

Autonomous Configuration in Wireless Sensor Networks

Yoshito TOBE^{†(a)}, *Member*, Niwat THEPVILOJANAPONG^{††(b)}, *Nonmember*, and Kaoru SEZAKI^{†††(c)}, *Member*

SUMMARY Because of the large scale of wireless sensor networks, the configuration needs to be done autonomously. 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 solicit packet to discover the child nodes. The sensor receiving this packet decides on an appropriate parent to which it will attach, it then broadcasts the same packet to discover its child nodes. Through this process, the tree is created autonomously without any flooding of the routing packets. SDC avoids periodic updating of routing information but the tree need to be reconstructed upon node failures or adding of new nodes. The states required on each sensor are constant and independent of network size, therefore SDC scales better than the existing protocols. Moreover, each sensor can make forwarding decisions regardless of the knowledge on geographical information. We evaluated 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, shorter delay, as well as high scalability in various scenarios.

key words: *wireless sensor networks, configuration, autonomous, data collection, 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]. A great number of such sensors can coordinate amongst themselves to achieve a larger sensing task both in urban environments and in inhospitable terrain [2]. For instance, we can use wireless sensor networks for environmental and habitat monitoring, tracking system, failure detection, and intrusion detection [3], [4]. However, the proclaimed limitations of sensor networks which are resource constraints including memory storage, computational power, communication bandwidth, and energy resources have motivated the challenge to designing a routing protocol that fulfills the re-

quirements of sensor networks.

In general, a large number of sensors are deployed in 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 for whatever reasons, they must be able to reorganize themselves to repair failed routes. On the other hand, the user may deploy additional sensors to mitigate the 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.

Although many protocols and algorithms have been proposed for traditional wireless ad hoc networks, they are not well suited to the unique features and application requirements of sensor networks. To illustrate this point, the differences between sensor networks and ad hoc networks are outlined below [5]:

- The number of sensor nodes in a sensor network can be several orders of magnitude higher than the nodes in an ad hoc network.
- Sensor nodes are densely deployed.
- Sensor nodes are prone to failures.
- Sensor nodes mainly use a broadcast communication paradigm whereas most ad hoc networks are based on point-to-point communications.
- Sensor nodes are limited in power, computational capacities, and memory.
- Sensor nodes may not have global identification (ID) because of the large amount of overhead and large number of sensors.

The remainder of the paper is organized as follows. Section 2 gives a background on wireless sensor networks. Section 3 addresses the problem statement of our work. Section 4 enumerates our proposal, *Scalable Data Collection* (SDC), in detail. Section 5 evaluates the performance of SDC through a number of simulations. Section 6 discusses related work. Finally, we conclude by summarizing our findings and identify future research opportunities in Sect. 7.

Manuscript received July 16, 2005.

Final manuscript received July 16, 2005.

[†]The author is with the Department of Information Systems and Multimedia Design, Tokyo Denki University, Tokyo, 101-8457 Japan.

^{††}The author is with the Department of Information and Communication Engineering, the University of Tokyo, Tokyo, 113-8656 Japan.

^{†††}The author is with the Center for Spatial Information Science (CSIS), the University of Tokyo, Tokyo, 153-8505 Japan.

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

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

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

DOI: 10.1093/ietfec/e88–a.11.3063

*“Base station,” “sink node,” and “sink” are used interchangeably throughout the paper.

2. Wireless Sensor Networks

In this section, we describe sensing devices, MAC protocols, and followed by routing protocols used to construct wireless sensor networks. We then give the examples of sensor network applications.

2.1 Sensor Node Hardware

Sensor node is a basic unit in a sensor network, with on-board sensors, processor, memory, wireless modem, and power supply. There are many devices developed for networked sensing applications. Smart dust [6] is a system-on-chip (SoC) node. Designers of these platforms try to push the hardware limits by fundamentally rethinking the hardware architecture trade-offs for a sensor node at the chip design level. The goal is to find new ways of integrating CMOS, MEMS, and RF technologies to build extremely low power and small footprint sensor nodes that still provide certain sensing, computation, and communication capabilities.

However, the assumptions (Sect. 3.1) and simulation setups (Sect. 5.1) in the following sections are based on a widely-used sensor node called Mica Mote [1]. In contrast to SoC node, Mica Mote is a dedicated embedded sensor nodes. 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. 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.

