A New Power-Efficient Scheme to Deliver Time-Sensitive Data in Sensor Networks

Shanzhong Zhu, Wei Wang, and Chinya V. Ravishankar
Department of Computer Science and Engineering
University of California, Riverside, CA 92521
Email: {szhu, wangw, ravi}@cs.ucr.edu

***Abstract*— We present a power-efficient scheme to deliver time sensitive data packets in sensor networks. Data generated by sensors are frequently time sensitive in applications such as hazard monitoring systems, traffic controlling systems and battlefield commanding systems. Such data are associated with end-to-end deadlines, within which they must reach the base station. We make two contributions in this work. First, we propose a novel load-balanced routing scheme that distributes data packets evenly among the nodes relaying data towards the base station, avoiding bottlenecks and increasing the likelihood that packets will meet their deadlines. Second, we propose a method of grouping smaller packets into larger ones by delaying data transmissions at the relaying nodes whenever slack times are positive. Our packet grouping scheme significantly reduces packet transmissions, reduces congestion, and saves power in the sensor network. We verify the effectiveness of our approach through extensive simulations using the `ns-2` simulation package.**

I. Introduction

There has been considerable interest in large networks of wireless sensors in recent years. Typically, sensors generate readings continuously, and deliver the data to a base station (BS) through wireless channels. Since wireless sensors are battery powered, and recharging is expensive or even impossible in harsh conditions, it becomes important to conserve power.

Real-time sensor networks have also received much research attention recently [1]–[3], since sensor data are frequently time-sensitive. For example, freeway traffic information must be delivered to the monitoring center promptly for real-time traffic reports. Similarly, rapidly changing conditions in hazard-monitoring or battlefield situations must be reported quickly to the BS. In all such cases, the data collected by sensors must arrive the BS before a deadline to ensure freshness and correctness.

The power needed for a wireless transmission increases as the square (or higher power) of the radio communication radius [4]. Since sensors are power-limited, data packets generated by sensors are typically delivered to the BS over a series of intermediate hops, rather than as a single long-range transmission. While this strategy is unavoidable, it does increase the inherent unpredictability of wireless channels. We therefore assume that user-specified deadlines are *soft*, meaning that packets arriving late are still useful. Our goal is to ensure a high confidence that packets will arrive at the BS before the deadline.

We use the term *sensor readings* to denote the values that originate at sensors. These are typically a few bytes in size. In contrast, *data packets* are created at relaying nodes to encapsulate one or more sensor readings, and serve as the units for network transmission. Packet size grows with the number of sensor readings encapsulated, and is limited by the network MTU. *Real-time deadlines* are associated with sensor readings, which must be delivered to the BS within the deadline. No deadlines are initially associated with packets, although a packet may implicitly inherit the deadline of the most urgent sensor reading it holds.

A. Power-Efficient Real-Time Delivery

We achieve both real-time delivery and power efficiency through two effective approaches: *load-balanced routing* and *packet aggregation*. We first apply load-balanced routing to distribute traffic as evenly as possible across the network, thereby reducing end-to-end delays and balancing traffic loads on the relaying nodes. Balancing loads is very important for real-time sensor networks, since a heavily loaded node will die soon and cause urgent data to be delayed or lost. Our load-balanced routing scheme guarantees even traffic distribution

on the nodes at each level. It boosts performance, such as power consumption and end-to-end delay. Next, we perform packet aggregation by grouping smaller data packets into larger ones at relaying nodes, thereby reducing the number of packets sent. We are able to use this strategy, since all data packets are destined for the BS.

Data transmission is often the dominant source of power consumption in sensor networks [5]. Aggregated packets convey the same payload as a series of smaller packets, but are far more power-efficient. Among other savings, we would transmit fewer packet headers, and also send fewer MAC-layer control packets (RTS/CTS/ACK), since channel contention is lowered.

Our aggregation strategy requires a relaying node to hold arriving data packets, accumulate a number of sensor readings, and regroup them into a larger packet. Since each sensor reading is subject to an end-to-end deadline, such grouping is possible only when the actual end-to-end transmission delay for a sensor reading is less than its deadline. The *slack time* of a sensor reading is the difference between its deadline and the source-to-BS transmission delay. A positive slack time allows the sensor reading to be held for some time at the relaying nodes along the path, without missing its deadline. Among the important questions we address is how to distribute slack time among the relaying nodes.

The work in RAP [1] and SPEED [2] addresses real-time data delivery in sensor networks. RAP introduces a novel *Velocity Monotonic Scheduling* algorithm to prioritize real-time packets at each node, depending on its distance to the BS and on packet deadlines. SPEED aims to meet deadlines by maintaining a desired packet delivery rate across the network. Our packet aggregation mechanism can enhance both RAP and SPEED, since fewer transmissions lead to lighter scheduling loads in RAP and make it easier in SPEED to sustain the desired delivery rate. SPEED uses non-deterministic geographic forwarding to balance traffic among multiple paths. Our load-balanced routing scheme has the same goal, but balances loads better among nodes at the same distance to the BS.

B. Our Contributions

We make several contributions in our work. First, we present a novel method for routing packets which balances traffic over the relaying sensor nodes. Balancing traffic contributes to both real-time delivery and power conservation in sensor networks. Our routing algorithm routes packets to the BS over multiple paths, so that the traffic on each node is distributed evenly.

Second, we show how to reduce the number of transmissions by holding and grouping sensor readings at the relaying nodes. We propose an algorithm to calculate the hold times for each sensor reading. When a sensor reading reaches its permissible hold time at a relaying node, a packet is formed by grouping all accumulated sensor readings and sent out. We study the performance of our packet aggregation scheme on top of the load-balanced routing scheme. However, our packet aggregation scheme is intended to complement any underlying routing scheme.

Finally, we perform extensive simulations on the ns-2 platform to verify the feasibility and efficiency of our scheme. We simulate sensor networks of different sizes based on the 802.11 MAC protocol, and measure both the deadline miss ratio and the power consumption. Our results show that packets can make their deadlines with high confidence and low power consumptions.

