

A Novel Systematic Resource Transfer Method for Wireless Sensor Networks

Winnie Louis Lee, Amitava Datta, and Rachel Cardell-Oliver
School of Computer Science and Software Engineering
The University of Western Australia
Perth WA 6009 Australia
Email: {winnie, datta, rachel}@csse.uwa.edu.au

Abstract—The use of wireless sensor networks for gathering environmental and safety-critical data in real time is increasing at a rapid rate. Some of the main criteria in designing sensor network architectures are energy-efficiency, self-management and self-healing. However, most protocols for data gathering and routing in sensor networks implicitly assume a regular rate of data gathering by individual nodes. While this is sufficient for sensing parameters that change slowly over time, individual nodes in a small part of a network may need to increase the rate of data gathering considerably for reporting important data in real-time. Though CSMA protocols can report increasing amount of data in theory, increased contention for the wireless medium in a small part of the network usually results in increased message collision and retransmission. In this paper, we present a novel TDMA protocol for transferring time slots from one part of the network to another part to support non-uniform and reactive sensing in different parts of a network. We discuss the design of this protocol and show that it performs much better compared to CSMA protocols both in terms of energy-efficiency and supporting non-uniform sensing.

I. INTRODUCTION

Wireless sensor networks (WSNs) consist of a large number of sensor nodes equipped with one or more sensing units that are used to gather information in diverse settings including natural ecosystems, battlefields, and man made environments [1]. These sensor nodes work under severe resource constraints such as limited battery power, computing power, memory, wireless bandwidth, and communication capability. Therefore, every activity in wireless sensor networks must be run efficiently without consuming too many resources. Typically, sensor nodes are deployed in remote and harsh conditions and so they need to be able to self-configure in the event of failures without prior knowledge of the network topology, and ideally, to self-manage without human interventions.

Most protocols designed for data gathering and routing in WSNs implicitly assume a regular rate of data gathering by individual sensor nodes. While this is sufficient for sensing parameters that change slowly over time, individual nodes in a small part of a network may need to increase the rate of data gathering considerably for reporting important data in real-time. For example, in structural monitoring applications, sensor nodes are deployed to monitor vibration (e.g., wind and earthquakes) that could damage the structure of a building [2]. It is critical for sensor nodes to send their data more often to the base station when they detect event triggers such as sensor

readings changing rapidly or exceeding user-specified thresholds. Though CSMA-based protocols can report increasing amount of data in theory, increased contention for the wireless medium in a small part of the network leads to increased message collisions and retransmissions.

Recently, researches are employing TDMA-based protocols to achieve collision-free communication [3]. TDMA-based protocols separate nodes in time, which means that nodes are only active (transmit or receive) during their scheduled slots. The main challenge of this scheme is to allow nodes to adjust their schedule according to their current sensing requirement. TRAMA [4] is an example of a TDMA-based protocol that allows nodes to adjust their schedule according to traffic conditions through regular neighborhood information exchange. If a node has few packets to send, it may release its slot for the remainder of the frame to other nodes that have many packets to send. This scheme results in a high bandwidth utilization, however TRAMA does so at the expense of high latency and high algorithmic complexity [3]. Furthermore, Miller and Vaidya [5] propose a two-radio based MAC protocol to allow senders to wake receivers using a second low power radio if a specified number of packets are buffered. Neither of these protocols support rescheduling based on nodes' current sensing requirements.

In this paper, we propose a *systematic resource transfer* (SRT) method that allows time slots (resources) from one part of the network to be transferred to another part. This allows non-uniform and reactive sensing in different parts of a network. We use a novel TDMA-protocol called Flexible-schedule-based MAC (FlexiMAC) [6], to support resource transfer among nodes in the network. FlexiMAC schedules node communication and continuously and efficiently collects and disseminates data, to and from sensor nodes in a data gathering tree. The problem of systematic resource transfer is interesting by itself and is an important property for most WSN applications. To the best of our knowledge, this problem has never been investigated before.

