FlexiTP: A Flexible-Schedule-Based TDMA Protocol for Fault-Tolerant and Energy-Efficient Wireless Sensor Networks

Winnie Louis Lee, Amitava Datta, *Member*, *IEEE*, and Rachel Cardell-Oliver

Abstract—FlexiTP is a novel TDMA protocol that offers a synchronized and loose slot structure. Nodes in the network can build, modify, or extend their scheduled number of slots during execution, based on their local information. Nodes wake up for their scheduled slots; otherwise, they switch into power-saving sleep mode. This flexible schedule allows FlexiTP to be strongly fault tolerant and highly energy efficient. FlexiTP is scalable for a large number of nodes because its depth-first-search schedule minimizes buffering, and it allows communication slots to be reused by nodes outside each other's interference range. Hence, the overall scheme of FlexiTP provides end-to-end guarantees on data delivery (throughput, fair access, and robust self-healing) while also respecting the severe energy and memory constraints of wireless sensor networks. Simulations in ns-2 show that FlexiTP ensures energy efficiency and is robust to network dynamics (faults such as dropped packets and nodes joining or leaving the network) under various network configurations (network topology and network density), providing an efficient solution for data-gathering applications. Furthermore, under high contention, FlexiTP outperforms Z-MAC in terms of energy efficiency and network performance.

Index Terms—Wireless sensor networks, energy efficiency, memory efficiency, end-to-end data delivery guarantees.

1 Introduction

Wireless sensor networks are used to gather information in diverse settings including natural ecosystems, battlefields, and man-made environments [1], [2]. Networks in these environments need to be able to self-configure and, ideally, to self-calibrate, without knowing anything of the network topology or sensor requirements in advance. Sensor nodes are typically powered by batteries, and the cost of replacing these batteries is expensive. It is desirable to prolong the lifetime of a network by minimizing energy consumption in sensor network operations. Furthermore, the users of sensor networks need end-to-end guarantees on data delivery [3], [4]: predictable throughput for gathered data, fair access to the network for all data-gathering nodes, and robust self-healing of the network when nodes join or leave the network and when communication conditions change [5], [6], [7]. In practice, however, it has proved difficult to achieve these guarantees, because of the severe resource constraints (battery power and data memory) of sensor network nodes and the hostile environments in which they must operate [8]. In addition to minimizing network resources while maintaining end-to-end delivery guarantees, modularity in sensor network designs is important. Most sensor network protocols focus on performance while addressing issues such as scheduling, routing, data buffering, and power management at different levels [9], [10]. These observations lead to the design of a flexible-schedule-based TDMA Protocol (FlexiTP) that provides end-to-end guarantees on data delivery while also respecting the severe energy and memory constraints of wireless sensor networks.

FlexiTP is a self-healing protocol designed for periodic data-gathering applications. Data-gathering sensor nodes sense their environment periodically at a fixed rate and deliver data to a base station [11]. Sensor nodes serve as both gatherers and routers for the data. FlexiTP supports modularity through a distributed slot-structure algorithm that provides scheduling, routing, and time synchronization functions. FlexiTP is a TDMA-based protocol in which nodes only transmit and receive packets at their own time slot(s) and sleep until their slots turn up again. This approach conserves energy as nodes in sleep mode consume much less energy than those in idle mode [12]. FlexiTP nodes require minimal local buffering since the network schedule is chosen so that each node forwards messages immediately. FlexiTP is designed for a static ad hoc network, in which nodes may suffer temporary or permanent failure, and new nodes may be added to the network at any time, but nodes otherwise remain in (nearly) the same location. FlexiTP adjusts to different node topologies and densities to create an efficient self-healing data delivery tree. Furthermore, buffering and retransmission are used to recover from a small number of lost packets.

The central contribution of the FlexiTP protocol is its synchronized and loose slot structure. Nodes can claim or remove a slot based on the current information in their lookup table (that is, schedule) without exchanging information with any other nodes in the network prior to

• *The authors are with the School of Computer Science and Software Engineering, University of Western Australia, Crawley WA 6009, Australia. E-mail: {winnie, datta, rachel}@csse.uwa.edu.au.*

Manuscript received 13 Apr. 2007; revised 18 July 2007; accepted 16 Aug. 2007; published online 12 Sept. 2007.
Recommended for acceptance by M. Ould-Khaoua.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2007-04-0112.
Digital Object Identifier no. 10.1109/TPDS.2007.70774.

modifying their schedules. The node lookup table provides information for scheduling, routing, and time synchronization purposes [13]. In FlexiTP, it is not necessary to specify the number of slots required for the network in advance, and nodes can join or leave the network at any time. This flexible slot structure makes FlexiTP strongly fault tolerant and also highly energy efficient. In FlexiTP, initially, all nodes in a network build their data-gathering schedules. After this initial global one-off setup phase, all repair operations are local. As network density increases, the initial network setup cost and packet latency increase. Our simulations based on Mica2 Mote hardware show that FlexiTP is scalable for a large number of nodes (up to 400 nodes) because it sustains the low initial network setup cost and low packet latency, as well as maintains good throughput for different network densities.

FlexiTP achieves a balance among data delivery guarantees, energy efficiency, and memory efficiency through the following schemes:

- *FlexiTP maximizes the network lifetime* by putting nodes into sleep mode without disrupting the ongoing traffic and fairly distributing energy-intensive operations across the network.
- *FlexiTP is scalable for large numbers of network nodes* because 1) local repair is used to recover from node and communication failures, 2) nodes minimize buffered information, and 3) communication slots are reused by nodes outside each other's interference range.
- *FlexiTP provides data delivery guarantees* because 1) scheduled communication is used to minimize communication errors from radio interference and to guarantee a communication path for each node and 2) local repair is used to recover from node and communication faults.

The rest of the paper is organized as follows: We discuss some related work in Section 2. We describe our FlexiTP approach in Section 3. Section 4 describes our simulation methodology and presents our simulation results. We conclude with some discussion and present future work in Section 5.

2 Related Work

Miller and Vaidya [14] propose a MAC protocol that addresses end-to-end guarantees on data delivery in terms of memory and energy efficiency. The protocol uses a second low-power radio to allow senders to wake receivers if a specified number of packets are buffered, and so, a buffer overflow can be avoided. The protocol also determines an optimal period of the wake-up to minimize energy consumption. The protocol in [15] uses the base station to adjust the modulation level (bits per symbol) of each cluster head in a network based on the cluster head's feedback in order to make a good trade-off between energy consumption and transmission quality. The major drawbacks of this protocol are the queuing process at cluster heads and the centralized decision making. The number of packets arriving at cluster heads is not controlled, and so, packets will be dropped if the buffer overflows, resulting in increased latency and packet loss. The two protocols proposed in [14] and [15] that did address end-to-end guarantees on data delivery issues in wireless sensor networks were different from our approach as they used a two-radio architecture. Even though a direct comparison between the performance of FlexiTP and these protocols has been performed, the different objectives between FlexiTP and these protocols makes this direct comparison difficult.

Several researchers have already shown extensive solutions for achieving energy-efficient protocols based on CSMA [10], [16], [17], [18], [19], TDMA [20], [21], [22], [23], [24], [25], [26], or both [27], [28], [29]. Their focus is to minimize sources of energy waste: idle listening, overhearing, collisions, and protocol overhead. FlexiTP is distinguished from previous protocols by providing end-to-end guarantees on data delivery within the energy and memory constraints of wireless sensor networks.

S-MAC [17] and T-MAC [18] use the CSMA technique and an RTS/CTS handshake in an attempt to avoid collisions. However, the overhead of RTS/CTS is high for typically small-sized data packets in sensor networks [28]. In S-MAC, a node that has more data to send can monopolize the wireless radio channel. This is unfair for other nodes that have short packets to send but need to wait for the completion of the transmission of the long packet. In contrast, the FlexiTP TDMA scheme guarantees fair access since no node can monopolize the medium and nodes have their own slots for transmission.

EAD [19] provides energy efficiency by rotating nodes in the network to act as backbone sensors, based on their residual energy. These backbone sensors are responsible for in-network data processing and packet routing, whereas the rest of nodes in the network sleep to conserve energy. This scheme provides high energy savings for event-driven applications where nodes only report events upon their detection. In contrast to EAD, FlexiTP is designed for periodic sensing applications, where each node has data to send at regular intervals. EAD uses CSMA/CA for data transmission, whereas FlexiTP eliminates RTS/CTS handshakes during data gathering, and this results in a low packet overhead. Similar to EAD, FlexiTP uses a forwarding-to-parent routing scheme where a tree is constructed at the initial network setup, and so, every node in the network knows where to route its packet during sensor network execution. This simple routing scheme is suitable for static sensor networks since it sustains high throughput and good energy efficiency [19].

TDMA-based protocols guarantee collision-free communication by scheduling slots for each node. TDMA protocols reduce idle listening, and this results in significant energy savings. The main challenges of this scheme are determining the collision-free slots to be assigned to nodes in multiple-hop networks [30], the overheads to set up and distribute a schedule through the network [16], and accurate time synchronization so that the nodes' time slots do not overlap [31]. Examples of TDMA-based MAC protocols are TRAMA [24], L-MAC [20], SS-TDMA [21], the protocol by Arisha et al. [22], and PACT [23].

