* field is a sequence number of the packet. The *len* field is the length of the packet and the *data* field is used for carrying any data.

## 2.2 Constructing the Tree

A base station initiates the tree construction by broadcasting a *child request* (CRQ) packet to discover the child nodes. *Nonmember*, a node which does not attach to the tree yet, determines its parent from received CRQ packets by choosing a node whose CRQ packet has arrived first as a parent or waiting for a short period of time ( $T_{crq}$ ) in order to collect a number of candidates and choose a node whose defined metric is the best one (highest received signal strength, highest remaining energy, for example). The candidates are kept in a *parental candidate* (PC) table which maintains the pairs of candidate ID and metrics. A *Member* which is an internal or leaf node of the tree also updates the PC table accord-

ing to incoming CRQ packets. We can also limit the size of the PC table by deleting the stale information in the case of overflow. In our simulations, we employ the latter method, i.e., a nonmember waits to collect the candidates and then chooses a parent as follows. Let us assume  $crq\_time$  and  $joined\_time$  denote the time that a node first received a CRQ packet and the time that a node joined the tree respectively. The nonmember chooses a node whose  $crq\_time$  is the minimum and if many nodes have the same value of this metric, it will choose a node whose  $joined\_time$  is the minimum. Less time implies the node is nearer the base station which results in the shorter path. We decide to use time related metric because it is easy to implement comparing to measuring received signal strength or the amount of remaining energy. Since the signal strength is affected by milieu, it is difficult to measure the exact value. On the other hand, the remaining energy is a variable metric, i.e., the energy decreases along the time. Even though SDC does not require strict time synchronization, we can employ the approaches proposed in [17] to achieve clock synchronization. Based on our simulations, the proposed metrics ( $crq\_time$  and  $joined\_time$ ) and the signal strength have comparable performance.

After choosing the parent, the nonmember sends a *child reply* (CRP) packet to the selected parent so as to inform that it will be a child node or a leaf node of the current tree. Upon receiving the CRP packet, the parent node confirms an acceptance of a new child by replying with a *child acceptance* (CAC) packet. If the child node does not receive the CAC packet within a period of  $T_{cac}$ , it will retransmit the CRP packet. This retransmission is performed two times. After three timeouts, it will choose a new parent from the PC table. If the PC table is empty, it will use the joining mechanism to discover a new parent (Sect. 2.3). After receiving the CAC packet from the parent, the child node does the same processes as its parent, i.e., broadcasting a CRQ packet to discover its child nodes. Note that the  $joined\_time$  is the time when a node received the CAC packet. These procedures are performed by every node throughout the network. To avoid the problem of communication gray zones reported in [18], the node should choose a node whose received signal strength is greater than an appropriate threshold. An example of the trees constructed by SDC is shown in Fig. 9.

## 2.3 Adding the Nodes

A newly deployed node finds a parent by using a *joining mechanism* as follows. A joining node broadcasts a *parent request* (PRQ) packet making the neighboring nodes aware of its existence. Any members of the tree that hear this packet reply by unicasting a CRQ packet to the joining node. Note that this CRQ packet is same as described in Sect. 2.2 except that we use unicasting instead of broadcasting. Then, the processes will follow the tree construction phase, i.e., the joining node sends a CRP packet to the selected parent and waits for a CAC packet as a confirmation of their rela-

tion. If the joining node does not receive any CRQ packet after broadcasting the PRQ packet, it infers that no any node is within its radio coverage or all of its neighboring nodes are not the members. In this case, it waits for incoming CRQ packets after one of its neighbors has become a member. As an option, the joining node can broadcast the PRQ packet periodically until receiving the CRQ packets. We note again that every sensor starts their cycle from the joining mechanism (i.e., broadcasting the PRQ packet first) while the sinks broadcast the CRQ packet first (Sect. 2.2).

## 2.4 Dealing with Node Failures

A *leaving mechanism* described in this section is used to reconstruct the tree if an internal node has failed due to numerous reasons. For instance, the battery of the node is depleted with the time, the node can be damaged due to harsh environment or by the enemy. The tree in SDC is self-organized and it is reconstructed on-demand, i.e., whenever the node has data to send. A detection of such failed nodes relies on the underlying MAC layer protocol. If the acknowledgement on MAC layer does not arrive, the node infers that a communicating party (which is its parent) has left from the network.

When an *orphaned node* is aware of the absence of its parent, it immediately switches to a new parent by choosing the most appropriate one from the PC table. This can be done by sending a CRP packet to a newly selected parent and waiting for a CAC packet. If there is no any candidate in the PC table, it behaves as if it is a newly deployed node by following the joining mechanism (Sect. 2.3). However, its child and grandchild nodes will not reply to this PRQ packet to prevent routing loop. Based on all of simulations done in Sect. 3, two-hop information is enough for dealing with routing loop problem. Every node beneath the orphaned node does nothing because they are not aware of the absent node (the left parent of the orphaned node). They still forward the packets to the orphaned node as usual, and the orphaned node keeps received packets in the buffer for sending later. In the worst case that the orphaned node does not have any parent in the PC table and no any response to the PRQ packet, it sends a *parent query* (PQR) packet to its child nodes asking whether they have any candidate for a parent. The child nodes reply with a *parent reply* (PRP) packet containing such information. Then, the orphaned node randomly chooses a child that has at least one candidate as its new parent by sending a *reverse* (REV) packet to inform a new relation, and that selected child will switch to a new parent chosen from the PC table. If all of its child nodes do not have any candidate for a parent, the orphaned node randomly chooses one child as a new parent by sending a REV packet and let the selected child find a new parent using the joining mechanism (Sect. 2.3). Note that the last scenario is quite a rare case that may occur in highly sparse sensor networks.