The rest of this paper is organized as follows: We review some related work in Section II. Our system model is introduced in Section III. In Section IV, we propose our load balanced routing scheme, which aims to balance the traffic over the relaying nodes. In Section V, we discuss our packet grouping mechanism that allows relaying nodes to combine packets together. Simulation results are shown in Section VI to verify our approach. Section VII concludes our work.

II. Related Work

The literature largely addresses, in isolation, the issue of real-time packet delivery in the demanding sensor network environment. For example, SWAN [6] proposes a stateless network model to deliver service differentiation in wireless ad-hoc networks. To address the delay requirements for the real-time traffic, rate control of the best-effort TCP and UDP traffic is performed at each node. In our work, we assume all data have real-time requirements. Our mechanism allows data items with looser deadlines to be held longer at the relaying nodes and grouped with those having tighter deadlines, so that we can form larger packets.

A novel real-time routing protocol, SPEED, was introduced in [2], with the goal of maintaining

a desired packet delivery rate across the sensor network, so that end-to-end delay becomes proportional to the source-destination distance. Each sensor chooses the neighbours that can sustain the desired packet delivery rate. If no such neighbours exist, packets are dropped to reduce congestion. Our approach can enhance SPEED by performing load-balanced routing and packet aggregation. Both mechanisms contribute to maintaining a higher delivery rate of real-time packets and reducing data transmissions in the network.

Several MAC layer protocols have been designed to accommodate real-time requirements. RAP [1] is a new real-time communication architecture for sensor networks. Its *velocity monotonic scheduling* mechanism is a key component in prioritizing real-time traffic at the MAC layer. In [7], an EDF-based MAC layer protocol was proposed. The periodicity of sensor-generated traffic allows contention to be resolved implicitly, without need for exchange-control packets for channel reservation. Our approach, in contrast, requires no real-time support from the MAC layer, and can work with existing MAC protocols such as 802.11.

In-network aggregation at intermediate sensor nodes to reduce data transmissions has been studied [8]–[10]. It is typical to compute partial results at intermediate nodes and send them to the BS. In our work, we do not simply consider query-level aggregation, but ask a more basic question: Since all sensor data are destined for the same base station, how do we group a number of incoming data packets into a larger one at the intermediate nodes under real-time constraints?

The idea of combining small packets into larger packets was first studied in the Internet context [11], where small packets such as TCP ACKs and TCP SYNs can be combined at routers to improve end-to-end performance. The time by which a packet can be delayed at a router is simply given as a parameter. In our work, however, the allowable delay for each packet at any node is computed based on its deadline. A novel packet aggregation scheme was proposed in sensor networks [12], which utilized the queueing delays at each relaying node to group small packets into larger ones. Although the scheme also aimed to reduce overall transmissions and better utilize the channel, it was not studied in the real-time context, while our major contribution is how to assign the slack time across nodes.

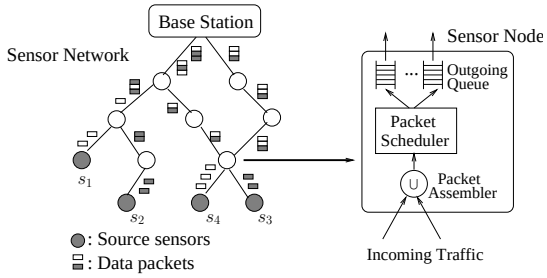


Fig. 1. The System Model

III. Our System Model

Sensor readings are the basic data units generated by sensors, and packets are units of transmission, and typically include multiple sensor readings. A deadline is associated with each sensor reading, depending on its urgency. We assume that deadlines are application-specific, and are determined *on-line*, at the time the sensor reading is generated.

Fig. 1 shows our real-time sensor network model. Let $\Omega = \{S_1, S_2, \cdots, S_n\}$ be the set of sensor nodes in the network, and $\mathbf{S} \subseteq \Omega$ be the set of *source* sensors from which readings originate. Sensor reading u_j is associated with a deadline d_j, the allowable elapsed time before it must reach the BS. At any time t, a sensor reading has the form: $(v_j, S_k, t_j^k, \tau_j(t))$, where v_j is the value of the reading, S_k is the source sensor, t_j^k is a generation timestamp, and $\tau_j(t)$ is the time remaining until the deadline. Initially, we will have $\tau_j(t_j^k) = d_j$. These values will be used to determine the permissible hold time for the reading at relaying nodes (see Section V).

A. Sensor Nodes

Our sensor node model is shown in Fig. 1. The *packet assembler* groups sensor readings from different incoming packets into larger packets. A sensor reading may be delayed upto a certain *hold time* determined by parameters such as the avaliable slack time, the incoming packet rate and maximum packet payload size. Sensor readings are grouped by the *packet assembler* until one of them reaches its hold time. At this point, the grouped packet is scheduled for transmission on an outgoing link by the *packet scheduler* based on our load-balanced routing scheme (see Section IV).

B. Data Generation Model

We assume sensor readings are generated *aperiodically* at sources with stringent deadlines. A typical scenario for us is 50–100 data items to be

generated each second in a network of a hundred sensors, and deadlines to be in the hundreds of milliseconds (see Section VI). This model is more general and challenging than assuming *periodical* generation of sensor readings, such as in [13].

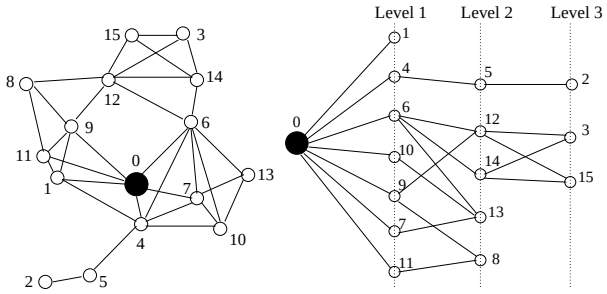
C. The Power Model

We achieve power efficiency by two means. First, we balance the power consumption at individual nodes by balancing the number of packets they transmit using our load-balanced routing scheme. Balancing power consumption extends network lifetime by avoiding bottleneck nodes that will exhaust their power quickly due to heavy transmission loads. Second, we manage to reduce the number of transmissions at each node by grouping small packets into larger ones, so that the power consumed by transmitting and receiving can be significantly reduced throughout the network.