TRAMA [24] organizes time into frames and uses a distributed election scheme based on traffic information at each node to determine which node can transmit at a particular slot. TRAMA uses a distributed hash function to determine a collision-free slot assignment. TRAMA builds a schedule when a node has data to send. This random

scheduling scheme increases queuing delays. In contrast, FlexiTP reduces queuing delays as schedules are assigned to nodes at the time of the initial network setup and nodes maintain this schedule throughout their lifetime. In this way, FlexiTP also avoids the need for a centralized scheduler. In addition, FlexiTP has a simpler algorithm to establish a schedule compared to TRAMA.

L-MAC [20] divides time into slots with a fixed frame consisting of a control message and a data unit. The main drawback of the L-MAC scheme is that it increases the idle-listening overhead since nodes must always listen to the control sections of all slots in a frame to allow nodes to receive data and to allow new nodes to join the network anytime. In B-MAC [10], each node also must listen to all control messages of its neighboring nodes to determine whether it is the intended receiver of the packet before going back to sleep. FlexiTP provides superior energy efficiency to L-MAC because nodes are only active in their own time slots, and so, no energy is wasted in idle listening.

SS-TDMA [21] is designed to operate on a regular grid topology such as rectangular, hexagonal, and triangular grids. The TDMA algorithm proposed in SS-TDMA is self-stabilizing. It tolerates faults such as nodes that are improperly initialized, slots assigned to corrupted nodes, and nodes with clock drifts. The main drawback of SS-TDMA is its constraint on the location of the nodes, which is impractical for many sensor network applications. Unlike SS-TDMA, FlexiTP caters to any network topology.

In the protocol by Arisha et al. [22], each cluster head in the network sends traffic and battery-level information from the nodes in its cluster to the base station. Based on this information, the base station builds a schedule (transmission and sleep) for nodes within each cluster. This approach is not scalable as cluster heads need to communicate directly with the base station. Furthermore, the protocol uses a fixed-length TDMA frame and, so, the maximum number of nodes must be known before deployment. In contrast, the FlexiTP slot structure can be changed dynamically during the runtime of the application.

PACT [23] uses passive clustering to create a communication network of cluster heads and base stations. The role of cluster heads and base stations is rotated based on the nodes' energy level. To prolong the network lifetime, only a subset of nodes (cluster heads and base stations) are active. At the start of a TDMA frame, these active nodes get preference in claiming a sequence of communication slots. PACT rotates the role of cluster heads to amortize the overheads of the TDMA scheduler. However, this method requires the cluster heads to rebuild a schedule for all nodes within their clusters. Unlike PACT, FlexiTP nodes maintain a fixed schedule throughout their lifetime until the network topology changes.

Z-MAC [28] is a hybrid MAC protocol that starts off as CSMA and switches to TDMA if the network load increases. Nodes execute a distributed slot selection algorithm to get a TDMA slot. When a node has data to send, it listens to all slots to check if its neighbors need to send data. If the node is the slot owner, it randomly backs off for a period of time and proceeds with sending its data if the medium is clear. The owner of a slot has a higher priority than nonowners. The owner uses its slot only if it has data to send; otherwise, it allows other nodes to use its slot. When multiple nodes within a subnetwork attempt to send messages at the same time, collisions often occur due to hidden terminals. In such high-contention networks, Z-MAC uses explicit congestion notification (ECN) messages to limit the occurrence of hidden terminals. When a sender detects heavy traffic load, it broadcasts the ECN message to its direct neighbors. These direct neighbors further propagate the ECN message to their neighbors; hence, nodes in a two-hop neighborhood of the sender get notified of the ongoing traffic. These neighbors then can only attempt to transmit a message during the scheduled slot of itself and its direct neighbors. These neighbors resume their previous states when they receive no more ECN messages. Similar to Z-MAC, FlexiTP performs a time slot assignment once at the initial network setup. This high initial overhead can be amortized over the network lifetime and eventually balanced by improved throughput and energy efficiency [28]. We compare the performance of FlexiTP with Z-MAC in Section 4.3.

3 FLEXITP

3.1 Protocol Overview

The main functions of FlexiTP are to manage

- the establishment of routes (data-gathering tree construction),
- the establishment of node schedules (time slot assignment),
- time synchronization to minimize clock drifts, and
- local repair for a network in the event of faults (fault tolerance).

FlexiTP has two main phases: an initial network setup and data-gathering cycles. Fig. 1 shows the order of steps executed in a network. Users can implement any data-gathering tree construction algorithm; the time slot assignment is the important feature of FlexiTP. Some steps within phases are carried out in parallel by the nodes, which are the neighbor schedule exchange and the fault-tolerant schedule exchange (depicted by dashed lines).

Initially, FlexiTP builds a data-gathering tree and assigns the nodes' schedules. Nodes then maintain their schedules throughout their lifetime in the network. Thus, this initial network setup is a one-off phase. During the initial network setup, FlexiTP uses CSMA/CA for packet transmission, and so, the nodes' receivers are always on (that is, in listening mode) and also uses a token-passing scheme. Nodes execute specific procedures only if they hold a token, hence avoiding collisions. After the initial network setup finishes, nodes perform regular data-gathering tasks using their TDMA schedules. They also can modify their schedules when the network topology changes. FlexiTP uses a forwarding-to-parent routing scheme [19] over a data-gathering tree. Nodes route packets to their parents until they lose connection with the parents.

Fig. 2 shows a node's schedule structure. The schedule only represents a list of slots when a node should be active, and so, the slots are not contiguous in time (discrete). In FlexiTP, slot number 1 is dedicated to the *Fault-Tolerant Listening Slot (FTS)* that is simply a short CSMA period

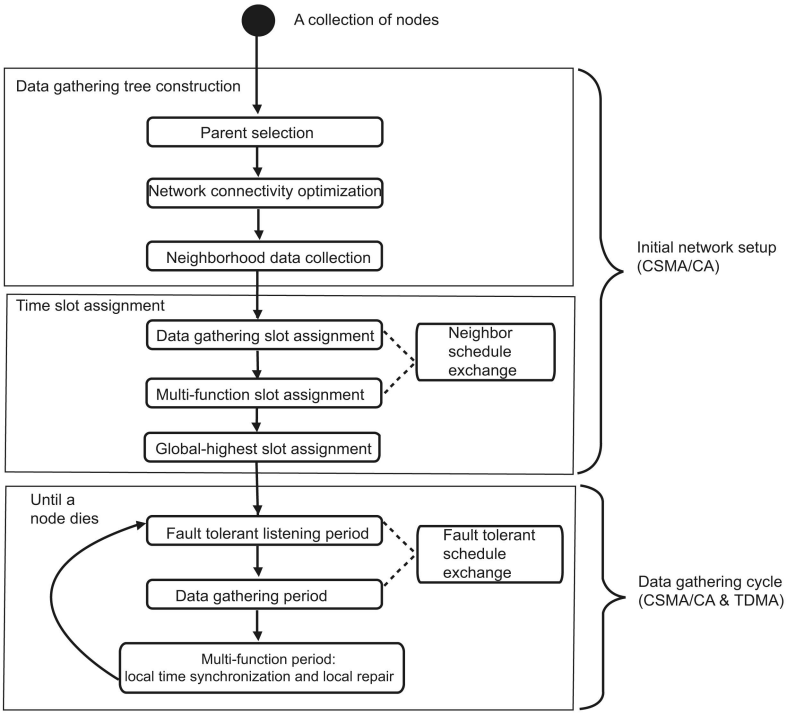

Fig. 1. FlexiTP phases.

where all nodes in the network are in listening mode. This feature allows nodes to adjust themselves according to their local neighborhood state. Nodes switch to the receive mode for their scheduled *Receive Slot List (RSL)* and to transmit mode for their scheduled *Transmit Slot List (TSL)*; otherwise, they switch to the sleep mode. A *Multifunction Slot (MFS)* is used for local time synchronization and is utilized for local repairs. The MFS in a node's TSL means that the node becomes the sender of a synchronization packet in that slot, whereas the receivers of the packet record the MFS in their RSL. The *Conflict Slot List (CSL)* records slots that are used by a node's first-level and second-level neighbors, which will be described later in Section 3.2. Nodes also maintain the *Global Highest Slot (GHS)* field, which contains the network's highest slot. Thus, nodes know the start and the end of a data-gathering cycle.

3.2 Terminology

This section describes terminologies used throughout the paper (illustrated in Fig. 3). A *first-level neighbor* of a sensor node is any node (not necessarily along the tree) within its communication range. Furthermore, any node that is within the communication range of the node's first-level neighbor(s) is referred to as a *second-level neighbor*. For example, B's first-level (direct) neighbors are A and D. Accordingly, B's second-level neighbors are BS, C, and E. *Parent* refers to an immediate node that forwards a source node's packet and that is closer to the base station (in terms of hop count) compared to the node. For example, A is the parent of B, and so, B is the *child* of A. *Ancestor* refers to a router that is more than one hop away from a source node to which the source node's packet is forwarded to and which is closer to the base station (in terms of hop count). For example, BS is the ancestor of B and D, that is, B and D are *descendants* of BS.

Fig. 2. A node's transmission schedule structure. The slots are not contiguous in time.