## 2.5 Collecting Data and Discussions

When the network size is larger, it is impossible to use only one sink even though we have an optimal protocol because

```

1: void main() {
2:   num_crp ← 0 // the number of CRP packets sent
3:   flag_prt ← DOWN // status of parent
4:   flag_crq_sent ← NO // status of CRQ packet sent
5:   flag_crq_rcv ← NO // status of CRQ packet received
6:   flag_crp_sent ← NO // status of CRP packet sent
7:   flag_selprt_call ← NO // status of calling select_parent()
8:   crq_time ← ∞ // set to infinity or some high value
9:   joined_time ← ∞ // set to infinity or some high value
10:  BS broadcasts CRQ packet
11:  Sensor broadcasts PRQ packet
12:  while rcv_pkt do // rcv_pkt: received packet
13:    if rcv_pkt is data packet then
14:      if destination is sink node then
15:        if flag_prt = UP then
16:          forward rcv_pkt to parent
17:          if link-layer detects failed link then
18:            link_failed()
19:          end if
20:        else // (flag_prt = DOWN || flag_prt = REPAIR)
21:          buffer rcv_pkt in queue
22:        end if
23:      end if
24:    else // rcv_pkt is control packet
25:      if rcv_pkt is CRQ packet then
26:        if (flag_prt = UP) || (my_addr = BS) then
27:          drop CRQ packet
28:        else
29:          pc_table ← (ID_src, crq_time, joined_time)
30:          if flag_crq_rcv = NO then
31:            crq_time ← CURRENT.TIME
32:            flag_crq_rcv ← YES
33:          end if
34:          if flag_selprt_call = NO then
35:            flag_selprt_call ← YES
36:            call select_parent() at T_crq seconds later
37:          end if
38:        end if
39:      else if rcv_pkt is CRP packet then
40:        if (flag_prt = UP) || (my_addr = BS) then
41:          send CAC packet
42:        end if
43:      else if rcv_pkt is CAC packet then
44:        flag_crp_sent ← NO
45:        num_crp ← 0
46:        joined_time ← CURRENT.TIME
47:        send all buffered packets in queue to parent
48:        if flag_prt = DOWN & flag_crq_sent = NO then
49:          broadcast CRQ packet
50:          flag_crq_sent ← YES
51:        end if
52:        flag_prt ← UP
53:      else if rcv_pkt is PRQ packet then
54:        if (my_addr = BS) || ((flag_prt = UP) & (ID_src ≠ parent) & (ID_src ≠ grandparent node)) then
55:          unicast CRQ packet
56:        end if
57:      else if rcv_pkt is PQR packet then
58:        send PRP packet
59:      else if rcv_pkt is PRP packet then
60:        randomly choose a new parent from its children
61:        send REV packet to selected parent
62:      else if rcv_pkt is REV packet then
63:        if pc_table is not empty then
64:          select_parent()
65:        else
66:          broadcast PRQ packet
67:        end if
68:      end if
69:    end if
70:  end while
71: }

```

Fig. 1 Main algorithm.

```

1: void sendCRP() {
2: send CRP packet to parent
3: flag_crp_sent ← YES
4: num_crp ← num_crp + 1
5: call wait_cac() at T_cac seconds later
6: }
1: void select_parent() {
2: flag_selprt_call ← NO
3: if BS is in pc_table then
4: parent ← BS
5: else
6: for all members in pc_table do
7: parent ← node whose crq_time is the minimum
8: if crq_time is equal then
9: parent ← node whose joined_time is the minimum
10: end if
11: end for
12: end if
13: sendCRP()
14: }
1: void wait_cac() {
2: if flag_crp_sent = YES then
3: if num_crp > 3 then
4: pc_table ← pc_table - parent
5: parent ← NULL
6: num_crp ← 0
7: if pc_table > 0 then
8: select_parent()
9: else
10: periodically broadcast PRQ packet until getting CRQ packet
11: end if
12: else // num_crp ≤ 3
13: sendCRP()
14: end if
15: end if
16: }
1: void link_failed() {
2: buffer packets in queue
3: flag_prt ← REPAIR
4: pc_table ← pc_table - parent
5: parent ← NULL
6: if pc_table > 0 then
7: if crq_time of at least one member in pc_table is lesser than own crq_time
then
8: select_parent()
9: else
10: broadcast PRQ packet
11: num_crp ← 0
12: end if
13: else // pc_table = NULL
14: broadcast PRQ packet
15: num_crp ← 0
16: end if
17: }

```

Fig. 2 The functions called by the main algorithm.

the traffic will concentrate around the sink incurring high loss rate. Thus, the user can deploy the sinks at some ratio compared to the number of sensors in order to distribute the loads. The sensors can operate with multiple sinks seamlessly without any change in our protocol. Although the sensors do not know the address (or ID) of the nearest sink, it should be a member of the tree created by a *potential* nearest sink because the CRQ packet from such sink should arrive first. The nodes can use the group ID to distinguish different sinks. Thereby, they can be a member of multiple trees in order to achieve robustness against the failed nodes, i.e., multipath routing is supported. To collect the data, each node just forwards its sensed data and all of received data to its parent. If it does not attach to the tree yet, it keeps such data in the buffer and send them later. The algorithms described thus far are summarized as pseudo-codes in Figs. 1 and 2.

A state transition diagram of a sensor is also illustrated

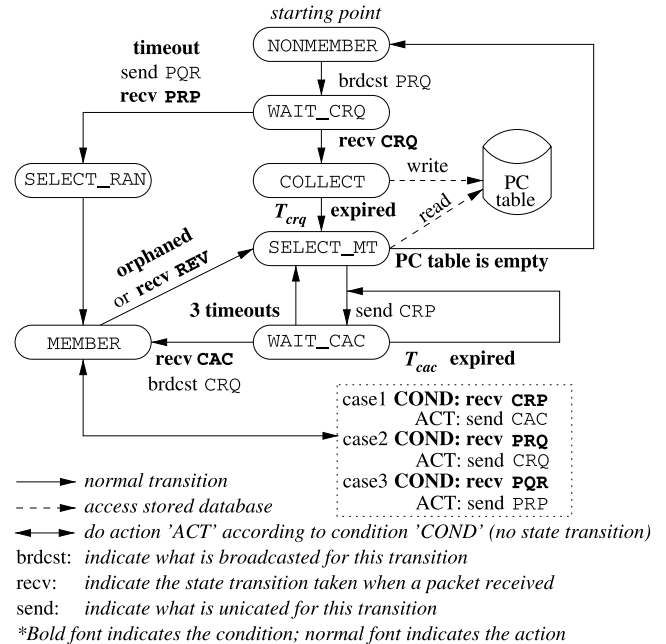


Fig. 3 State transition diagram.

in Fig. 3. The sensor starts from a NONMEMBER state where it broadcasts a PRQ packet to discover a parent and waits for a CRQ packet at a WAIT\_CRQ state. After collecting the candidates for a period of  $T_{crq}$  at a COLLECT state, it selects a parent based on the appropriate metrics at a SELECT\_MT state. A WAIT\_CAC is a state where the node waits for a CAC packet before entering a MEMBER state. At a SELECT\_RAN state, the node randomly selects one of the child nodes as a new parent.

Each node in SDC relies only on the knowledge of a parent, a grandparent, and a PC table which should be small enough to keep in the node itself. However, the PC table is an optional storage because it is decomposable information. The nodes can attach to a new parent faster with the help of the PC table but the information in the PC table may be stale, i.e., a newly selected parent is also a left node. If the nodes always broadcast the PRQ packet to discover a new parent without relying on the PC table, it will get fresh information. Since the states required on each node are constant (a fix-sized PC table) and they are independent of node density as well as network size, SDC is highly scalable. Every node in SDC broadcasts only once to discover the route, thereby no propagation of routing packets issued by each node throughout the network. In other words, the routing packets are limited to one-hop neighboring nodes. Moreover, SDC does not apply periodic updating that reduces traffic load so much. Furthermore, geographical information is also not necessary.

### 3. Performance Evaluation

To evaluate the performance of SDC, we use the *ns-2* [19] simulation tool to run a number of simulations described

in this section. We then compare the performance with the following protocols: (1) a proactive ad hoc routing protocol, i.e., Optimized Link State Routing (OLSR) protocol [11], (2) two reactive ad hoc routing protocols, i.e., Dynamic Source Routing (DSR) protocol [12] and Ad-Hoc On-Demand Distance Vector (AODV) protocol [13], and (3) a communication protocol for sensor networks, i.e., Directed Diffusion (DD) [20]. OLSR uses multipoint relaying technique to suppress redundant control messages. Hence, it is one promising protocol amongst other proactive protocols. DSR and AODV are two well-known protocols which show favorable performance over other protocols [21]. DD is a representative communication paradigm tailored for sensor networks. It is worth to compare with DD because it can do multipoint-to-point communication. We also ran the simulations on DSDV routing protocol [10] which shows worse performance than AODV and DSR as also reported in [21]. Therefore, we do not include the results of DSDV in this paper.