IV. Load Balanced Routing

Routing in ad hoc networks has been extensively studied, and various routing schemes, such as DSDV [14], DSR [15], and AODV [16], have been proposed. These generally work well in dynamic environments. We focus on static wireless ad-hoc sensor networks, where nodes are immobile and all packets are headed for the same destination, namely the BS. Power limitations in sensor networks make traffic balancing a critical issue, since a congested node relaying a high volume of packets will soon exhaust its battery and fail. Moreover, a bottleneck node will cause packets to experience longer delays, possibly missing their deadlines.

Several load balanced routing schemes have been proposed for wireless sensor networks [17]–[19]. In [17], a *load balanced backbone tree (LBB-tree)* was constructed to balance the loads over the nodes that are one hop away from the BS. In LBB-tree scheme, all packets generated by a given source will follow the same path to the BS. In contrast, we adopt multi-path routing in our scheme, distributing packets over several paths to achieve better balanced traffic (see Section VI-B). Hong et al. [19] proposed a multi-path routing scheme for sensor networks, where a level *i* node *randomly* picks a level *i-1* neighbour with equal probability to relay its data packet towards the BS. As we will show in Fig. 2, our multi-path routing scheme can ditribute traffic more balanced than this simple random scheme.



(a) A sensor network (b) The routing network

Fig. 2. The routing network for 15 sensor nodes

We first show how to build a *routing network* based on which packets are delivered to the BS (see Section IV-A). Then, we propose a novel load-balanced routing algorithm in Section IV-B.

A. The Routing Network

A *routing network* (RN) is a DAG (*directed acyclical graph*) [20] which allows packets to reach the BS over multiple paths. A common approach to building an RN is to assign a *level* number to each sensor depending on its distance to the BS, and deliver data packets from higher-level nodes to lower-level nodes [8], [9], [18], [21]. The BS is at level 0. Each node at level i has one or more *parents* at level $i-1$ to which it can send packets. The level of a node and its parents are determined by the performance metric used, such as hop count, delay, or signal strength, and can be different by applications. In one application [8], [18], [19], the level of a node equals the number of hops in the shortest distance to the BS, and its parents are the neighbours on its shortest paths. In another [21], its level may be $L_f + 1$, where L_f is the level of the node it first hears from during the route construction phase, and its parents may be those level-L_f neighbours with low delays. Our load balanced routing algorithm works well with either scheme.

We use the same construction scheme as in [19]. Fig. 2 shows the RN for a simple network of 15 sensor nodes. Clearly, any path from a source to the BS is a shortest path in the resulting RN.

B. The Routing Algorithm

Given the RN, we propose a novel *load balanced routing* (LBR) scheme to balance traffic loads level by level, from highest to lowest. Each node must be able to determine what fraction of traffic should be assigned to each of its outgoing links, so that the traffic loads can be balanced across the nodes at

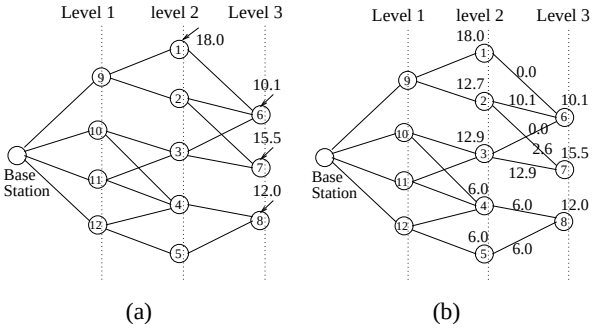


Fig. 3. *(a) A RN with 4 sources. (b) Final traffic distribution.*

the next (lower) level. We map the routing problem to the *maximum flow* problem [20] by constructing a *flow network* based on the link topology between the current and the next level. We then incrementally distribute the flow from the nodes at the current level to the nodes at the next level to achieve load balancing. We give an example in Fig. 3 to show how this approach works.

Fig. 3(a) shows a RN of 12 sensor nodes, among which nodes 1, 6, 7, and 8 are sources generating data packets at certain rates. We start by constructing a flow network for nodes in level 2 and 3.

1) **Constructing the Flow Network***:* Each level-2 or level-3 node in the RN is represented by a node in the flow network. A *virtual source VS* and a *virtual sink VT* are created. The edges and their capacities are set up as follows:

- An edge is created to connect *VS* to each level-3 node. Its edge capacity is the traffic load on the corresponding level-3 node. If there is any source node at level 2 (such as node 1), an edge connecting the *VS* to the node is also created, with the edge capacity being the load generated at the node.
- An *inter-level* edge is created to connect a level-3 node and a level-2 node if there is a link connecting the two nodes in the RN. The edge capacity is the traffic load on the corresponding level-3 node.
- A *sink edge* is created to connect each level-2 node to the virtual sink *VT*. The edge capacity is set to a small value initially, and adjusted incrementally until no more flows can be augmented to the edges (see Section IV-B.2).

The flows on the sink edges represent the loads on the corresponding level-2 nodes, while the flows on the inter-level edges represent the traffic on the corresponding links in the RN. We will balance the flows on the sink edges by adjusting their capacities incrementally.

2) **Incrementally Adjusting the Sink-Edge Capacities***:* Our key idea is to let the sink-edge capacities gradually *guide* the flows towards an even distribution. We initially set the sink-edge capacities to a small value so that all sink edges will be *saturated*, i.e., we maximize the flows on these edges so that they reach the edge capacities. We then increment the capacities by a small fixed value, η, recalculate the maximum flow, and check if the sink edges can still be saturated. We continue until some sink edge can no longer be saturated, indicating that no additional load can be placed on the corresponding node. The flow on that sink edge now has its final value. We can remove the sink edge, the corresponding node, and the associated inter-level edges from the flow network, and increment the capacities of the rest sink edges as before. This process stops when the maximum network flow reaches the total capacities out of the source *VS*, i.e., no more flows can be augmented to the sink edges. A detailed description of the algorithm as well as its complexity analysis can be found in [22].