There are three main contributions of our SRT method. First, it can reconfigure the traffic patterns of sub-networks in the network based on the analysis of sensor data by enabling a node to get extra slots from other nodes in the network. The requesting node can specify the number of extra slots required in a data gathering cycle and specify or change how

long these extra slots will be required for. Second, when the SRT method is used with FlexiMAC, FlexiMAC scheduling and routing scheme ensures slots are systematically transferred among nodes with a low energy overhead and fast convergence in which traffic patterns in the network can be reconfigured and stabilized in one data gathering cycle. Third, the SRT method can be performed locally or centrally. In the local (decentralized) scheme, sensor nodes can autonomously perform the local SRT function based on their local neighborhood state. Whilst, the central scheme enables the central manager (base station) to actively perform the central SRT function based on global states of the network. The main advantage of the centralized scheme is that the central manager can distribute loads among sensor nodes to prolong the lifetime of the network by determining which sub-networks need more slots than other sub-networks. For example, if data reported by nodes in certain parts of the network are stable over a period of time, the central manager can reconfigure the traffic pattern of the network to conserve node energy. Depending on the application management policy, the central manager may decide to shut down that sub-network completely, i.e., nodes in that sub-network are relieved of sensing task and do not send their data to the base station for a period of time. Alternatively, the central manager may allow that sub-network to sense and send data to the base station less frequently, or transfer slots from that sub-network to other parts of the network. Thus, central SRT provides the potential for better monitoring, control, and management of sensor networks.

In the remainder of this paper, we briefly describe the FlexiMAC protocol in Section II. We then describe the local and central SRT function in Section III and IV respectively. In Section V, we show that the SRT method performs much better than CSMA protocols both in terms of energy-efficiency and supporting non-uniform sensing. Finally, conclusions are drawn in Section VI.

II. FLEXIMAC

The SRT method can be added to any TDMA-based protocol that allows nodes to build, modify, or extend their scheduled number of slots during execution. In this paper, we use the FlexiMAC protocol as a base for SRT. The main feature of the FlexiMAC protocol is its synchronized and loose slot structure. Nodes can claim or remove a slot based on the current information in their lookup table (see Figure 1) without exchanging information with any other nodes in the network prior to modifying their schedules. 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 FlexiMAC, slot number 1 is dedicated to *Fault Tolerant-Listening Slot* (FTS) that is simply a short CSMA period where all nodes in the network are in listen 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), transmit mode for scheduled *Transmit Slot List* (TSL), or else they switch to the sleep mode. A *Multi-Function Slot* (MFS) is used for

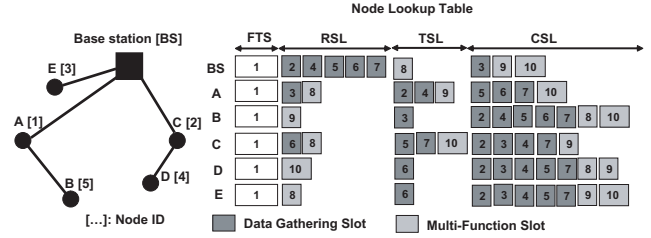


Fig. 1. Nodes' schedule in the data gathering tree. RSL of a node overlaps with TSL of the node's children.

local time synchronization and is utilized for resource transfer. In MFS, each parent node in the network synchronizes its children. This regular time synchronization scheme ensures that nodes' TSL match with their parent's RSL. Thus, when the sender wants to transmit it is guaranteed that the intended receiver(s) are awake and listening.

FlexiMAC uses a depth-first-search token passing mechanism for nodes to build their data gathering schedule. When a node holds the token, it can claim a slot if the slot is not listed in its RSL, TSL, and *Conflict Slot List* (CSL). The slot selection phase always starts from slot number 2. In Figure 1, node A claimed and listed slot number 2 in its TSL. Node A then broadcasted the slot information to its parent (BS) and neighbors. Node A's neighbors are any node within its communication range, which are node BS, B, and E. Since BS is the intended receiver of A's transmission, BS listed the slot in its RSL, while B and E listed the slot in their CSL. Again, BS, B, and E broadcasted the slot information to their neighbors and hence C and D also listed slot number 2 in their CSL. In this way, a slot claimed by a node is informed to neighbors that are within twice of the node's communication range. Since FlexiMAC prevents nodes that are potentially within the interference range of each other to claim the same slot, collision-free traffic can be guaranteed. 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 node D and E send their packets to their parents, while C and BS expect to receive a packet from their respective children in that slot. After all nodes in the network built their data gathering schedule, they claimed their MFS.