### 3.1 Simulation Environment

The *ns-2* simulator includes full simulation of the IEEE 802.11 physical and MAC layers. Our simulations use this MAC layer and assume symmetric links. Using this MAC layer does not affect the evaluation because we need to evaluate the network layer of five protocols. We randomly placed 50 sensors in a 200 m by 200 m square region. Each node has fixed radio coverage of 50 meters. All nodes have fixed positions without any movement for the entire simulation, i.e., *immobile* nodes. We use constant bit rate (CBR) as our traffic sources. These CBR sources send 64-byte data packets at the rate of 0.25, 0.5, 1, and 2 packet per second, i.e., 128, 256, 512, and 1,024 bps respectively, while the bandwidth of the sensor nodes is set to 19.2 kbps. The CBR agent will be attached to a UDP agent, which in turn attached to the source node. For all simulations, the communication patterns are peer-to-peer and the starting time of each connection is randomly chosen between 0 and 50 second. One node from each simulation is randomly selected as a base station and it is the only destination for all traffic sources, while all remaining nodes (49 nodes) are the source nodes (one flow per one source).

We run each scenario on five differing topologies so as to achieve the reliability of the results. We show the detailed results of each topology as well as the average values of five topologies. Each simulation is run for 1,800 simulated seconds.

The parameters of SDC used in the simulations are set as follows:  $T_{crq} = 0.1$  second (the time waiting for collecting the candidates), and  $T_{cac} = 0.3$  second (the time waiting for a CAC packet from a selected parent).

### 3.2 Performance Metrics

To compare with other protocols, we choose to evaluate them according to the following three metrics.

- *Packet delivery ratio* (PDR): the ratio between the number of data packets received by the destination and the number of data packets sent by the source.

- *Average delay*: the average end-to-end delay observed between transmitting a data packet and receiving it at the destination.

- *Average path length*: the average number of hops a data packet took to reach the destination.

PDR is important as it shows the loss rate seen by the transport protocols. It also affects the maximum throughput that the network can support. This throughput can be investigated by increasing transmission rate. PDR characterizes both the completeness and correctness of the protocol.

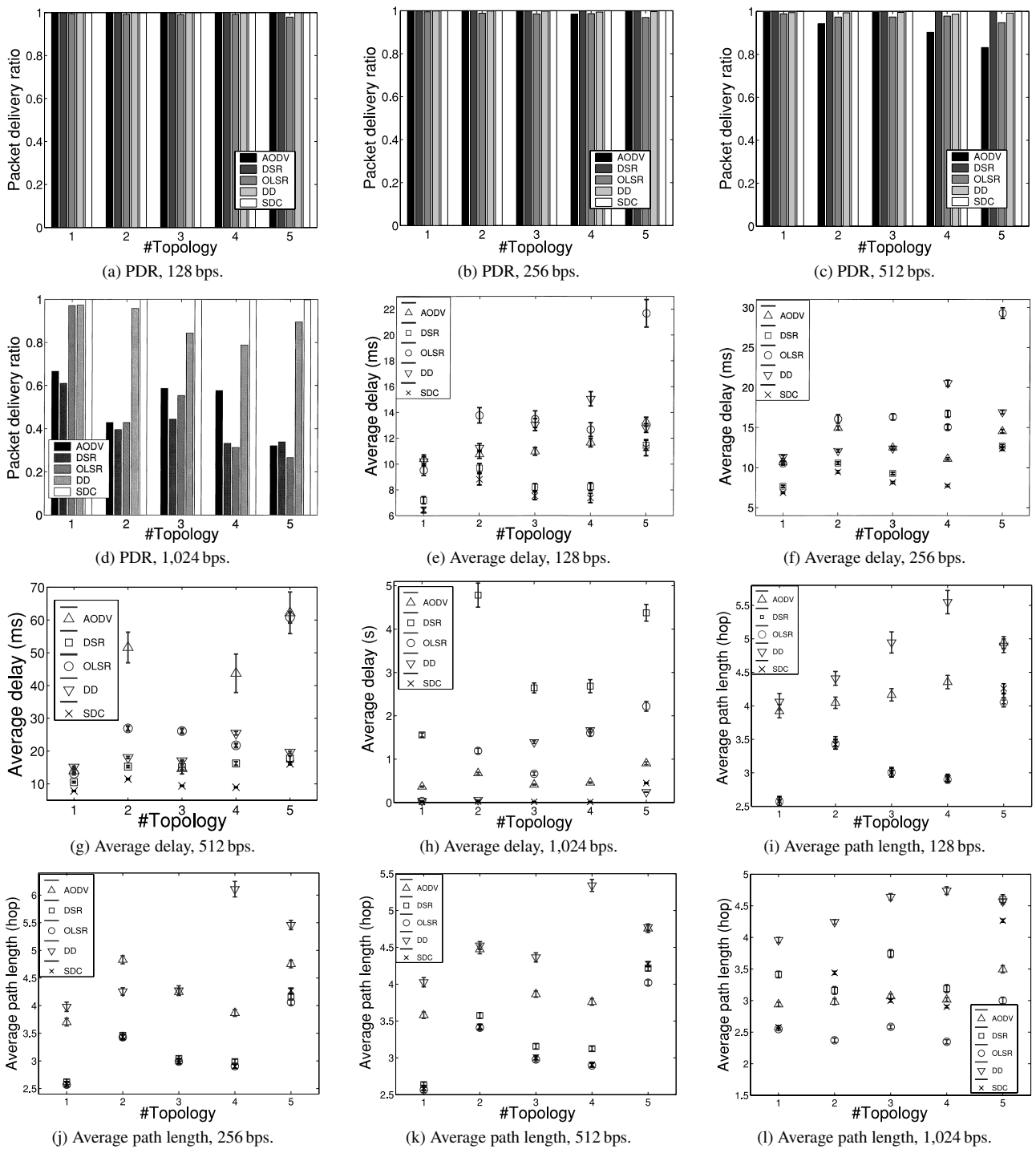
Average delay is an important metric for comparing as it measures the quality of path decided by routing algorithm. Protocols that send large number of routing packets can also increase the probability of packet collisions and may delay data packets by queuing them in the buffer.

Average path length measures the ability of the protocol to efficiently use network resources by selecting the shortest path from a source to a destination. Since a main cause of energy consumption in sensor networks is communication cost compared to computational cost (transmitting a single bit of data is equivalent to 800 instructions [22]), the path length implies dissipated energy.

We also use *tree convergence time* as a metric to confirm our simulation setup. This metric measures a period of time between sending a first control packet until the tree construction has finished. It can be used to evaluate the efficiency of tree construction procedures, i.e., how fast all source nodes can discover a route to the destination.