After obtaining the final balanced loads (see Fig. 3(b)), we can assign a *link probability* $\rho_i(l_j)$ to each level-3 node S_i's outgoing link l_j, where $\rho_i(l_j) = f(l_j)/f(S_i)$. The values $f(l_j)$ and $f(S_i)$ denote the fraction of traffic on link l_j and the load on node S_i, respectively. When a packet must be routed from S_i, it will randomly pick a parent based on $\rho_i(l_j)$. For example, Node 7 will relay a packet towards Node 2 with probability 17%, and towards Node 3 with probability 83%. The traffic distribution between level 1 and 2 is determined in the same way.

C. Power Efficiency

Each node runs a copy of LBR so that it can locally determine how to assign its traffic to the outgoing links. Each node needs two pieces of information: the *load distribution* at the current level, and the *topology* of other nodes at the current level (see Section IV-B.1).

We propagate the loads to other same-level nodes as follows: Each node is set to work in *promiscuous* mode [23] so that it can overhear packets sent by its neighbours. Let the *siblings* of node s be those nodes that share a parent with s. Node s (at level i) propagates its load to its siblings by piggybacking its load value on regular data packets sent to its parents at level $i-1$. As the parents send these

packets one level down the RN, their neighbours at level i, i.e., the siblings of node s, overhear these packets and get the load value. In turn, these sibling nodes piggyback the load value on data packets sent to their parents, and so on. s's load value is thus propagated to all *reachable* level-i nodes. If two nodes are not reachable at level i, their traffic loads can be scheduled independently, as with nodes 6 and 8 in Fig. 3.

Our scheme is power-efficient since it causes no extra transmissions. A node needs to propagate its load value only when it is detected to change significantly since a small change in the load has little effect on the load distribution. Nodes can propagate topology information to its siblings through a similar mechanism. Since sensor nodes are relatively static in typical applications, the topology information only needs to be exchanged initially, and updated once in a while.

V. Packet Aggregation

All packets are destined to the BS. Each packet contains one or more sensor readings, each of which is associated with a deadline. We will form larger packets by holding and accumulating sensor readings at the relaying nodes. The longer we hold them, the better chance we have to group more sensor readings into a packet. However, a sensor reading can only be held for a time bounded by its deadline. An important question for us to consider is: Given the deadline for individual sensor readings, how to determine its *hold time* at each relaying node along the path to the BS, so that the overall number of packet transmissions can be minimized?

A. Determining Hold Times

Let sensor reading u_j arrive at relaying node S_i at time t_j by S_i's clock. Let $\tau_i(u_j, t)$ be the time remaining until u_j's deadline at time t. S_i will try to delay retransmission of u_j to accumulate more readings, and to form a larger packet. However, S_i may delay u_j for no longer than some maximum time, depending on factors such as u_j's deadline and the maximum packet payload size.

At any time t, let $h_i(u_j, t)$ be the permissible *hold duration*, before which S_i must dispatch u_j. Let $e_i(p)$ be the E2E transmission delay from S_i to the BS along path p. We define u_j's *slack time* $\delta_i(u_j, t)$ at time t as

$$\delta_i(u_j, t) = \tau_i(u_j, t) - \max_p\{e_i(p)\}. \qquad (1)$$

This slack time bounds the sum of all remaining hold times for u_j on the way to the BS. Since LBR allows data packets to reach the BS over multiple paths, we are safe in defining slack time in terms of the *maximum* E2E transmission delay between S_i and BS. We will discuss how to evaluate $\max\{e_i(p)\}$ in Section V-B. If $\delta_i(u_j, t) \leq 0$, u_j must be forwarded immediately. If $\delta_i(u_j, t) > 0$, we can afford to hold u_j at S_i. We now address the question of what fraction of u_j's slack time should be allocated to its hold time at each S_i.

In Section V-A.1, we provide an upper bound for u_j's hold time on each relaying node. In Section V-A.2, we discuss how u_j's hold time on S_i affacts its outgoing packet rate, based on which we formulate an optimization problem in Section V-A.3, minimizing the total number of transmissions along u_j's path to the BS.

To simplify our analysis, we assume that each sensor node has enough memory to accommodate all the data arriving during the specified hold time. In practice, we can bound the allowable hold time at each node to avoid memory overflow.

1) **Bounding Hold Times:** At any time t, let $\Theta_i(t) = \{u_1, u_2, \ldots, u_{\alpha(t)}\}$ be the set of sensor readings accumulated at S_i. Let the permissible hold durations for these sensor readings be $\{h_i(u_k, t)\}$, $u_k \in \Theta_i(t)$. Our approach is to dispatch a packet containing all the accumulated sensor readings as soon as one of them reaches the end of its permissible hold time. That is, we dispatch at time t^* a packet containing all readings in $\Theta_i(t^*)$, if we find $h_i(u^*, t^*) = 0$ for some sensor reading $u^* \in \Theta_i(t^*)$.

When sensor reading u_j reaches S_i at time t_j, we can compute $h_i(u_k, t_j)$ for all $u_k \in \Theta_i(t_j)$. At this time, let u^* be the reading with the shortest remaining hold time, that is, $h_i(u^*, t_j) = \min\{h_i(u_k, t_j)\}$. Clearly, u_j will be dispatched no later than $h_i(u^*, t_j)$, and there is no point in setting u_j's hold time larger this value. Hence,

$$h_i(u_j, t) \leq h_i(u^*, t), \quad t \geq t_j. \qquad (2)$$

For convenience, we will use $h_i(u_j)$ and $\delta_i(u_j)$ instead of $h_i(u_j, t_j)$ and $\delta_i(u_j, t_j)$, when no confusion can arise.