3.3 Data-Gathering Tree Construction

The FlexiTP data-gathering tree construction is responsible for establishing connectivity among nodes, and so, a tree path for routing data from a source node to the base station is created. After a data-gathering tree is constructed, the local topology is known to nodes in the network in that each node knows its parent, children, descendants, and first-level neighbors. A data-gathering tree is constructed in three phases: parent selection, network connectivity optimization, and neighborhood data collection. The nodes use CSMA/CA during these phases.

3.3.1 Parent Selection

The base station generates a connectivity token and initiates the data-gathering tree construction using a simple flooding

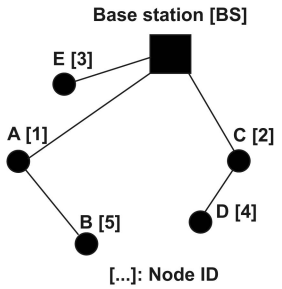

Fig. 3. A tree built for gathering data in the network of five nodes.

scheme. The connectivity token is distributed using a top-down subtree-by-subtree approach. A subtree consists of a parent and its children. The rule is that the connectivity token is passed from a node to the child with the smallest node ID in the current subtree. When a node obtains the connectivity token, it broadcasts a tree construction signal to find its prospective children. Nodes that are able to receive the signal become the children of the broadcaster node. FlexiTP allows the human manager to have control over the breadth of a data-gathering tree by specifying the maximum number of children a node can have. This phase ends when the token comes back to the base station finally, that is, after all the children of the base station and their descendants have selected their parents.

3.3.2 Network Connectivity Optimization

In the network connectivity optimization phase, a node that is unable to get a tree construction signal after a certain period of time (user parameter) will send a distress signal to find a parent. Recall that nodes use CSMA/CA during this phase, and so, a node backs off for a random period of time if it overhears another distress signal; if not, it broadcasts a distress signal. Neighboring nodes that receive this distress signal send a reply and confirm their availability to accept the distressed node as their child. The distressed node then selects the neighbor node with the lowest hop count to the base station to become its parent.

3.3.3 Neighborhood Data Collection

This phase collects information about a node's direct (first-level) neighbors. During the parent selection and network connectivity optimization phases, a node may send signaling packets and receive multiple replies from other nodes within its communication range. The node then records these nodes as its direct neighbors in its neighborhood table. This neighborhood data is then propagated to the base station, and hence, the base station can build a network connectivity map.

3.4 Time Slot Assignment

FlexiTP uses a depth-first-search schedule to reduce buffering. This scheduling scheme allows data sent by a source node to be forwarded by routers to the base station in an interleaving manner. This scheme enhances reliability by preventing the loss of packets due to buffer overflow. After a data-gathering tree is constructed, the local topology is known to nodes in the network such that each node knows its parent and children (if any). The base station then generates a time slot assignment token and initiates the time slot assignment phase. The token passing is done using the depth-first-search technique. The token is passed to the child that has the lowest node ID.

The FlexiTP time slot assignment function consists of four main phases: data-gathering slot assignment, MFS assignment, neighbor schedule exchange, and GHS assignment. In the *data-gathering slot assignment*, each node in the network builds its data-gathering schedule by selecting a slot to transmit its own data and then routers on the path to the base station assign *receive* and *forward* slots for that node's data. Once a node claims a slot, it executes the *neighbor schedule exchange* to propagate the claimed slot to its first-level and second-level neighbors. After all nodes in the network build their data-gathering schedules, they execute the *MFS assignment*. Finally, the base station informs the network's highest slot to all nodes in the network through the *GHS assignment*.

In FlexiTP, the slot selection always starts from slot number 2 because slot number 1 is dedicated to FTS. A node can claim a slot if the slot is not listed in its RSL, TSL, and CSL. Therefore, in selecting a slot, there is no need to know the true value of GHS. This slot selection scheme allows a slot to be reused by many nodes as long as the nodes' transmissions are not within interference range of each other, and hence, the spatial slot reuse can be maximized.

3.4.1 Data-Gathering Slot Assignment

When a node obtains the token, it allocates a transmit slot to send its own data by choosing the lowest slot number that is not listed in its RSL, TSL, and CSL fields. The node (current requestor) then informs the claimed slot to its parent (one of its first-level neighbors) by unicasting a *forward_slot_request*, as well as piggybacking an *inform_slot_request*. In this way, slots to forward the node's data to the base station are allocated. Furthermore, the *inform_slot_request* on the broadcast packet triggers the neighbor schedule exchange phase that will be described in Section 3.4.3. The *forward_slot_request* contains the claimed slot information: the slot number, the receiver's ID (parent ID), and the neighbor level counter. The requestor's first-level neighbors that receive the *forward_slot_request* checks the intended receiver of that packet. If a neighbor node is the intended receiver of the packet, it knows that it is the parent of the current requestor and so lists the received slot number in its RSL; else, it will execute the neighbor schedule exchange. The parent then assigns the corresponding transmit slot, that is, a slot to forward the requestor's data. The corresponding transmit slot must be bigger than the received slot number and is not listed in the parent's RSL, TSL, and CSL. Afterward, the parent executes the neighbor schedule exchange and sends the *forward_slot_request* to the requestor's ancestor (that is, routers). The data-gathering slot assignment as explained before is repeated until the base station receives the *forward_slot_request*. The data-gathering slot assignment ends when the token is returned to the base station.

3.4.2 Multifunction and Global Highest Slot Assignment

FlexiTP adopts a top-down MFS structure in which the base station initiates the MFS in a data-gathering cycle. In a multifunction period, each parent node in the network synchronizes its children. FlexiTP performs the MFS assignment right after the data-gathering slot assignment ends. Initially, the base station generates an MFS assignment token. The base station becomes the current parent to claim the MFS. It claims an MFS that is bigger than any slots in its RSL and TSL because the MFS is claimed after all nodes build their data-gathering schedule; furthermore, the MFS must not be listed in its CSL. The base station adds the claimed MFS in its TSL and then informs the MFS to all of its children by multicasting an *inform_slot_request* packet consisting of the slot number, the receivers' node ID

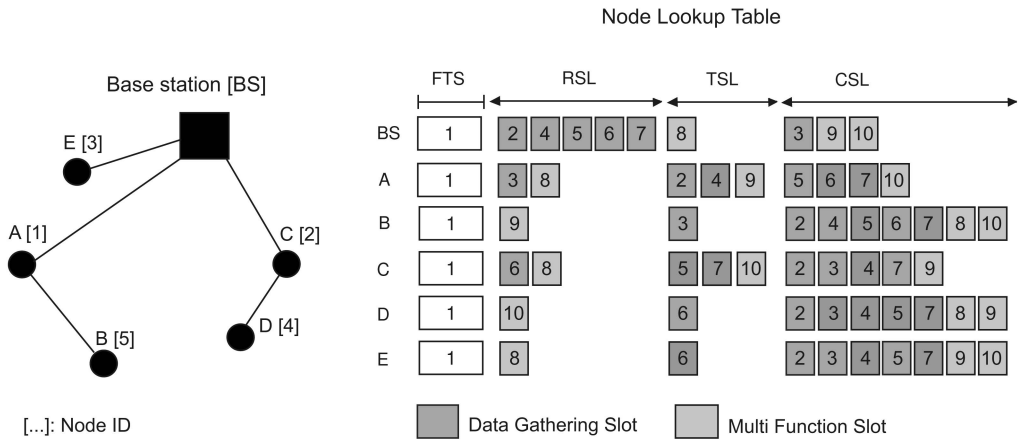

Fig. 4. FlexiTP nodes' schedule (lookup table).

(children node IDs), and the neighbor level counter. The children then assign the MFS as the reserved receive slot in their RSL. The neighbor schedule exchange is then also performed. Afterward, the base station performs the depth-first-search token passing. When a node in the network receives the token but has no child, the node skips claiming the MFS and continues passing the token accordingly. When the token is returned to the base station, the MFS assignment ends. After the data-gathering slot and MFS assignment phases, the base station knows the highest slot claimed by any node in the network. The base station then again initiates depth-first-search token-passing mechanism to inform the GHS to all nodes in the network.

3.4.3 Neighbor Schedule Exchange

The neighbor schedule exchange phase is performed through CSMA/CA communication in which a node broadcasts its claimed slot to its direct neighbors (first-level neighbors), and then, each of these direct neighbors propagate the slot information again to their direct neighbors (second-level neighbors). In this way, a slot claimed by a node is informed to all neighbors within twice the node's communication range. Since FlexiTP prevents nodes that are potentially within the interference range of each other to claim the same slot, collision-free traffic can be guaranteed.

If during the data-gathering slot assignment or the MFS assignment phase a node receives the *inform_slot_request* in a packet, it means that the neighbor schedule exchange phase needs to be executed. The neighbor slot exchange is controlled by the neighbor level counter information in the packet, which indicates the status of a neighbor, whether it is a first-level neighbor (the neighbor level counter is equal to one) or a second-level neighbor (the neighbor level counter is equal to two). If a node receives a packet with a level counter less than or equal to two, it will include the slot in its CSL. Furthermore, if the current level counter is less than two, the node increments the level counter value and broadcasts the packet to its direct neighbors; if not, it stops the neighbor schedule exchange.