### 3.3 Performance Comparison in Static Networks

The first set of the simulations uses five differing topologies with four offered loads. The comparisons of five protocols on three performance metrics are illustrated in Fig. 4. Table 1 also shows the values averaged over five topologies for each protocol. PDRs are very similar (nearly 100%) for all of five protocols at 128- and 256-bps traffic load (Figs. 4(a) and 4(b)), except OLSR whose average PDR is 1% less than others. With 512-bps load (Fig. 4(c)), however, PDRs of AODV obviously drop to 93% at average while the others still perform well. When we offer 1,024-bps load (Fig. 4(d)), only SDC can deliver nearly 100% of the originated data packets. AODV, DSR, and OLSR deliver less than 70% at this offered load but it seems that AODV and OLSR outperform DSR, i.e., approximately 50% of AODV and OLSR compares to 44% of DSR at average. DD does better than three ad hoc routing protocols (OLSR, AODV, and DSR) but it is not so good as SDC, i.e., it delivers 89% of the originated data packets. With high offered load, most of the data packets drop at the nodes around the sink due to high congestion. Since the sink in DD *periodically floods* interest messages to the entire network, the possibility of channel contention increases, resulting in dropped packets when the buffer is overflow. Moreover, data packets are



**Fig. 4** Performance comparison in 50-node networks. The average values of five topologies are shown in Table 1.

**Table 1** Performance metrics averaged over 5 topologies from the results in Fig. 4.

Load (bps)	Packet delivery ratio				Average delay (ms)				Average path length (hop)			
	128	256	512	1024	128	256	512	1024	128	256	512	1024
AODV	0.9999	0.9969	0.9308	0.51	11.6	12.9	38.3	574	4.30	4.29	4.08	3.08
DSR	0.9992	0.9992	0.9992	0.42	9.1	11.6	15.5	3289	3.23	3.26	3.36	3.68
OLSR	0.9898	0.9848	0.9722	0.51	14.2	17.4	29.8	1143	3.19	3.19	3.17	2.57
DD	0.9988	0.9970	0.9921	0.89	12.5	14.7	19.1	678	4.77	4.82	4.61	4.43
SDC	0.9999	0.9999	0.9999	0.99	8.2	8.8	10.8	105	3.24	3.24	3.24	3.24

initially sent *redundantly* along multiple paths which is another cause of contention. Although OLSR suppresses some redundant floodings, every node is required to periodically exchange control messages which incurs a large volume of traffics. These congestions force a node to find a new available route. When a node in DD detects failed or degraded paths, all nodes downstream (towards the sink) of the lossy link will apply the reinforcement rules, i.e., sending interest messages. For four left protocols, when a node cannot deliver a packet to the next hop due to congestion, it implies the absence of the next-hop node. In such case, DSR and AODV will flood a route request (RREQ) packet to discover a new route. These control packets incurred by DD, DSR, and AODV lead to a chaos of the network which in turn incurs more congestions and more packets dropped. In contrast, SDC discovers a new route by broadcasting which is limited to only one hop. Thereby, SDC is more robust to higher offered load than other four protocols.

We also show 99% confidence interval for the delay and path length in Fig. 4. SDC has better delay than other four protocols for all offered loads. The differences of delay increase for higher offered loads. OLSR has the highest delay at 128- and 256-bps load (Figs. 4(e) and 4(f)) because a node cannot transmit the packets immediately due to high contention with control traffics. DD must take an amount of time in order to settle in a steady state due to the reinforcement procedure. Since this is a fixed cost regardless of the offered load, DD has higher delay than AODV, DSR, and SDC at 128- and 256-bps load. Besides, the periodic interest messages mentioned above also defer data messages. Although DSR and SDC have comparable path length at 128- and 256-bps load (Figs. 4(i) and 4(j)), DSR has slightly higher delay at this offered load. The reasons are twofold due to the impact of flooding used by DSR. First, congestion often occurs in DSR as an effect of a number of control packets. Therefore, DSR wastes time to retransmit the collided packets. Second, SDC discovers a new path faster than DSR because it uses one-hop broadcasting and it also has the reserved parents in the PC table. It is not surprising that DSR has higher delay than SDC at 512- and 1,024-bps load (Figs. 4(g) and 4(h)) because it takes a little longer path than SDC (Figs. 4(k) and 4(l)) and the effect of congestion is much severer than light-load situation. AODV has higher delay than DSR and SDC as a result of longer path, i.e., approximately 1 hop longer. The exception is at 1,024-bps load where AODV takes shorter path than DSR, therefore it has lower delay than DSR. When we examine in the details, most of dropped packets in AODV at this load are the packets destined to a distant node while DSR and DD do better than AODV in delivering such packets. Hence, such two protocols (DSR and DD) take longer time to deliver the packets in average. The result of shortest path used by OLSR at 1,024-bps load can be explained in the same way as AODV, i.e., almost of packets destined to a distant node are not delivered. DD uses longest path because it selects an empirically low delay path which may not be the shortest path. In other words, DD avoids a short path which has high

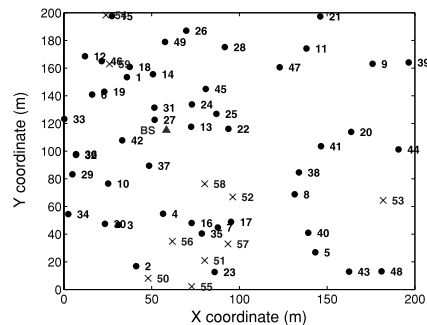


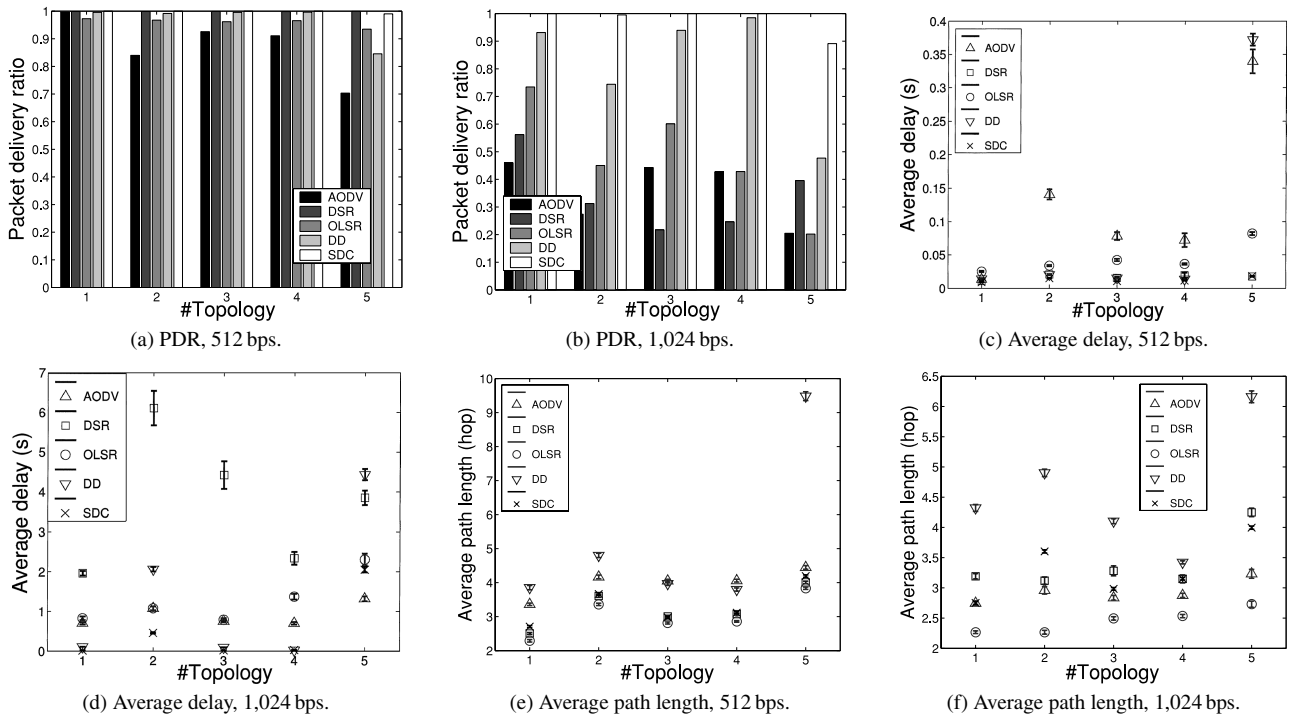
Fig. 5 A topology changes when 10 nodes (indicated by the cross) join or leave the network.