2.2 Medium Access Control (MAC)

The MAC sublayer manages access to the physical network medium, and its fundamental goal is to reduce or avoid packet collisions in the medium.

2.2.1 The S-MAC Protocol

The main goal of the S-MAC protocol is to reduce energy waste caused by idle listening, collisions, overhearing, and control overhead [7]. The protocol includes four major components: periodic listen and sleep, collision avoidance, overhearing avoidance, and message passing.

Periodic listen and sleep is designed to reduce energy consumption during the long idle time when no sensing events happen, by turning off the radio periodically. Collision avoidance in S-MAC is similar to the distributed coordinated function (DCF) for IEEE 802.11 ad hoc mode, using an RTS/CTS exchange. The node knows how long to sleep, because a duration field in each packet indicates how long the remaining transmission will be. Thus overhearing avoidance is accomplished by putting nodes to sleep while

their neighbors are talking to each other. A long message in S-MAC is fragmented into packets and sent in a burst with one RTS/CTS exchange to reserve this medium for the entire message. This saves repeated RTS/CTS overhead and reduces overall message-level latency.

2.2.2 The T-MAC Protocol

The T-MAC protocol [8] attempted to improve upon the performance of the S-MAC protocol. It proposes using a dynamic duty cycle as against the fixed one in S-MAC to further reduce the idle listening periods. The node switches itself to sleep mode when no activation event has occurred for a predetermined time period. The activation event can be a reception of some data, expiration of some timer, sensing of the communication, knowledge of an impending data reception through neighbors' RTS/CTS and so on.

2.3 Routing Issues

Routing is a process of determining a network path from a source node to a destination. Data-centric approaches manage the name, route, or a piece of data via properties, such as physical location, that are external to a communication network. This is to be contrasted with address-centric approaches which use logical properties of nodes related to the network structure.

2.3.1 Low Energy Adaptive Clustering Hierarchy

Low Energy Adaptive Clustering Hierarchy (LEACH) [9] randomly rotates the cluster head function among network nodes to more evenly distribute the energy load. Every round, all network nodes select a random number between 0 and 1. If it is below a dynamic threshold, the node elects itself to be a cluster head. It then announces this election, with a CSMA MAC protocol, to all other network nodes, which then associate themselves with cluster heads based on received signal strength. The nodes then announce their association decisions to their cluster heads, also via a CSMA MAC protocol.

2.3.2 Directed Diffusion

Directed Diffusion [10] is a data-centric routing protocol based on the name of data. Sinks generate information request tasks, or interests, that diffuse through the sensor network. An essential component of directed diffusion is the use of gradients associated with each interest, used to direct and control information flow back to the sink. Gradients can be manipulated to reinforce good information delivery paths and disable unproductive ones.

2.3.3 Rumor Routing

In order to get sources and sinks to meet each other, rumor routing spreads information from each regions of the sensor

field, so that the two growing regions eventually intersect [11]. When the curve emanating from a source meets the curve emanating from a matching sink, a path has been established between the two. When there are multiple sources and sinks, it makes sense to merge information between different interest requests when they encounter each other, and also to merge information from different data propagating paths when they meet each other.

2.4 Time Synchronization

Since the nodes in a sensor network operate independently, their clocks may not be, or stay, synchronized with one another. This can cause difficulties when trying to integrate and interpret information sensed at different nodes.

2.4.1 Interval Method

In many situations, the temporal ordering of events matters much more than the exact times when events occurred. Interval methods [12] provide a lightweight protocol that can be used to move clock readings around the network and perform temporal comparisons. The key idea of the protocol is to focus on time intervals between events and transform such time differences from the time framework of one node to that of another by estimating communication delays and applying interval methods.

2.4.2 Reference Broadcasts

The key idea of the reference broadcast system (RBS) [13] is to use the broadcast nature of the wireless communication medium to reduce delays and delay uncertainty in the synchronization protocol. This is achieved by having the receiver nodes within the communication range of a broadcast message sent by a sender node synchronize with one another, rather than with the sender. This is accomplished by having the sender send a reference message to receivers who record its time of arrival each in their own time frame. The receivers then exchange this information among themselves.