3.4.4 Example

We now illustrate the time slot assignment algorithms described previously by using the data-gathering tree shown in Fig. 4. The base station generates a token and initiates the time slot assignment function. The token passing is done using the depth-first-search technique. The token is passed to the child that has the lowest node ID. Thus, the base station will pass the token to node A first, and A becomes the token holder. At this stage, no node has claimed slot number 2, so A claims slot number 2 and records this slot in its TSL. Node A then propagates the slot information to its parent (BS) and its first- and second-level neighbors. Node A's first-level neighbors are nodes BS, B, and E. Since BS is the intended receiver of A's transmission, BS lists the slot in its RSL, whereas B and E list the slot in their CSL. Again, BS, B, and E broadcast the slot information to their direct neighbors, and hence, C and D also list slot number 2 in their CSL. Afterward, A passes the token to its child, node B.

When node B becomes the token holder, it claims the lowest available slot that is not listed in its RSL, TSL, and CSL, which is slot number 3. Again, B propagates this slot information to its parent and neighbors. Node B's parent, A, then selects the next available slot to forward B's data to the base station, which is slot number 4. Node A then also propagates the claimed forward slot to its parent and neighbors. This token-passing mechanism is replicated until all nodes in the network receive the token. Note that node E can reuse slot number 6 because this slot does not appear in E's CSL. Thus, in slot number 6, both nodes D and E send their packets to their parents, whereas C and BS expect to receive a packet from their respective children in that slot. When the token is returned to the base station permanently, the data-gathering slot assignment phase finishes. Afterward, the base station initiates the MFS assignment by generating a token and claiming MFS number 8. The base station propagates this slot to its children and its first- and second-level neighbors. The base station adds the MFS number 8 in its TSL, whereas its children (nodes C and E) add the MFS in their RSL, and its neighbors (nodes B and D) add the MFS in their CSL. The rest of this phase is performed similarly to the depth-first-search token passing used in the data-gathering slot assignment. Since the base station knows the network

topology and the nodes' schedules, it then uses the token-passing mechanism again to inform nodes about the GHS in the network and the start of the data-gathering cycle. After FlexiTP's initial network setup finishes, nodes perform regular data-gathering tasks using their TDMA schedules.

3.5 Time Synchronization Scheme

Time synchronization is critical in TDMA-based MAC protocols because nodes that are involved in a scheduled communication must wake up at the same time to exchange information. In FlexiTP, a hierarchical structure is constructed in the data-gathering tree construction phase. Each node in the tree knows its parent and its children. FlexiTP performs time synchronization locally, with each parent synchronizing its children. This multihop parent-children broadcast synchronization approach is similar to root-neighbors synchronization approach in FTSP [32].

In the MFS of a node, the node broadcasts its clock and the current GHS number known to that node to the children. The purposes of propagating the GHS are to allow nodes in the network to keep the period of data-gathering cycle up to date and to allow nodes to perform a local repair, which will be described in Section 3.6. The FlexiTP time synchronization scheme is desirable because children only need to have the same clocks as their parent to ensure that a parent is in the receive mode when a child sends data to it and vice versa. This local synchronization scheme is simple but effective because clock drift is minimized by synchronizing nodes during each data-gathering cycle. Furthermore, it incurs low overheads since the synchronization message is piggybacked to the MFS packet. In addition, FlexiTP adjusts its slot length to include radio-switching times, which are the times required for a sensor node radio to transit from the sleep state to idle and from the idle state to sleep. Thus, when the sender wants to transmit, it is guaranteed that the intended receiver(s) is awake and listening. In the worst case, a node falls out of synchronization before its MFS cycle. The node then tries to resynchronize itself to the network by listening to its neighborhood activities. If the node overhears any of its neighbor's data transmission, it retrieves the current slot number of the transmission and uses the slot number to adjust its virtual clock. It then uses this virtual clock as the reference for reading its lookup table (that is, schedule).

3.6 Fault Tolerance

Wireless sensor networks are prone to network dynamics such as dropped packets, nodes dying, being disconnected, powering on or off, and new nodes joining the network, and so, the networks need to be able to self-configure without knowing anything of the network topology in advance [33], [34], [35], [36], [37]. In FlexiTP, nodes listen to environment activities during the FTS to allow these nodes to self-configure in the event of failures without prior knowledge of the network topology. If there is a node that sends a distress signal during FTS, nodes in the network that receive this signal will reply to it and perform a local self-repair. In the event that a new node is added to the network or when an existing node in the network wants to find a new parent, FlexiTP allows nodes in the network to adopt these faulty nodes as their children, assign schedules for these faulty nodes, and rebuild their own schedules. FlexiTP utilizes the existing TDMA schedule to perform a local repair, and so, a collision-free and cost-free schedule rebuilding and schedule exchange is guaranteed, which will be described later.

3.6.1 Packet Loss

FlexiTP handles packet loss during the initial network setup, as well as data gathering. In the initial network setup, FlexiTP utilizes an RTS/CTS/ACK scheme of IEEE 802.11 to handle packet loss during the depth-first-search token passing. For example, when a parent does not receive a token ACK from its child (receiver), the parent keeps retransmitting the token until it receives an ACK, for a specified period of time (user parameter). If after the time expires, the parent still receives no ACK, it proceeds with passing the token to the next child. On the other hand, if a child does not receive a token ACK from its parent (the parent probably dies) after a period of time, it will find a new parent. Since in the initial network setup phase, nodes work in CSMA/CA and they are all in listening mode, the child can broadcast a distress signal for finding a new parent. The child's neighbors that receive the signal send a reply. The child then selects the best parent based on the lowest hop count to the base station. The selected parent then updates its list of children and propagates this topology change to the base station, and so, the token-passing flow in the network reflects the current network connectivity.

In real-world implementations, collision-free traffic cannot be guaranteed. There would be environmental factors such as obstacles and humidity that could affect a transmission. Moreover, field trials of a sensor network for environmental monitoring of soil moisture in [8] showed that nodes may lose connectivity for extended periods. In FlexiTP, during data-gathering cycles, nodes detect packet loss based on their RSL, that is, their scheduled receive slots. For example, when a node does not receive a packet when it expects data from its child, it will request a data retransmission from that child during the MFS. The child appends the requested data onto the data packet that will be sent in the immediate data-gathering slot in the next data-gathering cycle. Furthermore, if a node receives the data retransmission request repeatedly, it decides to find a new parent in the FTS (see Section 3.6.3 for details) because the current link connectivity is too noisy.

3.6.2 New Nodes

In FTS, a new node sends a distress signal. Nodes that are within the communication range of the new node become neighbors and prospective parents of the new node. The new node then selects one of them as its parent, based on the lowest hop count to the base station. The selected parent allocates two slots to the new node: one data-gathering slot (that is, the parent's GHS incremented by one) and one MFS (the parent's MFS). If the parent does not have an MFS because it did not have a child before, it will assign an MFS to itself first, which is equal to the current GHS incremented by two, before allocating its MFS to the new node. Afterward, the new node's routers build the new node's data-gathering schedule by utilizing existing data-gathering slots and MFSs. The selected parent (let it be a requestor)

first proposes a forward slot (that is, forward slot request) for the new node to its parent. The requestor piggybacks the proposed slot on the data packet sent in the immediate data-gathering slot. The requestor's parent then checks the proposed slot against its lookup table. If the proposed slot exists in its lookup table, a next available slot will be chosen to replace it. The approved slot will be listed in the requestor parent's RSL and also will be sent to the requestor in MFS. The requestor then claims the approved slot and lists the slot in its TSL. Afterward, the requestor's parent becomes the current requestor and proposes a forward slot to its parent. This scheme is executed by traversing the path to the base station. Note that every time a node claims a new slot, it triggers the fault-tolerant schedule exchange (executed within a data-gathering cycle), which is described in Section 3.6.4.

3.6.3 Dead Nodes

The term "dead node" refers to one of the following conditions: a node whose battery is weak or dead or a node that is unreachable or unable to communicate due to environmental factors such as fog and obstacles [38]. If a node does not receive the expected data during all scheduled receive slots of a child or descendant after two data-gathering cycles, it assumes that the child or descendant is dead. It then removes the child or descendant from its children list or descendant list, respectively. The node then performs a slot garbage collection in which all receive slots, transmit slots, and MFSs that are associated with that child or descendant are deleted from its lookup table, and so, these slots are not wasted on idle listening. For example, when node D in Fig. 3 fails to send data to its parent, node B, B does not forward D's data to BS. If BS and B do not receive D's data after two data-gathering cycles, they can assume that D is dead.

A node becomes an orphan if it does not receive a synchronization message in its MFS from its parent after two data-gathering cycles. The orphan node then broadcasts a distress signal in the FTS to find a new parent. The orphan node first tries to find prospective parents that are neither its children nor descendants. The orphan node then selects a prospective parent that has the lowest hop count to the base station as its new parent. If other nodes in the data-gathering tree are outside of the orphan's communication range, a child or descendant of the orphan is then eligible to be a new parent if the child or descendant can reach a node that has a path to the base station. The data-gathering schedule assignment for the orphan node is the same as the new node's assignment, except that there may be many proposed slots as the orphan node may have children and descendants. The orphan node needs to propose its data-gathering slots (including the forward slots for its children and descendants) in its TSL to the new parent. This method allows the orphan node to transfer its TSL to the new parent while maintaining its RSL (that is, children and descendants of the orphan node do not need to rebuild their schedules).