congestion. Moreover, DD is very sensitive to the changes in path quality (e.g., congestion), therefore, an intermediate node often initiates the reinforcement procedure which is a main cause of congestion as described above. OLSR takes the shortest path (only 0.05 hop shorter than SDC) because any link state protocols know the topology of the entire network so that it can calculate an optimal path which traverses the minimum number of hops, however, its delay is the highest as explained above (contention with control traffics). This can also imply that the tree created by SDC is nearly optimal in the viewpoint of path length and it is quite efficient when considering PDR and end-to-end delay.

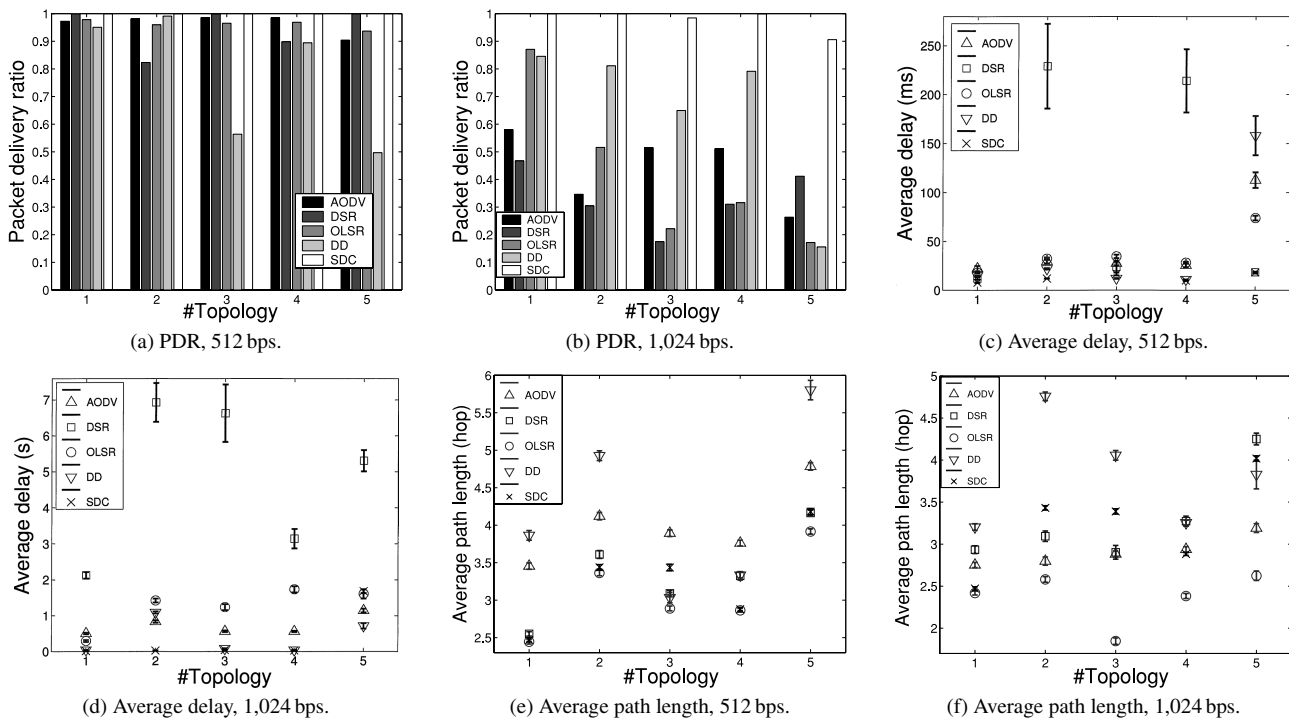
### 3.4 Effect of Network Dynamics

We also simulate the dynamic scenarios of sensor networks, i.e., joining and leaving scenarios. To simulate the joining scenarios, 10 nodes are randomly deployed at the half-way of the simulations (900th sec.) in addition to 50 nodes being deployed from the beginning of the simulations. For the leaving scenarios, we deploy 60 nodes at the beginning of the simulations and apply an energy model by providing much enough energy for 50 nodes and making the battery of 10 random nodes depletes around the half-way of the simulations. We choose heavy offered loads, namely, 512- and 1,024-bps load, to make the situations fairly challenging for the protocols. These experiments are used to evaluate the robustness and resilience of the protocols against network dynamics. Note that 10 nodes means approximately 20% of nodes have dynamic nature. This dynamic ratio follows the work done in [16]. An example of topological change according to these dynamic natures is shown in Fig. 5 which is the first topology used in Figs. 6 and 7. The nodes numbered 50–59 (indicated by the cross) are considered as joining or leaving nodes in the experiments.

Figure 6 presents the simulation results of joining scenarios and Table 2 also shows the values averaged over five topologies for each protocol. With 512-bps load (Fig. 6(a)), PDRs of SDC, DSR, and OLSR are consistent with those of static networks, while PDRs of DD and AODV drop 3% and 6% respectively. When we offer 1,024-bps load (Fig. 6(b)), SDC still achieves excellent PDR, i.e., 98%, while PDRs of other four protocols are lower than static scenarios because



**Fig. 6** Performance comparison when the nodes join the networks. The average values of five topologies are shown in Table 2.