2.5 Sensor Network Applications

Sensor networks can be used in various applications, e.g., environmental and habitat monitoring, tracking system, failure detection, intrusion detection [3], [4]. One example of ongoing work is habitat monitoring on the Great Duck Island [3]. 190 wireless sensors have been installed to monitor the habitat of the nesting petrels. The untethered, matchbox-sized sensors left in the burrows monitor the occupancy by recording temperature variations inside and wirelessly send the data to a gateway node on the island. In such applications, tasks of all sensor nodes are sensing specified values and reporting sensed data to the sink. Therefore, we need a protocol to do many-to-one communications originated from a number of sensor nodes, and destined at a sink node. Our target applications described above only require this kind of communication.

3. Problem Statement

We address a network mode we consider in this section, and followed by design goals of SDC.

3.1 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. 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 [14]–[19] are point-to-point routing protocols, i.e., $s, d \in \{N, BS\}$.

3.2 Objectives

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

- *Autonomous*. Because of a large number of sensor nodes, we need autonomous configuration to achieve sensing tasks efficiently. Each node in our protocol can discover path autonomously including joining and leaving of new and existing sensor nodes.
- *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 [20].
- *Robustness*. Our solution provides self-organized mechanisms in order to deal with the dynamic natures of sensor networks, i.e., joining and leaving scenarios.

4. Scalable Data Collection Protocol

SDC is based on a tree topology rooted at a sink node. Every sensor must be a member of the tree, i.e., an internal or leaf node, in order to communicate with the sink.

4.1 Constructing the Tree

A base station initiates the tree construction by broadcasting

a *child request* (CRQ) packet. A *Nonmember*, a node that is not attach to the tree, decides on its parent from the received CRQ packets by waiting for a short period (T_{col}) to collect a number of candidates and choosing a node whose defined metric is the best. The candidates are kept in a *parental candidate (PC) table*, which maintains the pairs of candidate IDs and metrics. A *Member* that is an internal, or leaf, node of the tree also updates its PC table according to the incoming CRQ packets. Assume T_{crq} and T_{join} denote the time that a node first received a CRQ packet and the time that a node joins the tree, respectively. The nonmember chooses a node whose T_{crq} is the minimum, and if many nodes have the same value of this metric, it will choose a node whose T_{join} is the minimum. Less time implies a node is nearer the base station, which results in the shorter path.

After choosing the parent, the nonmember sends a *child reply* (CRP) packet to the selected parent to inform that it is a child node. The child node then performs the same process as its parent, *i.e.*, broadcasts a CRQ packet to discover its own child nodes. Note that the T_{join} is the time when a node sends the CRP packet.

4.2 Adding the Nodes

A newly deployed sensor finds a parent by using a *joining mechanism* as follows. The joining node broadcasts a *parent request* (PRQ) packet, thereby making the neighboring nodes aware of its existence. All members of the tree that hear this packet reply by unicasting a CRQ packet to the joining node. Then, the process follows the tree construction phase, *i.e.*, the joining node sends a CRP packet to the selected parent. If the joining node does not receive any CRQ packet after broadcasting the PRQ packet, it waits for an incoming CRQ packets after one of its neighbors has become a member.

4.3 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 one of many possible reasons. For example, the battery of the node may be discharged with time, or the node may be damaged by the harsh environment or by the enemy. A detection of 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 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 parent from its PC table. This can be done by sending a CRP packet to a newly selected parent. If there is not any candidate in the PC table, it uses the joining mechanism as a newly deployed node. However, its child and grandchild nodes will not reply to this PRQ packet to prevent routing loop. Based on all of the simulations done in Sect. 5, two-hop information is sufficient for dealing with the routing loop problem. In the worst case where the or-

phaned node does not have any candidate in the PC table and there is no response to the PRQ packet, it sends a *parent query* (PQR) packet to its children 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 the child that has at least one candidate as its new parent by sending a *reverse* (REV) packet to inform it of the reverse relationship, *i.e.*, the selected child becomes a parent of the orphaned node, and switches to a new parent, chosen from its PC table. The relationships between other children of the orphaned node and the orphaned node do not change. If no child node has any candidate for a parent, the orphaned node still randomly

```

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