3.6.4 Fault-Tolerant Schedule Exchange

The purpose of the fault-tolerant schedule exchange is to inform a node's claimed slot to other nodes that are potentially within the interference range of the node. The fault-tolerant schedule exchange phase uses the FTS schedule and the existing TDMA schedule: data-gathering slots and MFSs. Every time a node claims one or more transmit slots, the fault-tolerant neighbor schedule exchange phase is triggered. The list of claimed slots is called *inform_slots*, and the slot information (slot IDs and the neighbor level counter) is piggybacked on the node's data packet sent in the immediate data-gathering slot and MFS. A receiver of the packet retrieves the neighbor level counter. If the neighbor level counter is less than or equal to three, the receiver lists slots in its CSL, increments the neighbor level counter, and piggybacks the updated *inform_slots* on the data packet sent in the immediate data-gathering slot and MFS; else, the receiver drops the *inform_slots*. In FTS, the node also broadcasts its claimed slots to its first-level and second-level neighbors. This scheme ensures that first-level, second-level, and some third-level neighbors of a node are informed of a new claimed transmit slot.

4 Performance Evaluation

We implemented FlexiTP in the network simulator (ns-2) [39], and the code is downloadable at [40]. Our simulation results are based on the mean value of 20 different network topologies involving up to 400 nodes located randomly in a network area of 300 $\times$ 300 m. The location of the base station was fixed at the top-center of the network map. Our simulation parameters were based on Mica2 Mote hardware [14], [41], [42], [43], [44]. Table 1 presents our simulation parameters. We set the length of the FlexiTP data-gathering slot and MFS to be 26 ms: 23.3 ms for a packet transfer [43], 2.45 ms [14] for a node radio to transit from the sleep state to idle, and 0.25 ms [14] for a node radio to transit from the idle state to sleep. Although the switching energy cost is non-negligible [44], using the FlexiTP TDMA approach is still better than not putting nodes into the sleep state if they are inactive. For example, suppose a node is inactive for three slots. By putting the node into the sleep state during these three slots, the amount of energy spent is $(T_{\text{tran\_on}} * P_{\text{tran\_on}}) + (T_{\text{tran\_off}} * P_{\text{tran\_off}}) + (3 * T_{\text{slot}} * P_{\text{sleep}}) = 0.08$ mJ. This value is much smaller than keeping the node idle, which consumes $(3 * T_{\text{slot}} * P_{\text{idle}}) = 2.34$ mJ.

4.1 Energy Efficiency

4.1.1 Initial Network Setup

The initial network setup cost is the energy cost for constructing a data-gathering tree, collecting neighborhood information, and building nodes' schedules. Recall that this setup cost is a one-off cost as nodes maintain their schedules throughout their lifetime in the network. Fig. 5 shows the average node energy overhead in the initial network setup and how much of it is spent on the time slot assignment phase. The initial network setup cost per node is less than 0.6 percent of the node's total initial energy of 54,000 J, across various network scenarios. Thus, our simulations based on Mica2 Mote show that FlexiTP is scalable for up to 400 nodes since it maintains a low initial network setup cost. Fig. 6 shows the average time required for all nodes in a network to perform the initial network setup and how long of it nodes are in the time slot assignment phase. As the

TABLE 1
FlexiTP Simulation Parameters in ns-2

Simulation parameters	Default value
Channel bandwidth	19.2 Kbps
Packet size	56 bytes (36 bytes for payload and 20 bytes for header)
Transmission range	60 M
Transmit power	63 mW
Receive power	30 mW
Idle power	30 mW
Sleep power	0.003 mW
Transition power	30 mW
Transition time	2.45 ms
FlexiTP slot size	27 ms
FlexiTP FTS period	100 ms
Node initial energy	54,000 J

network density increases, the initial network setup cost increases. This is due to the increase in idle/transmit/receive activities in configuring a network.

4.1.2 Data-Gathering Cycle

We measured node energy consumption per cycle by calculating the rate of total energy consumed for listening, switching, transmitting, receiving, and sleeping during a data-gathering cycle, averaged over the entire network. When the slot reuse feature of FlexiTP is switched on, the amount of energy spent on radio switching is reduced and hence results in energy savings, as confirmed by our simulation results in Fig. 7.

An important goal in developing algorithms for wireless sensor networks is to minimize the energy consumption of sensor nodes' operations in order to increase the lifetime of the network. In FlexiTP, nodes switch to the low-power sleep mode if they are not scheduled to receive or transmit data and therefore conserve energy and increase the node lifetime. Fig. 8 shows the average network lifetime before the first node in the network dies.

4.2 End-to-End Guarantees on Data Delivery

If the FlexiTP slot reuse scheme feature is switched on, all nodes always claim the lowest available slot, and so, slot reuse can be maximized. This method will also minimize the number of slots required in a network. We measured the percentage of slot reuse, that is, the rate of the total number of slots that are used at least twice to the total number of transmission slots in a network, averaged over the entire network. Simulation results shows that FlexiTP provides a slot reuse of at least 62 percent of the total transmission slots used in the network across different network scenarios and network densities.

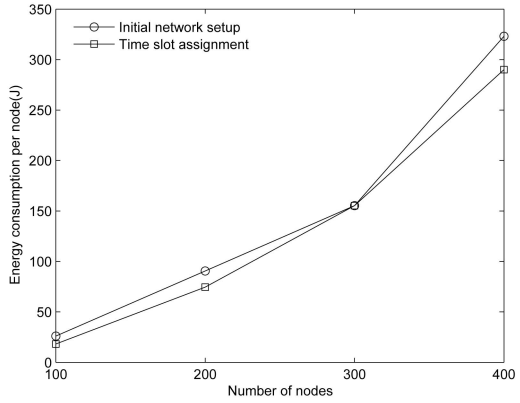

Fig. 5. Initial network setup cost per node versus network density.

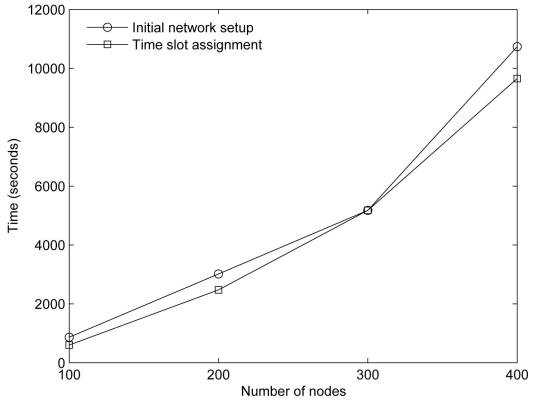

Fig. 6. Initial network setup duration versus network density.

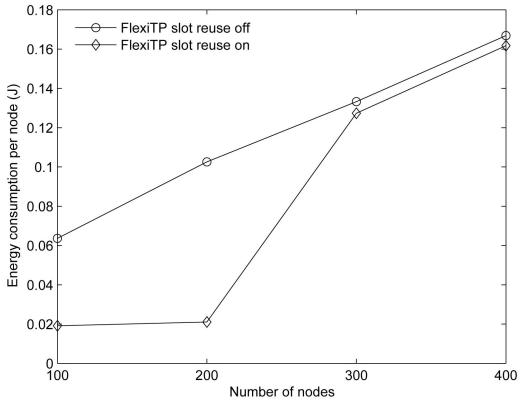

Fig. 7. Node energy consumption per data-gathering cycle versus network density.

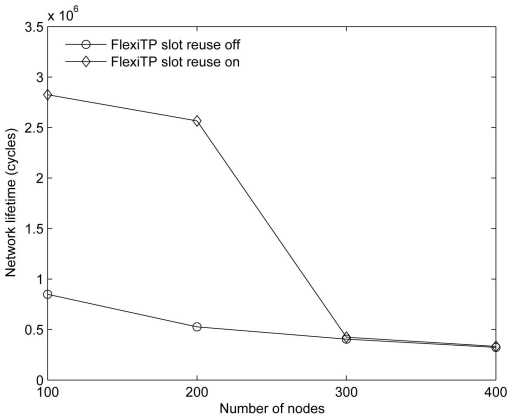

Fig. 8. Network lifetime in terms of data-gathering cycles versus network density.

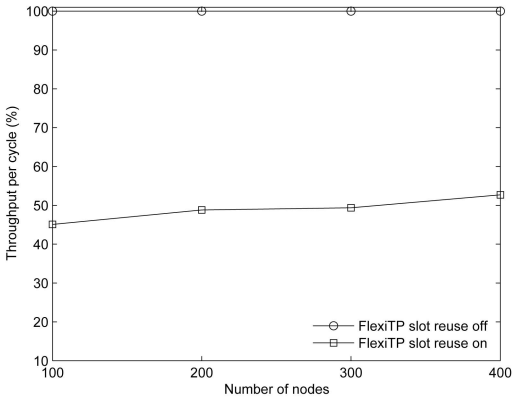

Fig. 10. Network throughput per data-gathering cycle versus network density.