**Fig. 7** Performance comparison when the nodes leave the networks. The average values of five topologies are shown in Table 3.

additional 10 nodes increase the total traffic loads which incur more congestion in the networks. The average delays of all protocols (Figs. 6(c) and 6(d)) increase as a result of additional data traffics and additional time required to dis-

cover a path for the joining nodes. The average path lengths slightly decrease as the node density increases (the number of nodes increases while the area is kept constant), except DD and SDC. As mentioned above, DD chooses a path ac-

**Table 2** Performance metrics averaged over 5 topologies from the results in Fig. 6.

Load (bps)	PDR		Delay (ms)		Path length	
	512	1024	512	1024	512	1024
AODV	0.8748	0.35	131	915	4.03	2.95
DSR	0.9985	0.34	17	3745	3.25	3.41
OLSR	0.9605	0.48	44	1277	3.03	2.46
DD	0.9649	0.82	87	1345	5.18	4.58
SDC	0.9983	0.98	13	512	3.26	3.25

**Table 3** Performance metrics averaged over 5 topologies from the results in Fig. 7.

Load (bps)	PDR		Delay (ms)		Path length	
	512	1024	512	1024	512	1024
AODV	0.9245	0.42	47	748	4.01	2.93
DSR	0.9418	0.32	99	4855	3.34	3.31
OLSR	0.9619	0.42	37	1261	3.10	2.37
DD	0.7994	0.67	44	394	4.19	3.82
SDC	0.9999	0.98	13	355	3.26	3.24

According to the delay and SDC decides a path based on the *crq\_time* and *joined\_time*, therefore, both protocols do not always use the shortest path. A comparison of delay and path length between each protocol can be discussed in the same way as static networks.

Figure 7 illustrates the results of leaving scenarios and Table 3 also shows the values averaged over five topologies for each protocol. SDC still has the superior PDR when some nodes leave the networks (Figs. 7(a) and 7(b)). PDRs of other protocols drop when comparing to those of static scenarios. We notice that PDRs of DD decrease more than 20% because local repair is done by all nodes downstream of the left node, i.e., all nodes towards the destination. Such all nodes apply negative reinforcement rules to discover a new path by flooding the new interests. Instead of using all downstream nodes, only one upstream node, which is employed by other four protocols, is better to initiate the local repair as the results shown. Consequently, the delay of DD at 1,024-bps load (Fig. 7(d)) is lower than the static case due to highly dropped packets destined to a distant node. Figure 7(f) also affirms this fact, i.e., the packets take shorter path compared to the result of static network. The average delays of all cases (except the one mentioned above) are higher than those of static scenarios because every protocol must take an amount of time in detecting a left node and finding a new path, while the average path lengths are a little shorter than static networks due to higher node density in dynamic scenarios. We conclude that SDC still achieves a good performance in dynamic sensor networks.

It may be difficult to compare confidence interval between static and dynamic networks because of very short bars in the figures. Moreover, the ranges of y-axis in Figs. 6 and 7 differ from Fig. 4. Therefore, Table 4 shows the confidence intervals of delay at 1,024-bps load averaged over five topologies. Each row (static, joining, and leaving case) in the table is the averages of Figs. 4(h), 6(d), and 7(d) respectively. We found that the delays of dynamic networks fluctuate in a wider range than those of static networks be-

**Table 4** Average confidence interval (ms) of delay at 1,024-bps load.

	AODV	DSR	OLSR	DD	SDC
Static case	20.98	156.39	58.99	16.39	2.68
Joining case	39.66	235.63	71.32	40.10	17.80
Leaving case	29.72	400.00	75.54	22.82	11.96

**Table 5** Average confidence interval (hop) of path length at 1,024-bps load.

	AODV	DSR	OLSR	DD	SDC
Static case	0.042	0.063	0.035	0.045	0.023
Joining case	0.049	0.065	0.032	0.057	0.024
Leaving case	0.040	0.064	0.036	0.070	0.025

**Table 6** Tree convergence time (second).

Static-128	Static-256	Static-512	Static-1024
0.86	0.86	0.92	0.87
Join-512	Join-1024	Leave-512	Leave-1024
0.99	0.99	0.95	0.95

**Table 7** Maximum time (second) a new node used to discover a parent.

Topology No.	#1	#2	#3	#4	#5
512-bps load	0.64	1.10	0.84	0.96	0.89
1,024-bps load	0.95	1.36	1.23	1.17	2.42

**Table 8** Maximum time (second) an orphaned node used to discover a new parent.

Topology No.	#1	#2	#3	#4	#5
512-bps load	1.09	0.68	2.06	1.42	2.39
1,024-bps load	2.79	0.72	5.79	4.72	6.64

cause the protocols require an amount of time in order to adapt to a new topology. In contrast, the confidence intervals of path length (Table 5) do not differ between static and dynamic cases because the path length depends on network topology or node density, not dynamic natures. The exception is DD which select low delay path.

We assume the battery of nodes depletes around the half-way of the simulations, i.e., 900th second. In other words, the nodes are alive for only 15 minutes which is short compared to the actual sensor nodes. However, such simulation setup does not matter with our experiments because we run the simulations long enough until the networks have been in a steady state, i.e., every node knows a next-hop node to which it will send the packets. The tree convergence time of SDC can be used to prove the above statements. Table 6 shows the convergence time averaged over five topologies for each simulated scenario. Note that we also show the results of static networks as a reference. The results show that SDC uses less than one second to finish the tree construction in both 50- and 60-node networks. When we deploy additional 10 nodes in the current network, Table 7 shows that a new node discovers a parent within 2.5 seconds. In the case of leaving scenarios, an orphaned node discovers a new parent within seven seconds as shown in Table 8. The maximum value in the tables means the longest time used by a node over 10 joining/leaving nodes in each

topology. These values are much less than the simulation time (1,800 seconds). Moreover, previous works also did the simulations shorter than ours [14], [20], [21], [23]. Furthermore, dying faster infers the severe situation. Therefore, a protocol which can tolerate a short-period (or severe) experiment should work well in the actual situation where the sensors die slower.

### 3.5 Varying the Network Size

The last set of the simulations uses differing number of nodes with two offered loads. As the network size (or working region) increases, we keep the node density and the ratio between the sink and the sensor (1:49) constant as detailed in Table 9. We simulate only two moderate loads (128- and 256-bps loads) due to limited computational resource. As mentioned in Sect. 2.5, the sensor nodes in SDC can discover a potential nearest sink without the knowledge of the sink address or geographical information. Hence, we do not run the simulations on AODV, DSR, and OLSR because the source nodes must know the destination address (the nearest sink) in advance. On the other hand, the sinks in DD must

determine the location of interests in advance, namely, geographical information is required. Since we have shown that SDC has better performance than other protocols in one-sink networks and the objective of this experiment is to show the scalability of SDC, we run only SDC on differing network sizes.

Figure 8 shows PDRs as a function of network size. Each data point in this figure represents an average of five runs with different randomly generated topologies. The results show that SDC is highly scalable because it can deliver more than 99.98% of the originate packets when the number of nodes is increased from 50 to 200 nodes. Figure 9 shows an example of trees created by SDC on 100-node network (two sinks with 98 sensors).