2) **Hold Time vs. Outgoing Packet Rate***:* If the deadline for an arriving u_j is so stringent that it is the most urgent sensor reading at S_i, u_j will expire its hold time first and govern the outgoing packet rate at S_i. We study how the hold times of

the urgent sensor readings affect the outgoing rate of the grouped packets at a given node, and use this analysis to propose a scheme to determine the hold times for each sensor reading.

Let u_j's slack time be so small that it will always exhaust its hold time earlier than any other sensor readings at each node along its path. That is, u_j will dictate the outgoing packet rate at each node. The following theorem characterizes the relationship between u_j's hold time and the outgoing packet rate at relaying node S_i.

Theorem 1. *At node S_i, let r_i be the aggregate rate of data packets received from its children and itself, and let $h_i(u_j)$ be u_j's permissible hold time when u_j arrives at S_i. If o_i is the outgoing rate of the grouped packets, then*

$$o_i = \frac{r_i}{r_i \cdot h_i(u_j) + 1} \tag{3}$$

By Little's Law [24], $r_i h_i(u_j)$ is the average number of packets accumulated before u_j leaves node S_i. Thus, by grouping $r_i h_i(u_j) + 1$ packets (together with the packet containing u_j) into one, the outgoing packet rate is $\frac{r_i}{r_i \cdot h_i(u_j)+1}$.

3) **Obtaining the Hold Times***:* If u_j governs the outgoing packet rates of the relaying nodes, we want to distribute the total slack time $\delta_0(u_j)$, obtained at source S_0^j, across the relaying nodes to minimize the number of transmissions. Let $\{S_0^j, S_1^j, \cdots, S_m^j, BS\}$ be the path taken by u_j, and $h_i(u_j)$ be u_j's permissble hold time on S_i^j ($0 \leq i \leq m$). Since longer packets will experience higher transmission error rates [25], we must bound the number of readings contained in a packet. Let M be the maximum payload size of the packet and $\bar{y}_i$ be the average incoming packet size at node S_i^j, both expressed in terms of the number of sensor readings. We want to obtain

$$\min \left(\sum_{i=0}^{m} o_i \right), \text{ subject to the constraints}$$

$$\sum_{i=0}^{m} h_i(u_j) \leq \delta_0(u_j), \tag{4}$$

$$\bar{y}_i(r_i \cdot h_i(u_j) + 1) \leq M, \quad 0 \leq i \leq m. \tag{5}$$

Constraint 4 restricts the total hold time to $\delta_0(u_j)$, and Constraint 5 restricts the outgoing packet size at each node to M. The term $\bar{y}_i(r_i h_i(u_j) + 1)$ represents the average outgoing packet size at S_i^j, since $r_i h_i(u_j) + 1$ is the total number of incoming packets grouped together. When the packet size reaches M, it should be sent out since there is no point in delaying it further.

Since o_i is governed by Equation 3, we have a nonlinear optimization problem in terms of $h_0(u_j), \cdots, h_m(u_j)$. We note that increasing hold times increases the size of packets but reduces their numbers, so that $\sum_i o_i$ decreases monotonically as $\sum_i h_i(u_j)$ increases. We therefore treat Constraint 4 as an equality and apply the *method of Lagrange multipliers* [26] to derive the optimal hold times that will minimze $\sum_i o_i$.

$$h_i' = \frac{1}{L_0}\left(\delta_0(u_j) + \sum_{q=0}^{m} \frac{1}{r_q} - \frac{L_0}{r_i}\right), \tag{6}$$

where L_0 is the level of the source node. We can next rearrange Constraint 5 to get

$$h_i'' = \frac{M - \bar{y}_i}{\bar{y}_i r_i}, \tag{7}$$

Finally, we incorporate Constraint 2 by writing

$$h_i''' = h_i(u^*, t_j). \tag{8}$$

We can now combine Equations 6, 7, and 8 to write

$$h_i(u_j) = \min\{h_i', h_i'', h_i'''\}, \quad 0 \leq i \leq m. \tag{9}$$

In practice, both h_i'' and h_i''' can be easily determined on S_i when u_j arrives. To obtain h_i', we must calculate $h_i' = \frac{1}{L_i}\left(\delta_i(u_j) + \sum_{q=i}^{m} \frac{1}{r_q} - \frac{L_i}{r_i}\right)$, where $\delta_i(u_j)$ is u_j's remaining slack time on arriving on S_i. However, it is difficult to evaluate $\sum_{q=i}^{m} \frac{1}{r_q}$ since it is unknown which path u_j will take after S_i, due to our LBR mechanism. But we do know that h_i' will take the most conservative (minimum) value if the path u_j takes is the most heavily loaded, that is, the value of $\sum_{q=i}^{m} \frac{1}{r_q}$ is the minimum among those of all possible paths from S_i to the BS. Since we need to collect the 1-hop delay values along the path with the maximum transmission delay to determine the slack time (see Equation 10), we can propagate the estimated r_q along with its 1-hop delay to S_i without additional cost (see Section V-B). The $\{h_i'\}$ value thus obtained is conservative, but incurs little overhead. We show in Section VI-C.1 that our approach performs far better than other schemes, and is easy to implement. A detailed description of our scheme can be found in [22].

B. E2E Delay Estimation and Propagation

The E2E transmission delay $e_i(p)$ is the sum of the 1-hop delays along path p, that is,

$$e_i(p) = \sum_{S_k \in p} l_{k,k+1}, \tag{10}$$

where S_k is the kth node on p and $l_{k,k+1}$ denotes the 1-hop delay from S_k to the next hop S_{k+1}. We estimate the 1-hop delay as in [2]. At node S_k, we record the round-trip time $\gamma_{k,k+1}$ between the time a packet arrives at the outgoing queue and the time the MAC-layer ACK is received for that packet. Now, $l_{k,k+1}$ is estimated as $l_{k,k+1} = \frac{1}{2}(\bar{\gamma}_{k,k+1} - \bar{v}_{k+1})$, where v_{k+1} is the ACK processing time at the receiving node, which can be piggybacked in the ACK packet.