III. LOCAL SYSTEMATIC RESOURCE TRANSFER

The human manager can define thresholds on sensor nodes that are used as event triggers and specify management tasks to be executed when the events occur. Throughout this paper, we will describe the SRT method using the following terms: *self-reporting node* refers to a sensor node that detects an event trigger and requests for extra slots, and *resource-supplier node* refers to a node whose sensor data do not change much and allows the temporary use of its slots by a self-reporting node.

In the decentralized scheme, a sensor node becomes a self-reporting node upon the occurrence of externally triggered events. It then initiates information exchange to try to get extra slots from nodes in its neighborhood through executing

the local SRT function. A neighboring node is a prospective resource-supplier node if it is not a self-reporting node and it has not lent its slots to a self-reporting node. A resource-supplier node only lends its data gathering slots not its MFS. The rationale behind it is to keep nodes in the network synchronized at all times, and to allow the human manager to access (query) the nodes even if their data gathering schedule are shut down or borrowed by other nodes. Note that a self-reporting node uses a resource-supplier node's schedule only for a specified length of time. They then simply resume to their previous schedule once this time expires.

Figure 2 (a) shows that node B , F , G , and H sense a sensor reading that exceeds the user-defined threshold. These nodes then become self-reporting nodes that request for one extra slot in a data gathering cycle from nodes in their neighborhood, in order to report their readings to the base station more frequently. The main task of the local SRT function is to find and select a resource-supplier node. Three techniques in finding a prospective resource-supplier node are to get extra slots from:

- 1) a self-reporting node's sub-network,
- 2) a self-reporting node's parent's sub-network, or
- 3) a self-reporting node's neighbors' sub-network.

A self-reporting node first checks if it can get extra slots from its sub-network. It checks if any of its children and descendants can be a prospective resource-supplier node. For example, in Figure 2 (a), the self-reporting node B waits for its children and descendants to send their packets to it during their scheduled data gathering slots. The children and descendants are prospective resource-supplier nodes if they do not piggyback an extra slot request or a notification that they have become a resource-supplier node for other self-reporting nodes. The self-reporting node then selects the immediate available slot and claims the resource-supplier node's data gathering slots. In the example, node C , the only child of the self-reporting node B , can lend its resources to B . In MFS, the self-reporting node B then informs the selected resource-supplier node C to shut down its schedule temporarily. Since the self-reporting node is one of the resource-supplier's routers, in the next data gathering cycle, the self-reporting node just claims the resource-supplier's data gathering slot that is already listed in its TSL table, i.e., the transmit slot that is originally used by node B to forward node C 's data is now used for sending its own data. Thus, this approach does not require extra communication slots to perform the local SRT function and hence reduces energy consumption.

If none of the self-reporting node's children or descendants could become a resource-supplier node, the self-reporting node tries to get extra slots from its parent's sub-network. It piggybacks its extra slot request onto the data packet sent to its parent using one of its descendant's data gathering slots. The parent then checks if any of its children or descendants other than the self-reporting node can lend its slots. Again, the parent selects the immediate available resource-supplier node for the self-reporting node. In MFS, the parent then

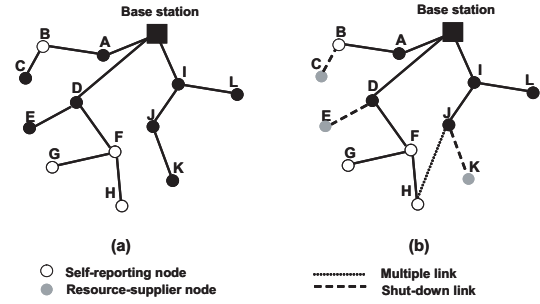


Fig. 2. (a) Four self-reporting nodes in the network: B , F , G , and H need to find a resource-supplier node. (b) After they execute the local SRT function, B , F , and H manage to get extra slots from the resource-supplier node C , E , and K respectively. Thus, B and F use their existing path, and H uses multiple paths, to send their data to the base station more frequently. Whilst, C , E and K shut down their schedule temporarily.

informs the self-reporting node that the extra slot is available, and also informs the selected resource-supplier node to shut down its data gathering schedule temporarily. For example, Figure 2 (a) shows that node F is unable to use its children's data gathering slots (node G and H) because they are also self-reporting nodes. Node F then requests extra slots from its parent, node D . Since node E is a prospective resource-supplier node, node D allocates E 's TSL slot to F . Thus, in the next data gathering cycle, the self-reporting node F uses the resource-supplier node E 's data gathering schedule in addition to its own schedule.