## 4. Related Work

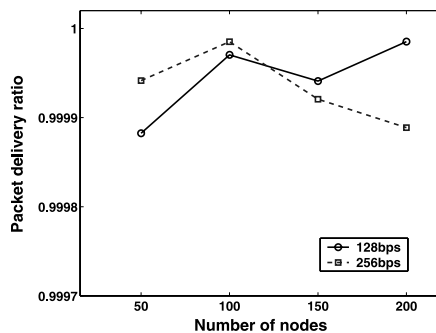
A number of routing protocols [24] has been developed to provide efficient and robust communication in mobile ad hoc network (MANET) which is different from wireless sensor network (WSN) in the issues of node mobility and network size. However, many MANET routing protocols can work well in immobile networks [21]. To provide robustness against changing topology, the proactive routing protocols (e.g., DSDV [10], OLSR [11]) must periodically update the states to follow current physical topology. Although, OLSR uses multipoint relaying technique to economically flood control messages, it still employs periodic exchange of messages which is a main cause of congestion and collision. Moreover, each node must maintain the routing table for the entire network which is not suitable for limited-memory sensor. On the other hand, the reactive approaches (e.g., DSR [12], AODV [13]) try to reduce a cost of periodic updating by discovering a route on-demand. Nonetheless, flooding used in such protocols still incurs congestion shown in our simulation results and it takes time to discover a new available route. Therefore, we propose an alternative which is more light-weight and tailored to specific communication required in WSN. Every node in SDC broadcasts only once to discover the route which obviously incurs less overhead than flooding and periodic updating.

Location-based routings can also apply well to WSN. LAR [15] and GPSR [14] are geographic routings that use location information to decrease overhead of route discovery and find routes quickly. Unfortunately, such approaches may not implement into many sensor networks because each sensor node requires exact geographic location. Current methods of determining geographic location [25]–[27] consume much energy and may not be possible in many sensor network scenarios. Moreover, location-aware device increases production cost for sensor nodes, especially in large-scale sensor networks. Hence, we do not rely on location information.

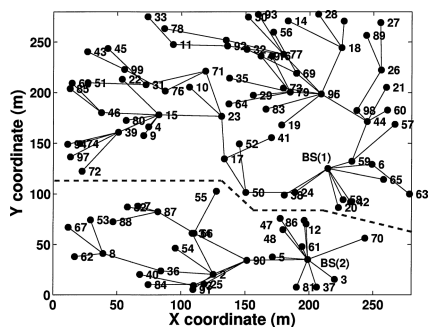
Directed Diffusion [20] is a data-centric routing protocol based on the name of data. The sinks draw interesting information by periodically flooding interests and setting up gradients within the network. These periodic interests

**Table 9** Simulation setup when varying the network size.

#node	region (m × m)	#sinks	#sensors
50	200 × 200	1	49
100	280 × 280	2	98
150	345 × 345	3	147
200	400 × 400	4	196



**Fig. 8** The fraction of data packets successfully delivered as a function of number of nodes.



**Fig. 9** Two sinks (BS(1) and BS(2)) and 98 sensors in 280×280 m square region.

waste the resources and disturb the delivery of data packets as done by periodic updating in proactive protocols. Moreover, Directed diffusion requires location information which may not be possible as described above.

Ye et al. [28] proposed a Minimum Cost Forwarding algorithm for collecting data in sensor networks. Each node maintains the least cost estimate from itself to the base station. Data packets are forwarded by broadcasting which does not guarantee reachability. Moreover, it wastes energy because the broadcast packets are received by every neighboring node. To deal with node failures, the authors proposed to increase the cost budget at the source node which results in non-optimal path. Another simple solution is to refresh the cost field which consumes both energy and time.

Woo et al. [29] introduced a many-to-one routing protocol based on link connectivity statistics which must be estimated on-line. Thereby, the node must listen for all packets, including the packets that are not addressed to it. This estimation also comes at computational and communication cost as well as memory storages.

Low Energy Adaptive Clustering Hierarchy (LEACH) [33] is a multipoint-to-point communication protocol aiming to achieve energy efficiency. A cluster head is self-elected and randomized rotation of cluster heads is employed to balance energy consumption. However, the authors do not consider the dynamics of network which are the important natures in WSNs. Moreover, the selection of cluster head is optimized by some probability which does not respect to geographic location. As a result, there is a possibility that the cluster heads could be concentrated in one part of the network. Cristescu and Vetterli [30] proposed the heuristics for finding minimum power gathering tree. Krishnamachari [31] investigated the benefit of aggregation through three suboptimal schemes. Since an optimal tree is NP-complete or NP-hard, both paper attempted to find a suboptimal solution without considering detailed interaction between nodes (e.g., how the tree is bootstrapped, how the tree adapts to network dynamics) which is the main parts of our work.

VPCR (Virtual Polar Coordinate Routing) [16] is a point-to-point routing protocol based on hierarchical tree as our work. To create tree, each node needs to know its angle information which must be consistent with physical topology. It means that each node must know its location, otherwise two reference nodes are required to determine the location. Moreover, each node must keep two-hop neighbors in order to achieve the best performance. Adding and removing nodes tend to adversely affect the tree because VPCR need to preserve the alignment with physical topology. In the worst case, the subtree must be rebooted which costs both time and energy. Unlike VPCR, the tree in SDC does not depend on the location of nodes that makes the protocol more robust to dynamic networks.

Constrained random walk [32] considers routing in dynamic natures of WSNs as our work. Randomized algorithm is used to determine the next hop in order to achieve load balancing. However, they consider only one-source network

in their evaluation because multiple-source network requires much more complex computation to balance the energy. In contrast, SDC considers both multiple sources and multiple sinks and it has high scalability.

Karlof and Wagner articulate various attacks and countermeasures against sensor networks [34]. They present the detailed analysis of all the major routing protocols (TinyOS beaconing, directed diffusion, geographic routings, LEACH, rumor routing, etc.). We are currently including a security scheme in SDC. This is one of our future works.

The assumptions (Sect. 2.1) and simulation setups (Sect. 3.1) in the paper are based on a widely-used sensor node called Mica Mote [1]–[3]. It consists of a 4 MHz Atmel microprocessor with 128 kB of programmable memory and 4 kB of data memory. The network device is a RF Monolithic 916 MHz, amplitude shift keying (ASK) RF transceiver. The bandwidth of Mica Mote is 38.4 kbaud encoded with Manchester code. It also has a group ID as a standard feature which can be used straightforwardly in SDC. Mica Mote runs on TinyOS which provides a programming environment and a complete network stack on this platform. A link-level acknowledgment can be sent by the receiver for each packet successfully received.

## 5. Conclusions

This paper has demonstrated that SDC efficiently collects the data packets across multi-hop, wireless sensor networks while maintaining a constant amount of local state and making only local decisions. SDC also provides the solutions against dynamic natures of sensor networks. In other words, it is a self-organized protocol which can deal with the joining and leaving nodes. One great advantage of SDC which differs from conventional MANET routing protocols is that the sensors and multiple sinks seamlessly work together and each sensor can automatically discover a potential nearest sink. We have examined the performance of SDC in terms of packet delivery ratio, end-to-end delay, and path length. The results have shown that SDC achieves notably high delivery ratio, low delay, and comparable path length with the existing protocols including proactive and reactive protocols for MANET and a routing protocol for WSN. Besides, it is more robust to high offered loads than the existing solutions. One application in sensor networks that incurs high traffic load is structure health monitoring (SHM) [7], [8]. Nonetheless, the actual traffic rate in any applications is greater than the data generation rate of a node due to forwarded traffic, especially around the sink.

We plan to study SDC in many issues. We will study the improvement of performance when employing an aggregation scheme in SDC. The aggregation is an important technique in sensor networks because it greatly reduces the number of data packets. Another research direction is to include the reliability in our protocol. This will be end-to-end reliability employed in the transport layer. We also plan to implement SDC in Mica Mote to study its performance in real world environment.