The 1-hop delay must be estimated periodically, and propagated when a significant change is detected. Each node maintains the maximum E2E delay value e_{max} and the e_{max} value for each of its parents. Whenever node S_k (at level L) detects a significant change in its 1-hop delay, it updates its e_{max} by recomputing the E2E delay along each of its parents and choosing the largest value. Now, S_k piggybacks its new e_{max} in a regular data packet sent towards the BS. Meanwhile, S_k's neighbours at level $L+1$ can overhear the packet and obtain the new e_{max}, based on which they can update their own e_{max} values. These $(L+1)$-level nodes, in turn, piggyback their e_{max} in regular data packets so that their neighbours at level $L+2$ can overhear it. In this manner, S_k's new 1-hop delay will be propagated to all reachable higher-level nodes, and their maximum E2E delays will be updated. This approach is power efficient since we exploit the ability of sensor nodes to overhear other nodes.

VI. Experiments

We evaluated our approach through extensive simulations on the ns-2 simulator [27].

A. The Simulation Setup

We used a single BS with multiple sensors deployed uniformly in a square region of $1500 \times 1500\ m^2$, with the BS at the center. We simulated networks of 100 and 150 nodes, with a subset of the nodes selected as *sources* that generated sensor readings periodically. The generation rates were varied to reflect different workloads. All sensor data were delivered to the BS. We chose UDP as our transport layer protocol, and 802.11 [23] as our MAC protocol, which was also used in other real-time sensor networks, such as [1] and [2].

Each sensor reading was set to 8 bytes and the maximum packet payload size was set to 128 bytes in our simulations. The default 802.11 MTU size (1500 bytes) is too large to serve as an effective test of our approach. We used the *shadowing* model [28] as our radio propagation model. We set the value of *path loss exponent* [27] as 2.0, and the value of *shadowing deviation* as 4.0, representing a typical outdoor environment. We set the radio communication range as $250m$ and chose the rate of correct reception as 0.95. We used the AODV routing protocol as the basis for our comparisons. AODV discovers a single fixed route for each source.

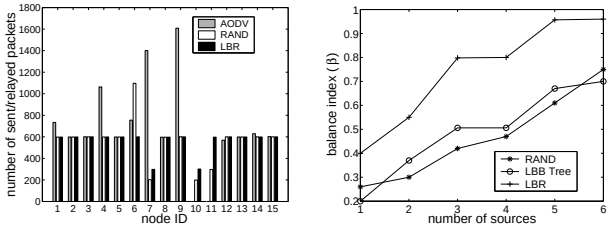
To measure power, we adopted the power parameters from the *Chipcon* CC1000 RF transceiver [29], which is used as the radio module in both MICA2 and MICA2DOT [30] sensor models. When operated at $433MHz$, its receiving power is $22.2mW$, and transmitting power is $31.2mW$, with the output power of $0dBm$. In our experiments, each node was set to the same power level initially, and we measured the remaining power after the simulation ran for some time.

B. Performance of LBR

1) **Load Balancing***:* We first simulated LBR on the sensor network shown in Fig. 2 for 300 seconds. Fig. 4(a) shows the traffic load distribution for all nodes. All *leaf* nodes (1, 2, 3, 8, 13, and 15) generated sensor readings at the rate of 2 units/sec. We compared LBR with AODV and the *random* (RAND) routing scheme. In RAND, since each node randomly picks an outgoing link in the RN with equal probability to send/relay a packet, the traffic distribution may not be even. Our results show that the loads are more balanced under LBR. We note that nodes 4, 7, 9 relay high traffic volumes under AODV. RAND balances traffic loads better than AODV in general, but node 6 relays heavy traffic under RAND. Traffic loads are very balanced in LBR.

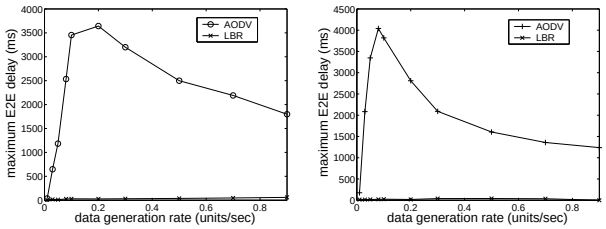
We also compare LBR with RAND and LBB-Tree [17], on the same topology, using the *Balance Index* ($\beta(S)$) metric [17] which measures the degree of load balancing among a given set of nodes S. Generally, the higher β is, the more balanced the loads are. The loads among nodes in S are perfectly balanced if $\beta(S) = 1$.

In Fig. 4(b), we show β on the level-1 nodes, since nodes at level 1 are very likely to be most heavily loaded, and balancing their loads is important. Sources were randomly chosen from the leaf nodes, and we calculated the average β value for a given number of sources. Each source generated readings at the rate of 2 units/sec. The β values



(a) Load distribution (b) β value (level-1 nodes)

Fig. 4. LBR performance for network in Fig. 2.



(a) 100 sensor nodes (b) 150 sensor nodes

Fig. 5. Average of max E2E delays (without packet aggregation)

for our scheme are much higher than those for the LBB-Tree and RAND, showing that our scheme balances loads much better. When there are only a few sources, the β values are low for all three schemes, since these few sources may not require all nodes to relay traffic. LBR can achieve a β value as high as 0.95.

2) **E2E Delays***:* Fig. 5 illustrates the maximum E2E delays from sensor sources to the BS for different node densities. The maximum E2E delay for a given sensor network is the longest delay experienced by packets, and measures the worst-case packet delay in the sensor network. We randomly generated ten node deployments for each sensor density, and averaged the maximum E2E delays over the deployments. All sensors generated data at the same rate, in the range 0.1– 0.9 units/sec.

As shown in Fig. 5, the maximum E2E delay under AODV is much higher than under LBR. Under AODV, the E2E delay increases drastically at the beginning and then starts to drop, due to the fact that a large percentage of packets get dropped in the network (see Fig. 7). The E2E delay value stays extremely low and stable under LBR. The significantly longer delay under AODV is due to two reasons. First, packets experience longer queueing delays at highly congested nodes. Second, the contention for the wireless channel is more fierce at congested nodes, also increasing delays. Packets clearly experience shorter delays under LBR, as we manage to remove the bottleneck nodes in the network by distributing the loads more evenly. Thus, packets can meet more stringent real-time requirements under LBR.