If none of children or descendants of the self-reporting node's parent can lend their data gathering slots, the self-reporting node waits for the FTS to send its extra slot request. This technique allows the self-reporting node to try to get extra slots from its neighboring sub-networks. In FTS, a self-reporting node broadcasts a *SRT_signal*, indicating that it requires extra slots. A neighboring node replies to the *SRT_signal* only if it has any child or descendant that is eligible to become a resource-supplier node and so the self-reporting node can use that node's child or descendant's data gathering slots to send data to the node. Following the previous example shown in Figure 2 (a), node G and H are unable to get a resource-supplier node during the data gathering period. In FTS, node H receives a reply from node J that it can use node K 's data gathering slots (node J 's child) and so H can use K 's slot to send its data to J . After FTS finishes, J informs K and K 's routers to update their lookup table with the new schedule, by piggybacking the schedule update onto data gathering and MFS packets in the current data gathering cycle. In the next data gathering cycle, H uses multiple paths to send its data more frequently to the base station, i.e., H gets extra slots for data gathering by using its resource-supplier node's path in addition to its own path. These multiple paths are: 1) H - F - D -Base station and 2) H - J - I -Base station. The self-reporting node G is unable to get any resource-supplier node within its communication range and so its extra slot request is propagated up to the base station in the next data gathering cycle. This is where the central SRT function comes into place.

IV. CENTRAL SYSTEMATIC RESOURCE TRANSFER

The central manager has an unlimited energy and so it is able to execute a wide-range of management functionalities by regularly analyzing sensor readings reported by all nodes in the network. For example, Figure 3 shows sensor readings in the shaded sub-network change rapidly compared to the rest of the network. This phenomenon cannot be detected locally as individual nodes have no global knowledge of the network. The central SRT function is then executed to allocate extra slots for that needy sub-network and so nodes in this sub-network can send their data to the base station more often. In the centralized scheme, the central manager performs sensor data analysis, processing and resource allocation. Thus, the central SRT function is cost-free since it removes the computation burden from energy-constrained sensor nodes. The central SRT function consists of four main steps: resource demand, resource-supplier selection, resource assignment, and resource delivery. The steps will be illustrated through an example shown in Figure 3.

Resource demand - The central manager decides the extra slots (let this number be α) required by a self-reporting node to send its data in a data gathering cycle based on management policy of a particular event. When a node asks for one extra slot, the central manager should also allocate one extra slot for each of the self-reporting node's routers. Thus, the total number of data gathering slots required to meet the self-reporting node's request is equal to α multiplied by the self-reporting node's tree level (the number of hops required by the node to reach the base station). Since the central manager knows the network connectivity, it can determine how many hops away the self-reporting node is from it. For simplicity, each self-reporting node in Figure 3 requires one extra slot in a data gathering cycle. But in general, each self-reporting node may ask for more extra slots. Thus, the number of extra slots required by self-reporting nodes in Figure 3(a) is equal to nine slots: one slot for *D*, two slots for *E*, three slots for *G*, and three slots for *H*.

Resource-supplier selection - The central manager determines which nodes are eligible to supply extra slots requested by a self-reporting node, and determines how many resource-supplier nodes are required to meet the demand. The resource-supplier node's ordering of selection criteria are as follows:

- 1) *Path sharing*. The resource-supplier node shares the same path as the self-reporting node and so some of the self-reporting node's routers may not need to rebuild their schedule.
- 2) *Same tree level*. The resource-supplier node has the same tree-level as the self-reporting node.
- 3) *Higher tree level*. The resource-supplier node has a higher tree-level than the self-reporting node.
- 4) *Lower tree level*. The resource-supplier node has a lower-tree level than the self-reporting node. Thus, more resource-supplier nodes are required to meet the self-reporting node's demand.