We first measured the packet latency metric, that is, the time required for a packet to reach the base station since it was sent by a source node. Fig. 9 shows the average latency per packet in a data-gathering cycle. As expected, FlexiTP with slot reuse results in shorter latency than FlexiTP without the slot reuse scheme.

Sensor nodes have highly constrained computing power and memory. Multihop routing requires routers to buffer multiple packets prior to forwarding them to the next hop. This method can result in a low throughput as the number of packets dropped is high due to the nodes' memory buffer overflow. FlexiTP addresses this issue so that nodes only buffer one packet at a time to be sent to the next hop. This ensures regular data delivery and hence provides predictable throughput. Fig. 10 shows FlexiTP throughput with and without the slot reuse scheme. Theoretically, in multihop wireless sensors, a node's slot is safe to be reused by other nodes that are two or more hops away [45], [46]. However, our simulation results show that this assumption does not guarantee a collision-free slot reuse as the FlexiTP with slot reuse scheme on produced a lower throughput than that when the slot reuse scheme is off. We think that this phenomenon is caused by interference and irregularity in the radio channel.

4.3 Protocol Comparison

We compared the performance of FlexiTP with Z-MAC in terms of energy efficiency (energy consumption per node, energy consumption per packet, and network lifetime) and network performance (packet delay, throughput, fairness, and network utilization). We followed the Z-MAC ns-2 installation manual detailed in [47] and configured Z-MAC according to default settings shown in Table 1 in [28]. However, the simulation of Z-MAC for networks beyond 100 nodes did not work. Thus, we can only show the comparison of FlexiTP versus Z-MAC for network topologies of 100 nodes.

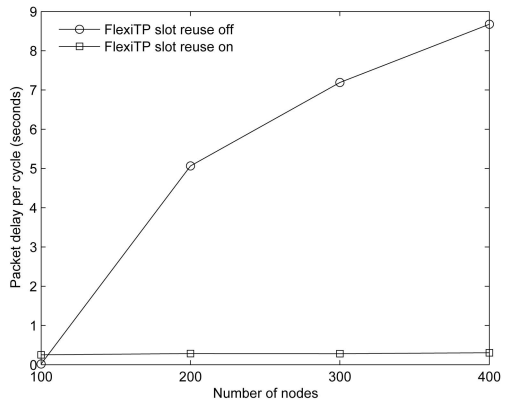

Fig. 9. Packet latency per gathering cycle versus network density.

We simulated a periodic data-gathering network for 300 seconds with interpacket arrival of 5 seconds. In all simulations, each sensor node in the network, except the base station, is a source node that always has data to send every 5 seconds (that is, fixed data rate) for 300 seconds. In this high-contention network case, FlexiTP outperforms Z-MAC in terms of energy efficiency and network performance, as confirmed by the simulation results shown in Table 2.

The packet delay metric measures the time between a packet generation and the packet reception at the base station, within the 300-second simulation. The energy consumption per node metric measures the total energy consumed for listening, transmitting, receiving, switching from sleep to idle mode, and vice versa, averaged over the entire network. FlexiTP outperforms Z-MAC in the average packet delay because each of a node's routers has already been preassigned a slot to relay the node's data to the base station, so that the time it takes for the data to reach from a source node to the base station is reduced and predictable. Furthermore, FlexiTP nodes send their data to their parent at the start of their slot, whereas Z-MAC nodes need to contend for the medium before sending their data. Z-MAC's small contention window in every slot results in a large slot size that increases data delay and energy consumption. In addition, Z-MAC nodes send ECN messages when they experience high contention, and this scheme increases the energy consumption per node for propagating ECN messages to a two-hop neighborhood and also extends the delay for moving toward TDMA operation, on already heavy-loaded networks.

We also calculated the energy consumption per packet, that is, the ratio of the total energy consumed by all nodes in the network to the total number of packets received at the sink, within the 300-second simulation. Table 2 shows that FlexiTP outperforms Z-MAC in the average energy

TABLE 2
FlexiTP versus Z-MAC

Metric	FlexiTP slot reuse off	FlexiTP slot reuse on	Z-MAC
Energy efficiency metrics			
Average energy consumption per node (J)	0.06	0.11	9.45
Average energy consumption per packet (J)	0.0000496	0.0000861	0.0205010
Average network lifetime (days)	3410	1663	20
Network performance metrics			
Average packet delay (seconds)	125.95	79.63	144.26
Average throughput (packets)	1109.85	1309.55	461.11
Average network utilization (%)	70%	82%	29%

consumption per packet, and this result is consistent with that of the average energy consumption per node. FlexiTP has a very low energy per packet because it guarantees the reliable delivery of packets and eliminates overheads (energy wastage) due to collisions, overhearing, idle listening, and overemitting. On the other hand, Z-MAC suffers contention-related collisions. Based on the average energy consumption per node and node initial energy of 54,000 J, Table 2 shows that FlexiTP prolongs the network lifetime by 83 times (with slot reuse) to 170 times (without slot reuse) compared to Z-MAC.

Table 2 also shows the throughput using FlexiTP versus that using Z-MAC, that is, the total number of packets received at the base station from all nodes in the network during the 300-second simulation. FlexiTP produces a higher throughput than Z-MAC because every node in the network is guaranteed a path to the base station. Since each FlexiTP node gets an equally sized time slot and all nodes in the network can access the channel, fairness is ensured. In high-contention networks, the ECN scheme of Z-MAC can promote fairness for nodes within one-hop neighborhood of each other and can limit hidden terminals in a local contention, however, at the cost of overloading the network with the ECN packets. This approach results in an added delay that leads to a decrease in throughput. Recall that FlexiTP with the slot reuse scheme on results in collisions that reduce the network throughput (as shown in Fig. 10). However, when the slot reuse scheme is on, a network's data-gathering cycle is shorter than that of a network without slot reuse. Table 2 shows that FlexiTP with the slot reuse scheme on results in a higher throughput than FlexiTP without slot reuse because it allows nodes to send more packets over short data-gathering cycles in 300 seconds.

Finally, we measured and compared the effective channel utilization of FlexiTP and Z-MAC. The channel utilization metric is the percentage of the total packets received at the base station (in bits) per time unit (in seconds) per channel bandwidth (in bytes), which can be formulated as follows (refer to Tables 1 and 2 for parameter values):

$$Utilization\_percentage = \frac{total\_packet\_received \times packet\_size \times 8}{simulation\_time \times (channel\_bandwidth \div 8)} \times 100. \quad (1)$$

Table 2 shows that FlexiTP achieves a better bandwidth utilization than Z-MAC under a large number of senders. This is because FlexiTP allows senders to transmit their packets without contending for the medium at all, whereas Z-MAC requires senders to compete for their own slots and their one-hop neighbors' slots.

Note that in [28], Rhee et al. show that Z-MAC can sustain a good performance under high contention. Their simulation results were based on low-density networks: a one-hop benchmark sensor network of 21 nodes and a two-hop benchmark sensor network of 18 nodes. In contrast, we simulated high-density networks; hence, there are a higher number of routers required for relaying a source node's data to the base station. This multihop benchmark intensely increases contentions in local neighborhoods and in the network overall. In high-density networks, Z-MAC's performance under high traffic loads degrades significantly.

4.4 Fault Tolerance

We tested FlexiTP's robustness by focusing on the most problematic scenarios such as adding or removing a substantial number of nodes to or from the network. The FTS is set to 500 ms. In the first simulation setup, we switched off a substantial number of nodes from the networks gradually over time, and so, there were an increasing number of orphan nodes. We evaluated FlexiTP's local repair performance in terms of the network reconnectivity ratio, energy expenditure, and local repair latency.

The reestablishment of network connectivity after node failures is depicted in Fig. 11. We found that the network connectivity reestablishment after FlexiTP local repair improves as the network density increases. Furthermore, the network gets partitioned as the number of node failures increases. The network connectivity represents the total number of nodes that are actually connected to the tree over the total number of nodes that are expected to be connected to the tree after a number of nodes in the network are switched off. This simulation results confirm that FlexiTP is adaptive to node failures as the remaining nodes still formed a connected network.

The energy expenditure for rebuilding the nodes' schedules is free since FlexiTP allows the nodes' routers to allocate slots to the nodes utilizing existing data-gathering and multifunction packets. We measured the average node energy expenditure to reestablish a network connectivity by computing the following energy costs: the

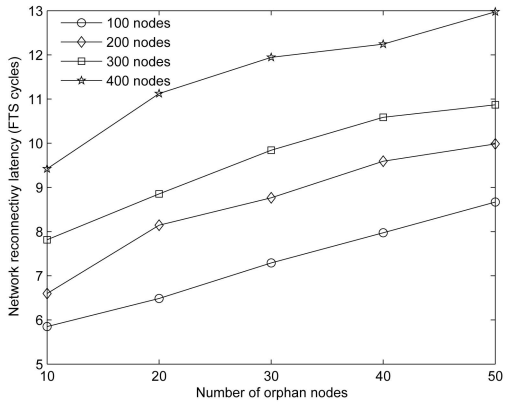

Fig. 11. Network connectivity reestablishment after a certain number of node failures.