```

Fig. 1 Main algorithm.

```

1: void sendCRP() {
2: send CRP packet to parent
3:  $T_{join} \leftarrow \text{CURRENT\_TIME}$ 
4: if ( $\text{flag\_prt} = \text{DOWN}$ ) & ( $\text{flag\_crq\_sent} = \text{NO}$ ) then
5: broadcast CRQ packet
6:  $\text{flag\_crq\_sent} \leftarrow \text{YES}$ 
7: end if
8:  $\text{flag\_prt} \leftarrow \text{UP}$ 
9: }
10: void select_parent() {
11:  $\text{flag\_selprt\_call} \leftarrow \text{NO}$ 
12: if BS is in  $\text{pc\_table}$  then
13:  $\text{parent} \leftarrow \text{BS}$ 
14: else
15: for all members in  $\text{pc\_table}$  do
16:  $\text{parent} \leftarrow$  node whose  $T_{crq}$  is the minimum
17: if  $T_{crq}$  is equal then
18:  $\text{parent} \leftarrow$  node whose  $T_{join}$  is the minimum
19: end if
20: end for
21: end if
22: }
23: void link_failed() {
24: buffer packets in queue
25:  $\text{flag\_prt} \leftarrow \text{REPAIR}$ 
26:  $\text{pc\_table} \leftarrow \text{pc\_table} - \text{parent}$ 
27:  $\text{parent} \leftarrow \text{NULL}$ 
28: if  $\text{pc\_table} > 0$  then
29: if  $T_{crq}$  of at least one member in  $\text{pc\_table}$  is less than own  $T_{crq}$  then
30: select_parent()
31: else
32: broadcast PRQ packet
33:  $\text{pc\_table} \leftarrow \text{NULL}$ 
34: end if
35: else // if  $\text{pc\_table} = \text{NULL}$ 
36: broadcast PRQ packet
37: end if
38: }

```

Fig. 2 The functions called by the main algorithm.

chooses one child as a new parent by sending a REV packet and will let this selected child find a new parent using the joining procedure. Note that the last scenario is quite rare but may occur in very sparse networks.

4.4 Collecting Data and Discussions

When the network size becomes larger, it is impossible to use only one sink although we have an optimal protocol because the traffic will concentrate around the sink incurring a high loss rate. Thus, the user must deploy additional sinks at some ratio compared to the number of sensors in order to distribute the traffic 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 the nearest sink should arrive first. The nodes can use the group ID to distinguish different sinks. Thereby, they can be a member of multiple trees 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 the received data to its parent. The algorithms described thus far are summarized as pseudo-codes in Figs. 1 and 2, and a state transition diagram of a sensor is also illustrated in Fig. 3.

Because the number of states required on each node

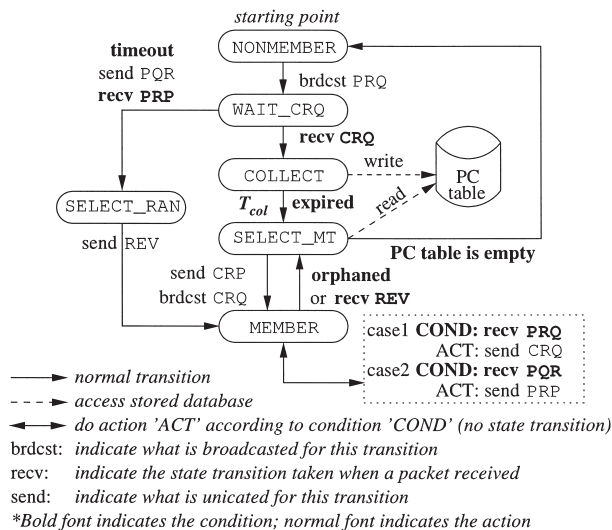


Fig. 3 State transition diagram.

is constant (a fix-sized PC table), and they are independent of node density and network size, SDC is highly scalable. Every node in SDC needs to broadcast only once to discover the route, thereby no propagation of routing packets. In other words, the routing packets are limited to one-hop neighboring nodes. Moreover, SDC does not apply periodic updating scheme that reduces traffic load so much. Furthermore, geographical information is also unnecessary.