In the case where the total demand of all self-reporting

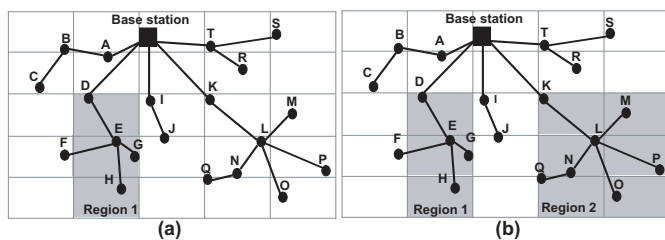


Fig. 3. (a) Sensor readings in the shaded Region 1 change rapidly in a period of time. The supply of extra slots from the entire network is higher than the demand of extra slots by self-reporting nodes in the Region 1. (b) Sensor readings in both shaded Region 1 and 2 change rapidly. The supply of extra slots is less than demand of extra slots by self-reporting nodes in the Region 1 and 2.

nodes is less than supply, any eligible resource-supplier node can supply their slots to each individual self-reporting node according to the above criteria. For example, Figure 3(a) shows that any nodes other than self-reporting nodes in the shaded Region 1 can lend their slots, e.g., node *K*, *L*, *N*, and *O* can become the resource-suppliers for the self-reporting nodes *D*, *E*, *G*, and *H* respectively.

Figure 3 (b) shows an example where extra slots in the network are limited such that demand is higher than supply. We propose three techniques to address this issue: data aggregation, round-robin cycle strategy, and combination of data aggregation and round-robin cycle strategy. The data aggregation technique can address the demand-supply problem if self-reporting nodes within a sub-network report similar data values over a period of time. The nodes' data can be aggregated instead of forwarding the nodes' individual data, hence reducing the number of extra slots required. For example, the self-reporting node *E*, *G*, and *H* in Figure 3(b) reported similar data values in the past three data gathering cycles because their locations are within proximity of each other. The central manager then allocates fewer extra slots to these nodes by allowing node *D* to aggregate these nodes' data prior to sending them. Instead of allocating nine extra slots, only five extra slots are assigned to these nodes: one slot each for *H* and *G* to send data to *E*, one slot for *E* to send the aggregated data of its children and its own data to *D*, one slot for *D* to forward the aggregated data received from *E* to the base station, and one slot for *D* to send its own data to the base station. In the round-robin strategy, self-reporting nodes may share the same extra slots and use them alternately in alternate data gathering cycles. For example, self-reporting nodes in the shaded Region 1 can use extra slots in the immediate data gathering cycle after receiving them in the current cycle, then wait for one cycle before using the extra slots again. Consequently, when self-reporting nodes in the shaded Region 2 receive the extra slots in the current cycle, they need to wait for one more data gathering cycle prior to using the extra slots.

Resource assignment - The central manager is responsible for ensuring that the supplied slots do not conflict with the schedule of self-reporting nodes and their routers (parents

and ancestors). This approach avoids schedule duplication and collisions by simply checking whether the supplied extra slots exist in the tables (RSL, TSL, and CSL) of a self-reporting node and its routers. The central manager performs this function by coupling node schedule information with node neighborhood information and so the central manager can construct the RSL, TSL, and CSL table of nodes in the network. For example, in Figure 3(a), the resource-supplier node *C* can lend its schedule to the self-reporting node *E*. Say, node *C*'s data gathering schedule is slot number 5, 6, and 7. Since node *C* has a higher tree level than *E*, only two slots of *C* is given to *E* that are slots 5 and 6. The central manager checks whether slot number 5 is in *E*'s tables. Since *E* is one of *C*'s neighbors, *C*'s TSL is recorded in *E*'s CSL. However, now *E* can use slot number 5 because *C* has given up its schedule, i.e., *C* will no longer be transmitting data during that slot for a specified period. Again the same slot checking is performed at each of the self-reporting node's routers. Thus, the next slot checking is performed on node *D* (*E*'s parent) and *B* (*C*'s parent). If a resource-supplier node shares the same path as the self-reporting node, a slot checking is only performed on uncommon paths. In Figure 3(a), the resource-supplier node *F* and the self-reporting node *H* shares the same path at node *D* and *E*. Node *H* can simply use *F*'s existing data gathering schedule that is already listed at their common routers. Furthermore, in the case where a number of resource-supplier nodes is required for a single reporting node, the central manager first filters and sorts extra slots provided by the resource-supplier nodes in an increasing order, and then the similar slot checking process is executed.