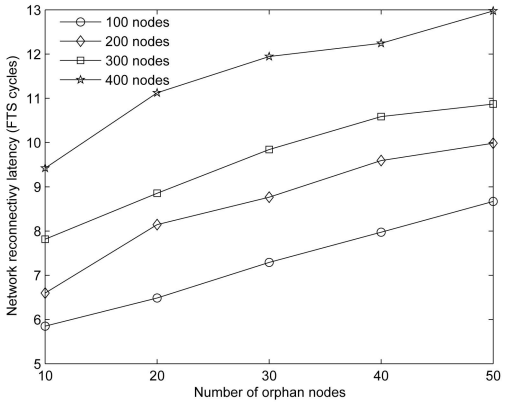

Fig. 13. Local repair latency in terms of the number of FTS cycles after a certain number of node failures.

cost for an orphan node to send a distress signal to find a parent, the cost for prospective parents to send a reply to the orphan node, the cost for the orphan node to select a parent by sending a confirmation signal, and the cost for the selected parent to send a schedule to the orphan node. Fig. 12 shows that the average energy expenditure of a node involved in the local repair is less than 1.2 J, that is, only around 0.000022 percent of the node's initial energy, across all scenarios. Fig. 13 shows the local repair latency in terms of the total number of FTS cycles required for orphan nodes to get a new parent and rebuild their schedules, that is, in the range of 6 to 13 cycles. The local repair latency can be improved by increasing the FTS period. The human manager can customize this feature according to the application's dynamics and needs.

In the second simulation setup, we added a number of new nodes to the networks gradually over time and evaluated how fast FlexiTP can incorporate the new nodes to the existing networks. Fig. 14 shows that as the number of new nodes introduced to the network increases, the average number of FTS cycles required for these nodes to select a parent increases. Based on the simulation results, it is apparent that FlexiTP is robust to node additions. However, FlexiTP's local repair performance starts to deteriorate when the number of nodes added is too large. This is because new nodes have to wait for a few more FTS cycles before they manage to select a parent. We also observed that the increase in the network density increases the latency of new nodes in selecting a parent. This is as expected because in high network densities, the networks are dense, and so, there are more nodes in the network that can reply to a new node's distress signal, possibly causing nodes to overhear each other's transmission. When that happens, other new nodes that lose in a contention wait for the next FTS and hence increases the latency for selecting a parent. The latency for selecting a parent can be improved by increasing the FTS period, and so, nodes in a network can take more new children during the FTS period in each data-gathering cycle, that is, more new nodes can join the network in each FTS.

5 Conclusions and Future Work

FlexiTP is a novel TDMA-based protocol for scheduling, routing, and the application layer (time synchronization) and provides end-to-end guarantees on data delivery such as predictable throughput for gathered data, fair access to the network for all sensor nodes, and robust self-healing while also respecting the severe operating constraints of wireless sensor networks. To the best of our knowledge, FlexiTP is the first integrated protocol for sensor networks that proposes a scheduling scheme that provides a balance among end-to-end guarantees on data delivery, energy efficiency, and memory efficiency.

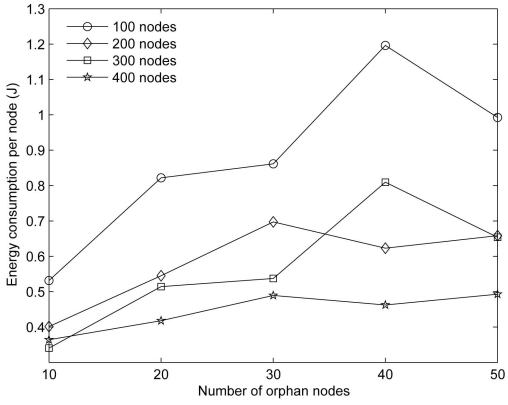

Fig. 12. Node energy expenditure for reestablishing a network connectivity after a certain number of node failures.

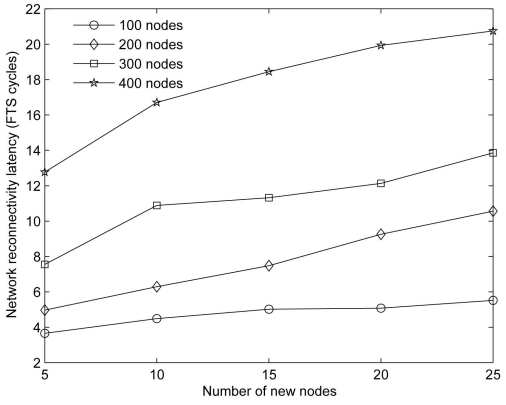

Fig. 14. Local repair latency in terms of the number of FTS cycles after a certain number of new nodes join the network.

FlexiTP offers a synchronized and flexible slot structure in which nodes in the network simply build, modify, or extend their schedules based on the local information available to them (RSL, TSL, and CSL). FlexiTP utilizes TDMA scheduling for efficient data gathering in wireless sensor networks. Nodes are only active during their scheduled slots; otherwise, they sleep. This fixed scheduling approach offers a high energy saving by reducing idle listening, avoiding collisions, and avoiding overhearing. FlexiTP uses a forwarding-to-parent routing scheme where nodes route packets to their parents until they lose connection with the parents. This routing scheme is suitable for periodic gathering sensor networks because nodes remain static throughout their lifetime.

FlexiTP implements local synchronization to minimize clock drifts among nodes: A parent synchronizes its children during a designated MFS. As the network density increases, the length of the data-gathering cycle increases. One can increase the frequency of time resynchronization within a data-gathering cycle by assigning additional MFSs in order to provide a tighter time synchronization. In FlexiTP, nodes can continue to function accurately in the event of failure of individual nodes (for example, erroneous network link, dying node, and topology changes) by performing a local repair in the FTS. The length of FTS can be adjusted according to the application requirements. For example, users can set a very short FTS (for example, 100 ms) for a sensor network application that is deployed in a stable environment or an application that can tolerate a slow network stabilization. Through comprehensive analysis and simulation results, we substantiate our argument that FlexiTP is fault tolerant and energy efficient across different network configurations. Our simulation results confirm that in high contention networks, FlexiTP performs better than Z-MAC in terms of energy savings and network performance.

The distributed and flexible slot-structure algorithm of FlexiTP enables nodes to modify their lookup tables during network execution. Lee et al. [48] leverage this feature to reconfigure nodes to report their data more rapidly or slowly, depending on the significance and importance of their data to the end user. In [48], FlexiTP is extended for event-sensing applications. Lee et al. [48] describe how FlexiTP is used as a baseline protocol for transferring time slots from one part of the network to another part and hence supporting nonuniform and reactive sensing in different parts of a network. Currently, we are working on using FlexiTP to support peer-to-peer communication in wireless sensor networks, whereby a node can communicate with any nodes in the network without going through the base station, that is, a node can become a sink for other nodes' data.

ACKNOWLEDGMENTS

The authors would like to thank the three anonymous reviewers for their valuable comments on the draft of this paper. They also would like to thank Ajit Chakrapani Warrier, who is one of Z-MAC primary developers, for his advice on the ns-2 implementation.

REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks,* vol. 38, no. 4, pp. 393-422, 2002.

[2] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A Wireless Sensor Network for Structural Monitoring," *Proc. Second ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys '04),* Nov. 2004.

[3] J. Kim, K. Park, J. Shin, and D. Park, "Look-Ahead Scheduling for Energy-Efficiency and Low-Latency in Wireless Sensor Networks," *Proc. Third ACM Int'l Workshop Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN '06),* Oct. 2006.

[4] F. Chen, F. Dressler, and A. Heindl, "End-to-End Performance Characteristics in Energy-Aware Wireless Sensor Networks," *Proc. Third ACM Int'l Workshop Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN '06),* Oct. 2006.

[5] R. Iyer and L. Kleinrock, "QoS Control for Sensor Networks," *Proc. IEEE Int'l Conf. Comm. (ICC '03),* May 2003.

[6] D. Chen and P.K. Varshney, "QoS Support in Wireless Sensor Networks: A Survey," *Proc. Int'l Conf. Wireless Networks (ICWN '04),* June 2004.

[7] L.H.A. Correia, D.F. Macedo, A.L. dos Santos, and J.M. Nogueira, "Issues on QoS Schemes in Wireless Sensor Networks," Technical Report RT.DCC.004/2005, DCC/UFMG, Apr. 2005.

[8] R. Cardell-Oliver, K. Smettem, M. Kranz, and K. Mayer, "A Reactive Soil Moisture Sensor Network: Design and Field Evaluation," *Int'l J. Distributed Sensor Networks,* vol. 1, no. 2, pp. 149-162, 2005.

[9] D. Culler, P. Dutta, C.T. Ee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, and J. Zhao, "Towards a Sensor Network Architecture: Lowering the Waistline," *Proc. 10th Workshop Hot Topics in Operating Systems (HotOS '05),* June 2005.

[10] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," *Proc. Second ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys '04),* Nov. 2004.

[11] M. Bhardwaj, T. Garnett, and A.P. Chandrakasan, "Upper Bounds on the Lifetime of Sensor Networks," *Proc. IEEE Int'l Conf. Comm. (ICC '01),* June 2001.

[12] G. Anastasi, M. Conti, M.D. Francesco, and A. Passarella, "An Adaptive and Low-Latency Power Management Protocol for Wireless Sensor Networks," *Proc. Fourth ACM Int'l Workshop Mobility Management and Wireless Access (MobiWac '06),* Oct. 2006.