C. Performance of Packet Aggregation

Having verified the benefits of LBR, we next evaluated how the packet aggregation mechanism further reduces transmissions. Unless explicitly specified, we ran the simulations in this Section for 350 seconds on a topology of 100 nodes. Fig. 6 shows the total number of network-level packet transmissions in the entire network. All sensors generated data at the same rate, and we associated a single deadline for all sensor readings from a given source. The deadline was set to df times the average E2E transmission delay from the source to the BS. We compared our packet aggregation scheme (LBR-G) under $df = 2$ and $df = 10$ with LBR (without aggregation) and AODV. LBR is a special case of LBR-G with $df = 1$.

The LBR-G scheme sends far fewer packets than AODV when the data generation rates are low. However, after rate 0.3 units/sec, the total number of packet transmissions under AODV levels off but keeps increasing under our scheme. This apparent paradox is explained by Fig. 7. In AODV, heavy packet loss occurs after a data rate of 0.3 units/sec due to congestion at bottleneck nodes. A benign leveling off of the number of transmissions under AODV actually masks an underlying disaster. In contrast, packets are routed more evenly in the network under our scheme, fewer packets are dropped, and much higher throughput is achieved. As deadlines get less stringent, packet transmissions are further reduced, since packets can afford more delays at the relaying nodes and more packets are likely to be grouped together.

1) **Allocating Hold Times***:* Fig. 8 compares our hold time allocation algorithm (LBR-G) with two other hold time allocation approaches, namely *UNIFORM* and *SRC*, under two df values. Each node generated data at the same rate, shown on the x-axis. UNIFORM distributes the available slack time uniformly across all the relaying nodes along each path to the BS, while SRC allocates all available slack time to the source node. Clearly, our scheme incurs much fewer packet transmissions than the others. For example, when $df = 2$, our scheme transmits about 45% fewer packets than SRC, and about 40% fewer packets than UNIFORM at the rate $1\ units/sec$. As df increases, the difference

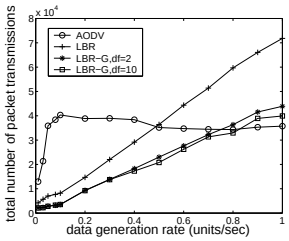


Fig. 6. Number of packets compared

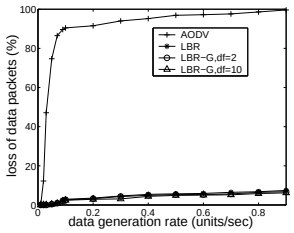


Fig. 7. Packet loss compared

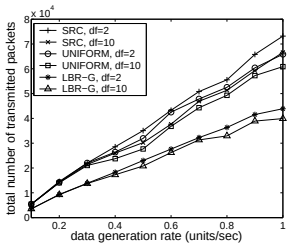


Fig. 8. Hold time allocation schemes compared

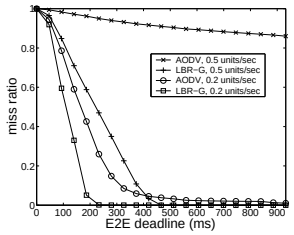


Fig. 9. E2E deadline miss ratio

between LBR-G and UNIFORM/SRC becomes less significant, since more packets will reach the maximum payload size with less stringent deadlines under all schemes.

2) **Meeting Real-time Requirements***:* In Fig. 9, we show the fraction of packets missing their deadlines for data generation rates of 0.2 units/sec and 0.5 units/sec. Each sensor reading was associated with the same deadline. When the deadlines are so stringent that the slack time is negative, most packets will miss their deadline under both schemes, and the LBR-G is reduced to LBR. However, the miss ratio drops rapidly for LBR-G as the deadlines are less stringent, and LBR-G always has a lower miss ratio than AODV. When the traffic load is high (0.5 units/sec), the advantage of LBR-G is more dramatic, since both load balanced routing and packet aggregation contributes to lower E2E delays.

SPEED [2] reported their miss ratio in a network of 100 sensors, of which only 6 generated traffic, with the others presumably serving as relaying nodes. With a total data generation rate of 20 packets/sec and E2E deadline of 200 ms, SPEED shows a deadline miss ratio of around 2%, compared with 12% for AODV. Our scheme shows roughly the same miss ratio as SPEED for the same total data generation rate and deadline. However, we run our experiments in a more demanding set of conditions, in which *all* nodes in the system generate packets. As a result, AODV shows a miss ratio of nearly 30% in our context. This shows that our scheme is more effective than SPEED in comparable conditions.

3) **Power Savings***:* We measured power consumption at each node. Fig. 10 shows the power consumed per on-time sensor reading for AODV and LBR-G. This metric reflects how well each scheme can meet real-time requirements in a power-efficient way.

Our results clearly show that the power consumed per on-time sensor reading is far lower for

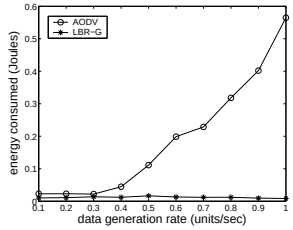


Fig. 10. Energy Consumed per on-time sensor reading (deadline: $500\,ms$)

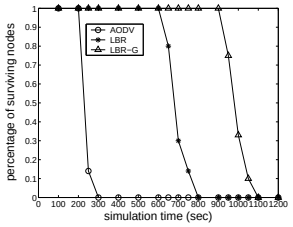


Fig. 11. Fraction of surviving nodes (deadline: $500\,ms$)

LBR-G. AODV power consumption increases drastically as the data rates increase, while it remains extremely stable for LBR-G. There are two reasons for this effect. First, AODV requires more power because it needs more MAC layer transmissions. Second, many packets fail to meet their deadlines under AODV than under LBR-G (see Fig. 9).