5. Performance Evaluation

To evaluate the performance of SDC, we use the *ns-2* [21] simulation tool to run a number of simulations. We then compare the performance with the following protocols: (1) a proactive ad hoc routing protocol, i.e., OLSR [15], (2) two reactive ad hoc routing protocols, i.e., DSR [16] and AODV [17], and (3) a communication protocol for sensor networks, i.e., Directed Diffusion (DD) [10]. We also ran the simulations on DSDV routing protocol [14] which shows worse performance than AODV and DSR as also reported in [22]. Therefore, we do not include the results of DSDV in this paper.

5.1 Simulation Environment

The *ns-2* simulator includes full simulation of the IEEE 802.11 physical and MAC layers. Our simulations used this MAC layer and assumed symmetric links. Using this MAC layer did not affect the evaluation because we needed to evaluate the network layer of five protocols. We randomly placed 50 sensors in a 200 m by 200 m square region. Each node had fixed radio coverage of 50 meters. All nodes had fixed positions, with no movement for the entire simulation. We used a constant bit rate (CBR) as our traffic sources. These CBR sources sent 64-byte data packets at the rate of 0.25, 0.5, 1, and 2 packets per second, i.e., 128, 256, 512, and 1,024 bps, respectively, while the bandwidth of the sen-

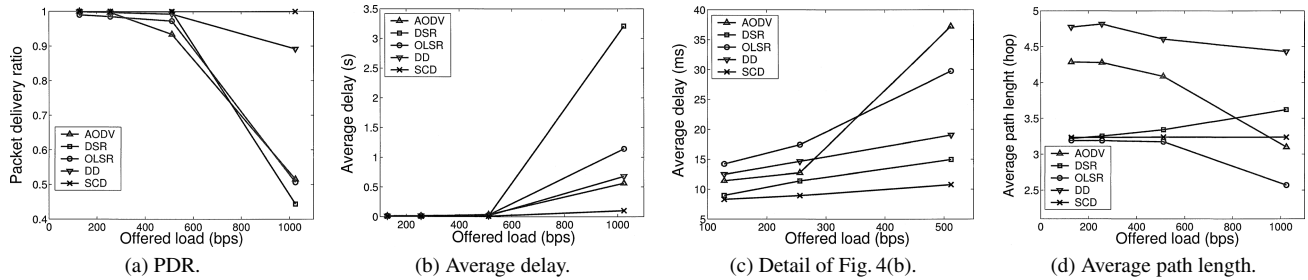


Fig. 4 Performance comparison in static networks.

sors was set to 19.2 kbps. The CBR agent was attached to a UDP agent, which in turn attached to the source node. For all of the simulations, the communication patterns were peer-to-peer and the starting time of each connection was randomly chosen. One node from each simulation was randomly chosen as a base station and the only destination for all traffic sources, while all the remaining nodes (49 nodes) were the source nodes (one flow per one source). Each simulation was run for 1,800 simulated seconds. Each result in the following graphs and tables represents an average of ten runs with identical traffic model, but different randomly generated topologies. The time waiting for collecting the candidates (T_{col}) was set to 0.1 second.

5.2 Performance Metrics

We studied our protocol through 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 because 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. Therefore, this metric characterizes both the completeness and correctness of the protocol.

Average delay is an important metric for comparing as it measures the quality of path determined by the routing algorithm. Protocols that send a 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 use network resources efficiently by selecting the shortest path from the source to the destination. Shorter path means fewer transmissions, which in turn implies lower dissipated energy.

5.3 Simulation Results

The comparisons of five protocols for three metrics are il-

lustrated in Fig. 4. SDC is the only protocol that can deliver nearly 100% of the originated packets for all offered loads (Fig. 4(a)). PDRs of DD and DSR are similar with those of SDC at light load but it can deliver only 89% and 44% at 1,024-bps load, respectively. PDRs of AODV and OLSR start to drop at 512-bps load and both can deliver only a half of originated packets at the extreme load. With high offered load, most of the data packets drop at the nodes around the sink due to high congestion. Because the sink in DD *periodically floods* interest messages to the entire network, the probability of channel contention increases, resulting in dropped packets when the buffer is overflow. Although OLSR uses multipoint relaying technique to suppress some redundant floodings, every node is still required to *periodically* exchange control messages which incurs a large volume of traffics. The congestions due to high offered load also force a node to find a new available route. When a node in DD detects failed or degraded paths, all nodes downstream of the lossy link (all nodes towards the sink) will apply the reinforcement rules, i.e., flooding the 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, OLSR, DSR, and AODV lead to a chaos of the network, which in turn incurs more congestions and more dropped packets. 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.