Resource delivery - In MFS, the central manager propagates the extra slots allocated to self-reporting nodes and the information on how long these reporting nodes can use those extra slots. Thus, the systematic resource transfer packet is piggybacked on the MFS packet. The systematic resource transfer packet contains a list of extra slots, a list of self-reporting nodes that should update their schedules with the allocated extra slots, a list of resource-supplier nodes that should release their slots, and the timer for the extra slots and temporarily released slots (in terms of the number of data gathering cycle). A node that receives the packet checks whether its ID or any of its descendant's ID is on the list of self-reporting nodes or resource-supplier nodes. If a node or a node's descendant is on the list of self-reporting nodes, the node updates its table with the allocated extra slot and sets the timer for the extra slot. Whilst, if a node or a node's descendant is on the list of resource-supplier nodes, the node shuts down the corresponding released slot (i.e., the data gathering slot that belongs to the node or the node's descendant), and also sets a timer for the temporarily released slots. The node then updates the systematic resource transfer packet by removing the retrieved data from the packet prior to propagating this packet to its children. If a node and a node's descendants is not on the list, the node will drop the piggybacked systematic resource transfer data and only propagate the MFS packet. In this way, the systematic resource transfer packet is only

propagated to intended nodes, not all nodes in the whole network. Thus, the central SRT function allows the network to reconfigure its traffic pattern within one data gathering cycle in an energy-efficient manner, i.e., in a data gathering cycle, the self-reporting nodes can get extra slots and the resource-supplier nodes are informed to shut down their data gathering schedule. After the timer for borrowing or releasing extra slots expires, the self-reporting nodes and their routers remove extra slots from their tables, while the resource-supplier nodes and their routers resume their previous schedule. Note that the central manager can extend the timer before the timer expires by simply propagating timer extension information in MFS.

V. SIMULATION RESULTS

We have developed a C# simulator to model the behavior of FlexiMAC data gathering sensor networks and simulate the SRT method. All of the simulation results are based on the mean value of 50 different network topologies involving 200 nodes with a default transmission radius of 30 meters (m), located randomly in a network area of 300 m x 300 m. The location of the base station was fixed at the top-center of the network map. The base station was programmed to find at least three children in order to spread energy dissipation among nodes. In all experiments, the self-reporting node is restricted to ask for one extra slot in a data gathering cycle. Though, the SRT method allows a self-reporting node to ask for more extra slots.

In the first experiment, we gradually generated sensor data with value exceeding the user threshold in some nodes in a network. We started with introducing the event-triggered sensor data to 10 nodes and then gradually introduced it to up to 70 nodes in a network. This experiment is used to measure the performance of local SRT and pure CSMA in terms of energy efficiency and capability to support non-uniform sensing. In our simulations, we compare local SRT configured with FTS (CSMA period) equal to one second to pure CSMA configured with acknowledgement scheme and a random backoff of 26 to 130 ms (equivalent to 1 to 5 FlexiMAC slots) if the channel is busy. The period allowed for all nodes in a network to try to send their data using pure CSMA is equal to FlexiMAC data gathering cycle period for that particular network topology.

We measured energy efficiency by calculating the total transmissions required by a self-reporting node to try to send its data. In pure CSMA, the total transmissions is equal to the number of transmissions and retransmissions made by a self-reporting node to try to send its data. In local SRT, if a self-reporting node gets extra slots from its sub-network, the number of transmission is one, which is for sending a packet to inform the selected resource-supplier. If a self-reporting node gets extra slots from its parent's sub-network, the total number of transmissions is two: one for sending an extra slot request to its parent and one for receiving extra slots from the parent. Furthermore, the total number of transmissions of a self-reporting node that tries to get extra slots from its neighborhood sub-networks is equal to the number of

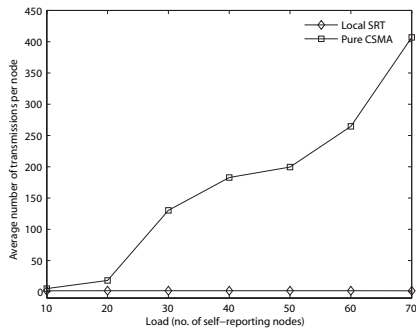


Fig. 4. Average number of transmissions per node versus load (number of nodes with demand).

transmissions and retransmissions during contention in the FTS. Figure 4 shows local SRT maintains a very low energy overhead compared to pure CSMA across diverse demand. In local SRT, self-reporting nodes always try to get extra slots through FlexiMAC scheduled-communications before resorting to CSMA in the FTS. Whilst, in pure CSMA, as the demand increases, the contention increases and so results in high collision and retransmission.