Extending the network lifetime is an important goal for a real-time sensor network. In Fig. 11, we compare the fraction of surviving nodes under AODV and our scheme, after a given simulation time. The initial energy level was set to $5\,J$. The survivor fraction starts to drop rapidly from $200\,sec$ under AODV, while it remains 100% under both LBR and LBR-G. The LBR scheme achieves 150% longer network lifetime than AODV, and LBR-G achieves 58% longer lifetime than LBR, which suggests that both the load-balanced routing scheme and the packet aggregation scheme contribute significantly to extending the network lifetime.

VII. Conclusions

We presented a power-efficient scheme to deliver real-time data in sensor networks. We proposed a novel load-balanced routing scheme (LBR), in which packets can take multiple paths to the BS. LBR distributes data traffic very evenly over nodes at each level to avoid congestion and improve E2E transmission delays. We introduced a packet aggregation scheme over LBR, which allows sensor data units to be held at the intermediate nodes and grouped to form larger packets and reduce transmis-

sions. Larger packets are more power-efficient than many small packets, since channel contention is lower. We proposed an algorithm to determine hold times at the relaying nodes based on E2E delays.

Our simulation results show that LBR can significantly reduce the E2E transmission delays compared with AODV, which means we can achieve more stringent real-time requirements under LBR. Our packet aggregation scheme can further reduce packet transmissions in the network, thus saving more power for sensors.

ACKNOWLEDGMENT

This work was supported by a grant from Tata Consultancy Services, Inc.

REFERENCES

[1] C. Lu, B. M. Blum, T. F. Abdelzaher, S. John A, and T. He, "RAP: A real-time communication architecture for large-scale wireless sensor networks," in *Proc. of the 8th IEEE RTAS Symposium*, San Jose, September 2002.

[2] T. He, J. A. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A stateless protocol for real-time communication in sensor networks," in *Proc. of the 23rd International Conference on Distributed Computing Systems (ICDCS)*, Providence, Rhode Island, May 2003.

[3] X. Liu, Q. Wang, L. Sha, and W. He, "Optimal QoS sampling frequency assignment for real-time wireless sensor networks," in *Proc. of the 24th IEEE RTSS Symposium*, Cancun, Mexico, December 2003.

[4] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. of the 33rd Hawaii Intl. Conf. on System Sciences*, 2000.

[5] S. Madden and M. J. Franklin, "Fjording the stream: An architecture for queries over streaming sensor data," in *Proc. of the 18th ICDE Conf*, San Jose, 2002.

[6] G. S. Ahn, A. T. Campbell, A. Veres, and L. H. Sun, "SWAN: Service differentiation in stateless wireless ad hoc networks," in *Proc. of the 21st IEEE INFOCOM*, New York, June 2002.

[7] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazzo, "An implicit prioritized access protocol for wireless sensor networks," in *Proc. of the 23rd IEEE Real-Time Systems Symposium (RTSS)*, Austin, Texas, December 2002.

[8] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: A tiny aggregation service for ad-hoc sensor networks," in *Proc. of OSDI*, December 2002.

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. of MobiCom*, August 2000.

[10] A. Manjhi, S. Nath, and P. B. Gibbons, "Tributaries and deltas: Efficient and robust aggregation in sensor network streams," in *Proc. of the ACM SIGMOD*, June 2005.

[11] B. R. Badrinath and P. Sudame, "Gathercast: The design and implementation of a programmable aggregation mechanism for the internet," in *Proc. of the 9th International Conference on Computer Communications and Networks*, October 2000.

[12] T. He, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Aida: Adaptive application-independent data aggregation in wireless sensor networks," in *ACM Transactions on Embedded Computing Systems*, vol. 3(2), May 2004.

[13] O. Chipara, C. Lu, and G.-C. Roman, "Efficient power management based on application timing semantics for wireless sensor networks," in *Proc. of the 25th ICDCS*, June 2005.

[14] C. E. Perkins, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proc. of the ACM SIGCOMM'94*, 1994.

[15] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*. Kluwer Academic Publishers, 1996, vol. 353.

[16] C. E. Perkins and E. M. Royer, "Ad hoc on-demand distance vector routing," in *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, Feburary 1999.

[17] P. H. Hsiao, A. Hwang, H.T.Kung, and D. Vlah, "Load-balancing routing for wireless access networks," in *Proc. of the 20th IEEE INFOCOM*, Anchorage, Alaska, April 2001.

[18] S. C. Huang and R. H. Jan, "Energy-aware load balanced routing schemes for sensor networks," in *Proc. of the 10th Intl Conference on Parallel and Distributed Systems*, Newport Beach, California, July 2004.

[19] X. Hong, M. Gerla, W. Hanbiao, and L. Clare, "Load balanced, energy-aware communications for mars sensor networks," in *Proc. of the Aerospace Conference, vol 3*, 2002.

[20] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. McGraw-Hill Book Company, 1997.

[21] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis, "TINA: A scheme for temporal coherency-aware in-network aggregation," in *Proc. of the 3rd ACM MobiDE Workshop*, September 2003.

[22] S.Zhu, W.Wang, C.V.Ravishankar: PERT: A New Power-Efficient Real-Time Packet Delivery Scheme for Sensor Networks, Technical Report, Univ. of California, Riverside (2006), http://www.cs.ucr.edu/∼szhu/pert.pdf.

[23] "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," in *IEEE 802.11 Standard, 1997*.

[24] L. Kleinrock, *Queueing Systems: Theory, Volume 1*. John Wiley and Sons, 1975.

[25] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proc. of the 21st INFOCOM*, June 2002.

[26] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier methods*. Belmont, Mass. : Athena Scientific, 1996.

[27] S. McCanne and S. Floyd., "ns network simulator," http://www.isi.edu/nsnam/ns/.

[28] T. S. Rappaport, *Wireless Communications, Principles and Practice*. Prentice Hall, 1996.

[29] "Chipcon CC1000 RF transceiver datasheet," http://www.chipcon.com, april, 2004.

[30] "MPR/MIB mote sensor hardware users manual," http://www.xbow.com/Support/manuals.