[13] W.L. Lee, A. Datta, and R. Cardell-Oliver, "FlexiMAC: A Flexible TDMA-Based MAC Protocol for Fault-Tolerant and Energy-Efficient Wireless Sensor Networks," *Proc. 14th IEEE Int'l Conf. Networks (ICON '06),* Sept. 2006.

[14] M.J. Miller and N.H. Vaidya, "A MAC Protocol to Reduce Sensor Network Energy Consumption Using a Wakeup Radio," *IEEE Trans. Mobile Computing,* vol. 4, no. 3, pp. 228-242, 2005.

[15] Y. Yuan, Z. Yang, Z. He, and J. He, "An Integrated Energy Aware Wireless Transmission System for QoS Provisioning in Wireless Sensor Network," to be published in, *Elsevier Computer Comm.,* May 2005.

[16] A. El-Hoiydi and J.-D. Decotignie, "WiseMAC: An Ultra Low Power MAC Protocol for Multi-Hop Wireless Sensor Networks," *Proc. First Int'l Workshop Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS '04),* July 2004.

[17] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," *Proc. IEEE INFOCOM '02,* June 2002.

[18] T. van Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," *Proc. First ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys '03),* Mar. 2003.

[19] A. Boukerche, X. Cheng, and J. Linus, "A Performance Evaluation of a Novel Energy-Aware Data-Centric Routing Algorithm in Wireless Sensor Networks," *Wireless Networks,* vol. 11, no. 5, pp. 619-635, 2005.

[20] L.F.W. van Hoesel and P.J.M. Havinga, "A Lightweight Medium Access Protocol (LMAC) for Wireless Sensor Networks," *Proc. First Int'l Workshop Networked Sensing Systems (INSS '04),* June 2004.

[21] S.S. Kulkarni and M.U. Arumugam, "TDMA Service for Sensor Networks," *Proc. 24th Int'l Conf. Distributed Computing Systems Workshops (ICDCSW '04),* Mar. 2004.

[22] K. Arisha, M. Youssef, and M. Younis, "Energy-Aware TDMA-Based MAC for Sensor Networks," *Proc. IEEE Workshop Integrated Management of Power Aware Comm. Computing and Networking (IMPACCT '02),* May 2002.
[23] G. Pei and C. Chien, "Low Power TDMA in Large Wireless Sensor Networks," *Proc. IEEE Military Comm. Conf. (MILCOM '01),* Oct. 2001.
[24] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves, "Energy-Efficient Collision-Free Medium Access Control for Wireless Sensor Networks," *Proc. First ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys '03),* Mar. 2003.
[25] V. Rajendran, J.J. Garcia-Luna-Aceves, and K. Obraczka, "Energy-Efficient, Application-Aware Medium Access for Sensor Networks," *Proc. Second IEEE Int'l Conf. Mobile Ad Hoc and Sensor Systems (MASS '05),* Nov. 2005.
[26] S.C. Ergen and P. Varaiya, "PEDAMACS: Power Efficient and Delay Aware Medium Access Protocol for Sensor Networks," *IEEE Trans. Mobile Computing,* vol. 5, no. 7, pp. 920-930, July 2006.
[27] B. Hohlt, L. Doherty, and E. Brewer, "Flexible Power Scheduling for Sensor Networks," *Proc. Third Int'l Symp. Information Processing in Sensor Networks (IPSN '04),* Apr. 2004.
[28] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: A Hybrid MAC for Wireless Sensor Networks," *Proc. Third ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys '05),* Nov. 2005.
[29] N. Ramanathan, M. Yarvis, J. Chhabra, N. Kushalnagar, L. Krishnamurthy, and D. Estrin, "A Stream-Oriented Power Management Protocol for Low Duty Cycle Sensor Network Applications," *Proc. Second IEEE Workshop Embedded Networked Sensors (EmNetS '05),* May 2005.
[30] M.L. Sichitiu, "Cross-Layer Scheduling for Power Efficiency in Wireless Sensor Networks," *Proc. IEEE INFOCOM '04,* Mar. 2004.
[31] K. Langendoen and G. Halkes, "Energy-Efficient Medium Access Control," *Embedded Systems Handbook.* CRC Press, 2005.
[32] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "Robust Multi-Hop Time Synchronization in Sensor Networks," *Proc. Int'l Conf. Wireless Networks (ICWN '04),* June 2004.
[33] A. Gonzalez, I. Marshall, L. Sacks, I. Henning, and T. Khan, "A Self-Synchronised Scheme for Automated Communication in Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP '04),* Dec. 2004.
[34] G. Gupta and M. Younis, "Fault-Tolerant Clustering of Wireless Sensor Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC '03),* Mar. 2003.
[35] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentell, "Fault Tolerance Techniques for Wireless Ad Hoc Sensor Networks," *Proc. First IEEE Int'l Conf. Sensors,* June 2002.
[36] A. Boukerche, R.W.N. Pazzi, and R.B. Araujo, "Fault-Tolerant Wireless Sensor Network Routing Protocols for the Supervision of Context-Aware Physical Environments," *J. Parallel Distributed Computing,* vol. 66, no. 4, pp. 586-599, 2006.
[37] F. Araujo and L. Rodrigues, *On the Monitoring Period for Fault-tolerant Sensor Networks,* http://www.di.fc.ul.pt/~ler/reports/ladc05.pdf, 2007.
[38] A. Boukerche, I. Chatzigiannakis, and S. Nikoletseas, "Power-Efficient Data Propagation Protocols for Wireless Sensor Networks," *SCS J. Simulation: Trans. Soc. for Modeling and Simulation Int'l,* 2005.
[39] *The Network Simulator—ns-2,* http://www.isi.edu/nsnam/ns, 2007.
[40] W.L. Lee, *FlexiTP Implementation in ns-2,* http://www.csse.uwa.edu.au/~winnie/programs/flexitp, 2007.
[41] I. Raicu, L. Schwiebert, S. Fowler, and S.K.S. Gupta, "Local Load Balancing for Globally Efficient Routing in Wireless Sensor Networks," *Int'l J. Distributed Sensor Network,* 2005.
[42] *MICA2 Mote Datasheet,* http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf, 2007.
[43] G. Anastasi, A. Falchi, A. Passarella, M. Conti, and E. Gregori, "Performance Measurements of Motes Sensor Networks," *Proc. Seventh ACM Int'l Symp. Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '04),* Oct. 2004.
[44] G. Xing, C. Lu, Y. Zhang, Q. Huang, and R. Pless, "Minimum Power Configuration in Wireless Sensor Networks," *Proc. ACM MobiHoc '05,* May 2005.
[45] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: Distributed Randomized TDMA Scheduling for Wireless Ad-Hoc Networks," *Proc. ACM MobiHoc '06,* May 2006.
[46] B. Deb, S. Bhatnagar, and B. Nath, "A Topology Discovery Algorithm for Sensor Networks with Applications to Network Management," Technical Report DCS-TR-441, Rutgers Univ., May 2001.
[47] I. Rhee, *Z-MAC: Hybrid MAC for Wireless Sensor Networks,* http://www.csc.ncsu.edu/faculty/rhee/export/zmac/software/zmac/zmac.htm, 2007.
[48] W.L. Lee, A. Datta, and R. Cardell-Oliver, "A Novel Systematic Resource Transfer Method for Wireless Sensor Networks," *Proc. IEEE Global Telecomm. Conf. (GLOBECOM '06),* Nov. 2006.

Winnie Louis Lee received BCM degree (with honors) in mathematics and computer science and the PhD degree in computer science from the school of computer science and software engineering, from the University of Western Australia, Perth, Australia, in 2003 and 2008, respectively. Her research interests are in wireless sensor networks, MAC and cross-layer protocols, peer-to-peer networking in wireless sensor networks, and sensor network management strategies.

Amitava Datta received the MTech and PhD degrees in computer science from the Indian Institute of Technology, Madras, in 1988 and 1992, respectively. He did his postdoctoral research at the Max Planck Institute for Computer Science, the University of Freiburg, and the University of Hagen, all in Germany. He joined the University of New England, Australia, in 1995 and, subsequently, the School of Computer Science and Software Engineering, University of Western Australia, Perth, Australia, in 1998, where he is currently an associate professor. He was a visiting professor in the Computer Science Institute, University of Freiburg, in 2001, 2003, and 2005. His research interests are in parallel processing, optical computing, computer graphics, information visualization, bioinformatics, and mobile and wireless computing. He has served as a program committee member for several international conferences in these areas, including the International Parallel and Distributed Processing Symposium in 2001, 2005, and 2008. He is on the editorial board of the *Journal of Universal Computer Science* published by Springer and the *Journal of Pervasive Computing and Communications* published by Troubador. He is a member of the IEEE, the IEEE Computer Society, the IEEE Communications Society, the ACM, and the Mathematical Association of America.

Rachel Cardell-Oliver received the MSc degree in distributed systems software from the University of Western Australia and the PhD degree in protocol verification from the Computer Laboratory, University of Cambridge. She is an associate professor in the School of Computer Science and Software Engineering, University of Western Australia, Perth, Australia. Her research interests include building wireless sensor networks for environmental monitoring, programming languages and requirements logic for sensor networks, formal methods for distributed systems, and test generation from formal specifications.