Figure 5 shows the percentage of unmet demand using either pure CSMA or local SRT. As the demand increases, pure CSMA performance in supporting non-uniform sensing starts to deteriorate significantly, while local SRT still meets around 90% of the demand. In pure CSMA, we calculated the percentage of self-reporting nodes in a network that are unable to send their data within the given period, which is equal to the FlexiMAC data gathering cycle period for that particular network. In local SRT, we calculated the percentage of self-reporting nodes that are unable to get extra slots locally within a FlexiMAC data gathering cycle. Note that although a self-reporting node fails to get extra slots locally, it may get the slots centrally through central SRT.

In the second experiment, we generated rapid data changes in 50 to 175 nodes in a network to measure the performance of central SRT. In the experiment, a parent and its children in a sub-network reported similar sensor data. This sub-network state allows central SRT to utilize the data aggregation technique when these nodes became self-reporting nodes. Thus, central SRT can take even more demand. Table I shows the percentage of unmet demand using central SRT versus rapid data change demand. As the demand increases, the supply of extra slots in the network decreases and so results in a reduced demand allocation. Simulation results show central SRT satisfies more than 85% of the demand even in high demand cases.

VI. CONCLUSIONS

We presented a novel TDMA protocol for transferring time slots between nodes to support non-uniform and reactive sensing in different parts of a network. The systematic resource transfer can be executed locally or centrally. Autonomous

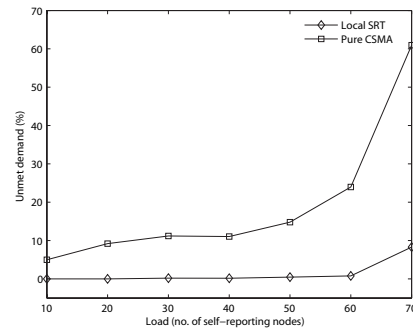


Fig. 5. Unmet demand versus load (number of nodes with demand).

Load	50	75	100	125	150	175
Unmet demand	0%	0.3%	0.5%	2.6%	3.7%	13.4%

TABLE I

UNMET DEMAND VERSUS RAPID DATA CHANGE LOAD (NUMBER OF NODES WITH DEMAND).

sensor nodes perform the local systematic resource transfer function based on their local neighborhood state. Upon the occurrence of externally triggered events, a sensor node becomes a self-reporting node and initiates information exchange to try to get extra slots from nodes in its neighborhood. The central manager regularly analyzes sensor data reported by all sensor nodes and performs the central systematic resource transfer function when it detects event triggers, providing the potential for better monitoring, control, and management of sensor networks. Our simulation results confirm that local systematic resource transfer function performs better than pure CSMA in terms of energy efficiency and capability to support non-uniform sensing. In the simulations, a self-reporting node only asks for one extra slot in a data gathering cycle. Clearly, pure CSMA will perform much worse if a self-reporting node asks for more than one extra slot. Furthermore, central systematic resource transfer performs well under varying demand, due to data aggregation and round-robin allocation strategy whenever the demand is higher than supply.

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," in *Proc. ACM SenSys Conf.*, Nov. 2004.
- [3] K. Langendoen and G. Halkes, *Embedded Systems Handbook*, chapter Energy-efficient Medium Access Control, CRC Press, 2005.
- [4] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves, "Energy-Efficient Collision-Free Medium Access Control for Wireless Sensor Networks," in *Proc. ACM Sensys Conf.*, Mar. 2003.
- [5] M.J. Miller and N.H. Vaidya, "A MAC Protocol to Reduce Sensor Network Energy Consumption Using a Wakeup Radio," *IEEE Transactions on Mobile Computing*, vol. 4, no. 3, pp. 228–242, 2005.
- [6] 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," in *Proc. IEEE ICON Conf.*, Sep. 2006.