SDC has shorter delay than the other four protocols for all offered loads (Figs. 4(b) and 4(c)) because it encounters less contention as discussed above, and it also takes shorter paths (Fig. 4(d)). The path length of AODV and OLSR decreases as the load is increased because most of dropped packets are the packets destined for a distant destination. DD uses the longest path because it selects an empirically low delay path which may not be the shortest path. 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. However, this advantage of OLSR comes with a large amount of control traffics. We conclude that the tree created by SDC is nearly optimal in the viewpoint of path length, and it is quite efficient when considering PDR and delay.

5.4 Impact of Network Dynamics

To simulate the joining scenarios, 10 nodes were randomly deployed halfway through the simulations, with 50 nodes being deployed from the beginning of the simulation. For the leaving scenarios, we deployed 60 nodes at the beginning of the simulations and used an energy model that provided sufficient energy for 50 nodes, making the battery of 10 random nodes discharge at approximately the half-way point of the simulation. We chose heavy offered loads, i.e., 512- and 1,024-bps load, to make the situations fairly challenging for the protocols. These experiments were used to evaluate the robustness and resilience of the protocols against network dynamics. Note that 10 nodes means approximately 20% of the nodes in the network have a dynamic nature.

Table 1 presents the results of the joining scenarios. With 512-bps load, 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, SDC still achieved an excellent PDR, i.e., 98%, whereas PDRs of the other four protocols are lower than those of static scenarios because additional 10 nodes increase the total traffic loads, which incur more congestion in the networks. The average delays of all protocols also increase as a result of additional data traffics. The average path lengths slightly decrease as the node density increases. The exception is DD because it chooses a path according to the delay.

Table 2 shows the results of the leaving scenarios. SDC still had a superior PDR when some nodes left the networks, whereas PDRs of other protocols dropped compared to those of static scenarios. PDRs of DD decrease more than 20% because local repair was done by all nodes downstream of the left node. Instead of using all downstream nodes, only one upstream node, which is employed by the other four protocols, was better to initiate the local repair as shown by the simulations. Consequently, the delay of DD at 1,024-bps load is shorter than those of static case due to highly dropped packets destined to a distant destination. The path length

Table 1 Performance comparison in joining scenarios.

Load (bps)	PDR		Delay (ms)		Path length	
	512	1024	512	1024	512	1024
	AODV	0.8753	0.36	129	912	4.02
DSR	0.9988	0.35	17	3737	3.24	3.40
OLSR	0.9605	0.48	44	1277	3.03	2.46
DD	0.9649	0.82	87	1345	5.18	4.58
SDC	0.9980	0.98	13	516	3.33	3.30

Table 2 Performance comparison in leaving scenarios.

Load (bps)	PDR		Delay (ms)		Path length	
	512	1024	512	1024	512	1024
	AODV	0.9656	0.44	44	720	4.00
DSR	0.9435	0.33	279	4830	3.35	3.29
OLSR	0.9619	0.42	37	1261	3.10	2.37
DD	0.7794	0.65	44	394	4.19	3.82
SDC	0.9999	0.98	13	360	3.28	3.24

also affirms this fact, i.e., the packets in DD take shorter path compared to the results of static network. The average delays of all cases, except DD at 1,024-bps load, are higher than those of static scenarios because every protocol must take some 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. We conclude that SDC still achieves a good performance in dynamic sensor networks.

5.5 Impact of 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 3. We simulate only two moderate loads (128- and 256-bps loads) due to limited computational resource. As mentioned in Sect. 4.4, 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 5 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.