## References

- [1] J. Hill and D. Culler, "Mica: A wireless platform for deeply embedded networks," *IEEE Micro.*, vol.22, no.6, pp.12–24, 2002.
- [2] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *Proc. 9th international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp.93–104, Nov. 2000.
- [3] "Mica2 mote." <http://www.xbow.com/index.htm>
- [4] D. Estrin, R. Govindan, J.S. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," *Proc. 5th International Conference on Mobile Computing and Networking (MOBICOM)*, pp.263–270, Aug. 1999.
- [5] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," *Proc. 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pp.88–97, Sept. 2002.
- [6] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat monitoring: Application driver for wireless communications technology," *SIGCOMM Comput. Commun. Rev.*, vol.31, no.2, supplement, pp.20–41, 2001.
- [7] K. Mechitov, W. Kim, G. Agha, and T. Nagayama, "High-frequency distributed sensing for structure monitoring," *Proc. First International Workshop on Networked Sensing Systems (INSS)*, pp.101–104, June 2004.
- [8] S. Kim, D. Culler, and J. Demmel, "Structural health monitoring using wireless sensor networks," 2003. <http://www.eecs.berkeley.edu/~binetude/course/cs294.1/paper.pdf>
- [9] A.S. Tanenbaum, *Computer Networks*, 4 ed., Prentice Hall PTR, Aug. 2002.
- [10] C.E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," *Proc. ACM SIGCOMM*, pp.234–244, Sept. 1994.
- [11] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," RFC 3626, IETF, Oct. 2003.
- [12] D.B. Johnson, D.A. Maltz, and J. Broch, "DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks," in *Ad Hoc Networking*, ed. C.E. Perkins, ch. 5, pp.139–172, Addison-Wesley, 2001.
- [13] C.E. Perkins, E.M. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," RFC 3561, IETF, July 2003.
- [14] B. Karp and H.T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," *Proc. 6th International Conference on Mobile Computing and Networking (MOBICOM)*, pp.243–254, Aug. 2000.
- [15] Y.B. Ko and N.H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," *Proc. International Conference on Mobile Computing and Networking (MOBICOM)*, pp.66–75, Oct. 1998.
- [16] J. Newsome and D. Song, "GEM: Graph embedding for routing and data-centric storage in sensor networks without geographic information," *Proc. 1st International Conference on Embedded Networked Sensor Systems (SenSys)*, pp.76–88, Nov. 2003.
- [17] J. Elson and K. Roemer, "Wireless sensor networks: A new regime for time synchronization," *SIGCOMM Comput. Commun. Rev.*, vol.33, no.1, pp.149–154, 2003.
- [18] H. Lundgren, E. Nordstro, and C. Tschudin, "Coping with communication gray zones in IEEE 802.11b based ad hoc networks," *Proc. International Workshop on Wireless Mobile Multimedia (WoW-MoM)*, pp.49–55, Sept. 2002.
- [19] "Network simulator - ns-2." <http://www.isi.edu/nsnam/ns/>
- [20] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," *Proc. International Conference on Mobile Computing and Networking (MOBICOM)*, pp.56–67, Aug. 2000.
- [21] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," *Proc. 4th International Conference on Mobile Computing and Networking (MOBICOM)*, pp.85–97, Oct. 1998.
- [22] S. Madden, R. Szewczyk, M.J. Franklin, and D. Culler, "Supporting aggregate queries over ad-hoc wireless sensor networks," *Proc. 4th IEEE Workshop on Mobile Computing and Systems Applications (WMCSA)*, pp.49–58, June 2002.
- [23] C.E. Perkins and E.M. Royer, "Ad-hoc on-demand distance vector routing," *Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pp.90–100, Feb. 1999.
- [24] M. Abolhasan, T. Wysocki, and E. Dutkiewicz, "A review of routing protocols for mobile ad hoc networks," *Ad Hoc Networks*, vol.2, no.1, pp.1–22, Jan. 2004.
- [25] P. Bahl and V.N. Padmanabhan, "RADAR: An in-building rf-based user location and tracking system," *Proc. IEEE INFOCOM*, pp.775–784, March 2000.
- [26] L. Doherty, K.S.J. Pister, and L.E. Ghaoui, "Convex optimization methods for sensor node position estimation," *Proc. IEEE INFOCOM*, pp.1655–1663, April 2001.
- [27] N.B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," *Proc. International Conference on Mobile Computing and Networking (MOBICOM)*, pp.32–43, Aug. 2000.
- [28] F. Ye, A. Chen, S. Lu, and L. Zhang, "A scalable solution to minimum cost forwarding in large sensor networks," *Proc. 10th International Conference on Computer Communications and Networks (ICCCN)*, pp.304–309, Oct. 2001.
- [29] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," *Proc. 1st International Conference on Embedded Networked Sensor Systems (SenSys)*, pp.14–27, Nov. 2003.
- [30] R. Cristescu and M. Vetterli, "Power efficient gathering of correlated data: Optimization, np-completeness and heuristics," *Poster of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc)*, June 2003.
- [31] B. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," *Proc. International Workshop on Distributed Event-Based Systems (DEBS)*, pp.575–578, July 2002.
- [32] S.D. Servetto and G. Barrenechea, "Constrained random walks on random graphs: Routing algorithms for large scale wireless sensor networks," *Proc. 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pp.12–21, Sept. 2002.
- [33] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," *Proc. 33rd Annual Hawaii International Conference on Systems Sciences*, pp.3005–3014, Jan. 2000.
- [34] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Proc. 1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA)*, pp.113–127, May 2003.



**Niwat Thepvilojanapong** received his B.E. degree in Electrical Engineering from Chulalongkorn University in 1999, and his M.E. degree in Information and Communication Engineering from the University of Tokyo in 2003. He is currently pursuing the Ph.D. degree in Information and Communication Engineering at the University of Tokyo. His research interests include wireless sensor networks, mobile ad hoc networks, and network measurement. He is a student member of the ACM and IEEE.



**Yoshito Tobe** received Ph.D. in Media and Governance from Keio University in 2000. He is currently a Professor in the Department of Information Systems and Multimedia Design at Tokyo Denki University. His research includes multimedia communications, Internet measurement, wireless sensor networks, and ubiquitous networked sensing. He is a member of ACM, IEEE Communications Society, SICE, and IPSJ.



**Kaoru Sezaki** received his B.E., M.E. and Dr.E. degrees in electrical engineering from the Univ. of Tokyo in 1984, 1986, and 1989, respectively. He joined the Institute of Industrial Science at the Univ. of Tokyo in 1989. He is now an associate professor at the Center for Spatial Information Science at the University of Tokyo. From 1994 to 2000, he was a guest associate professor at the National Center for Science Information Systems. He has been a guest associate professor at National Institute of Informatics since 2000. he has been a special member of Telecommunications Business Dispute Settlement Commission since 2001. From 1996 to 1997, he was a visiting scholar at the Univ. of California, San Diego. His research interests include communication networks, location- and context-aware network services, high-speed switching systems, collaboration systems with haptics, GIS and image processing. He is a member of the IEEE.