Table 3 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

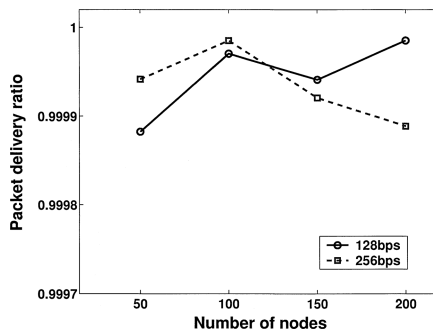


Fig. 5 The fraction of data packets successfully delivered as a function of number of nodes.

6. Related Work

A number of routing protocols [23] has been developed to provide efficient and robust communication in mobile ad hoc network (MANET) which is different from wireless sensor network (WSN) in many issues. However, many MANET routing protocols can work well in immobile networks [22]. To provide robustness against changing topology, the proactive routing protocols (e.g., DSDV [14], OLSR [15]) 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 [16], AODV [17]) 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 WSNs. LAR [19] and GPSR [18] are geographic routings that use location information to decrease overhead of route discovery and find routes quickly. Unfortunately, current methods of determining geographic location [24], [25] consume much energy and may not be possible in many sensor network scenarios.

The sinks in directed diffusion [10] draw interesting information by periodically flooding interests and setting up gradients within the network. These periodic interests waste the resources and disturb the delivery of data packets as done by periodic updating in other proactive protocols. Moreover, Directed diffusion requires location information which may not be possible as described above.

LEACH [9] 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.

7. 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) [4]. 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.

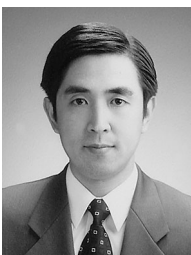
References

- [1] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *Proc. ASPLOS*, pp.93–104, Nov. 2000.
- [2] D. Estrin, R. Govindan, J.S. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," *Proc. MOBICOM*, pp.263–270, Aug. 1999.
- [3] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," *Proc. ACM WSNA*, pp.88–97, Sept. 2002.
- [4] K. Mechtov, W. Kim, G. Agha, and T. Nagayama, "High-frequency distributed sensing for structure monitoring," *Proc. INSS*, pp.101–104, June 2004.
- [5] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Comput. Netw.*, vol.38, no.4, pp.393–422, March 2002.
- [6] J.M. Kahn, R.H. Katz, and K.S.J. Pister, "Next century challenges: Mobile networking for "Smart Dust"," *Proc. ACM MOBICOM*, pp.271–278, Aug. 1999.
- [7] W. Ye, J.S. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," *Proc. IEEE INFOCOM*, pp.1567–1576, June 2002.
- [8] T. van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," *Proc. ACM SenSys*, pp.171–180, Nov. 2003.
- [9] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," *Proc. Hawaiian International Conference on Systems Sciences*, pp.3005–3014, Jan. 2000.
- [10] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," *Proc. ACM MOBICOM*, pp.56–67, Aug. 2000.
- [11] D. Braginsky and D. Estrin, "Rumor routing algorithm for sensor networks," *Proc. WSNA*, pp.22–31, Sept. 2002.
- [12] K. Romer, "Time synchronization in ad hoc networks," *Proc. ACM MobiHoc*, pp.173–182, Oct. 2001.

- [13] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," Proc. OSDI, Dec. 2002.
- [14] 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.
- [15] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," RFC 3626, IETF, Oct. 2003.
- [16] 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.
- [17] C.E. Perkins, E.M. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," RFC 3561, IETF, July 2003.
- [18] B. Karp and H.T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," Proc. ACM MOBICOM, pp.243–254, Aug. 2000.
- [19] Y.B. Ko and N.H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," Proc. ACM MOBICOM, pp.66–75, Oct. 1998.
- [20] A.S. Tanenbaum, Computer Networks, 4th ed., Prentice Hall PTR, Aug. 2002.
- [21] "Network simulator—ns-2." <http://www.isi.edu/nsnam/ns/>
- [22] 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. ACM MOBICOM, pp.85–97, Oct. 1998.
- [23] 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.
- [24] 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.
- [25] N.B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," Proc. ACM MOBICOM, pp.32–43, Aug. 2000.



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.



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.



Niwat Thepvilojanapong received his B.E. degree in Electrical Engineering from Chulalongkorn University (Thailand) 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, network measurement, and ubiquitous computing. He is a student member of

the IEEE, the IEEE Communications Society, and the ACM.